Learning Pareto Set for Multi-Objective Continuous Robot Control

Tianye Shu[1] , **Ke Shang**[*1,2] , **Cheng Gong**[1,3] , **Yang Nan**[1] and **Hisao Ishibuchi**[*1]
[1]Department of Computer Science and Engineering, Southern University of Science and Technology
[2]National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University
[3]Department of Computer Science, City University of Hong Kong
12132356@mail.sustech.edu.cn, kshang@foxmail.com,
{12150059, nany}@mail.sustech.edu.cn, hisao@sustech.edu.cn

Abstract

For a control problem with multiple conflicting objectives, there exists a set of Pareto-optimal policies called the Pareto set instead of a single optimal policy. When a multi-objective control problem is continuous and complex, traditional multi-objective reinforcement learning (MORL) algorithms search for many Pareto-optimal deep policies to approximate the Pareto set, which is quite resource-consuming. In this paper, we propose a simple and resource-efficient MORL algorithm that learns a continuous representation of the Pareto set in a high-dimensional policy parameter space using a single hypernet. The learned hypernet can directly generate various well-trained policy networks for different user preferences. We compare our method with two state-of-the-art MORL algorithms on seven multi-objective continuous robot control problems. Experimental results show that our method achieves the best overall performance with the least training parameters. An interesting observation is that the Pareto set is well approximated by a curved line or surface in a high-dimensional parameter space. This observation will provide insight for researchers to design new MORL algorithms.

1 Introduction

In many real-world control problems, we often have multiple conflicting objectives. For example, in walking robot control, we may need to maximize walking speed and minimize energy consumption. Instead of a single optimal control policy, such a multi-objective control problem has a set of Pareto-optimal control policies called the Pareto set. Each Pareto-optimal policy in the Pareto set corresponds to a different trade-off (i.e., a different user preference) over conflicting objectives. To find these Pareto-optimal policies, multi-objective reinforcement learning (MORL) algorithms are widely used [Roijers *et al.*, 2013; Hayes *et al.*, 2022; Rame *et al.*, 2023].

*Corresponding authors.

Many MORL algorithms [Van Moffaert and Nowé, 2014; Parisi *et al.*, 2014; Xu *et al.*, 2020] search for a finite set of independent policies to approximate the Pareto set. These algorithms have two potential drawbacks when handling continuous multi-objective control problems. One drawback is that a finite number of policies cannot represent the entire Pareto set. Thus, the following undesirable situation can happen: No policy in the obtained finite policy set is close to the user preference. The other drawback is that these methods are memory-consuming when the control problem has many objectives. This is because the number of required policies to approximate the Pareto set increases exponentially as the number of objectives increases. For example, more than 1,000 Pareto-optimal deep policies (represented by deep neural networks) are stored by an evolutionary learning algorithm for a three-objective robot control problem [Xu *et al.*, 2020].

One idea to address these drawbacks is to train a single deep neural network to represent the whole Pareto set [Pirotta *et al.*, 2015; Yang *et al.*, 2019; Basaklar *et al.*, 2023]. Based on this idea, two different types of methods have been proposed: embedding-based method and manifold-based method. The embedding-based methods learn a generalized policy whose output is conditioned on the input preference [Yang *et al.*, 2019; Abels *et al.*, 2019; Basaklar *et al.*, 2023]. These algorithms are usually off-policy algorithms with a specifically designed experience replay strategy. The Bellman equation is extended into a multi-objective version by including the user preference [Yang *et al.*, 2019; Basaklar *et al.*, 2023]. To improve the sample efficiency, Alegre et al. [2023] proposed a novel preference selection strategy based on Generalized Policy Improvement (GPI) [Barreto *et al.*, 2020]. The manifold-based methods learn a parametric function to directly approximate the Pareto set as a manifold in the policy parameter space. For example, Pirotta et al. [2015] proposed a gradient-based approach called PMGA. Parisi et al. [2017] improved PMGA by using the importance sampling technique [Owen and Zhou, 2000]. However, these manifold-based approaches have not yet been applied to deep policies with tens of thousands of parameters. Two difficulties have been pointed out for these manifold-based methods in the literature: (1) The number of required parameters to represent the Pareto set is much more than (e.g., quadratically increases with) the dimensionality of the parameter space [Chen *et al.*, 2019; Basaklar *et al.*, 2023], and (2) the

Pareto set in the parameter space cannot be efficiently represented by a single continuous policy family [Xu *et al.*, 2020].

In this paper, we propose a manifold-based approach called Hyper-MORL to learn the Pareto sets (i.e., to search for Pareto-optimal deep policies) of multi-objective continuous control problems. Our assumption is that the Pareto set in a high-dimensional parameter space could be well approximated by a continuous manifold in a low-dimensional subspace. This assumption comes from the following property of multi-objective optimization: In general, the Pareto set of an m-objective problem is an $(m-1)$-dimensional manifold in an n-dimensional parameter space even if $n \gg m$ [Hillermeier, 2001]. Based on this assumption, we propose an on-policy MORL algorithm to learn a hypernet that maps a user preference (represented by a direction vector) to a Pareto-optimal policy (represented by a point) in the parameter subspace. We examine our method on several complex multi-objective robot control problems [Xu *et al.*, 2020] which are usually solved by deep policies with tens of thousands of parameters. Compared with two state-of-the-art MORL algorithms [Xu *et al.*, 2020; Basaklar *et al.*, 2023], our method shows the best overall performance with the least training parameters. To validate our initial assumption, we investigate the Pareto set learned by our method. We show that the Pareto set of continuous robot control problems can be well approximated by a single continuous policy family.

2 Background

2.1 Multi-Objective Markov Decision Process

A multi-objective RL problem (e.g., multi-objective continuous robot control problem) is usually modeled as a multi-objective Markov decision process (MOMDP). An MOMDP is described by the tuple $\langle S, A, P, \boldsymbol{R}, \gamma \rangle$ where S, A, $P(s'|s,a)$, $\boldsymbol{R}$ and γ represent state space, action space, probability transition function, reward function and discount factor, respectively. Here the reward function $\boldsymbol{R}$: $S \times A \rightarrow \mathbb{R}^m$ returns a reward vector $\boldsymbol{r} = (r_1, r_2, ..., r_m)$ where m is the number of objectives.

For an MOMDP, a policy $\pi(a|s)$ represents the probability of selecting the action $a \in A$ at the state $s \in S$. In each time step t, an agent with the policy π receives a state $s^{(t)}$ and selects an action $a^{(t)} \sim \pi(a|s^{(t)})$. Then, a reward vector $\boldsymbol{r}^{(t+1)} = \boldsymbol{R}(s^{(t)}, a^{(t)})$ and the next state $s^{(t+1)} \sim P(s|s^{(t)}, a^{(t)})$ are given by the MOMDP. In this manner, we can obtain a trajectory $\tau = (s^{(0)}, a^{(0)}, \boldsymbol{r}^{(1)}, ..., s^{(T-1)}, a^{(T-1)}, \boldsymbol{r}^{(T)})$ where $s^{(0)}$ is the initial state and T is the time horizon. The expected returns $\boldsymbol{J}(\pi) = (J_1(\pi), J_2(\pi), ..., J_m(\pi))$ is defined as:

$$J_i(\pi) = E_\tau[\sum_{t=1}^{T} \gamma^{t-1} r_i^{(t)}],\ i = 1,\ 2,\ ...,\ m, \qquad (1)$$

where $r_i^{(t)}$ is the i-th component of the reward vector $\boldsymbol{r}^{(t)}$.

Our target is to find a set of Pareto-optimal policies (called Pareto set) to maximize $\boldsymbol{J}$. Here the optimality is defined by the Pareto dominance relation as follows.

Definition1 (Pareto Dominance Relation)**.** *A policy π is said to be dominated by another policy π' (i.e., $\pi' \succ \pi$) iff $\forall i \in \{1, ..., m\}$, $J_i(\pi') \geq J_i(\pi)$ and $\exists j \in \{1, ..., m\}$, $J_j(\pi') > J_j(\pi)$.*

In this paper, we assume a policy $\pi(a|s, \theta)$ (π_θ for short) is parameterized by $\theta \in \Theta \subseteq \mathbb{R}^n$ where Θ is the parameter space. The Pareto set $PS(\Theta)$ in the parameter space is defined as

$$PS(\Theta) = \{\theta \in \Theta | \nexists \theta' \in \Theta, \pi_{\theta'} \succ \pi_\theta\}. \qquad (2)$$

The image of $PS(\Theta)$ in the objective space is Pareto front.

2.2 Hypernets

Hypernets are a type of neural networks that take a context vector as input and generate the parameters for the target neural networks [Chauhan *et al.*, 2023]. Compared with traditional neural networks, hypernets can be trained in a single-model manner to generate weights for multiple neural networks for solving related tasks [von Oswald *et al.*, 2020]. Due to its parameter-efficiency and expressiveness [Galanti and Wolf, 2020], hypernets have been used in many fields such as reinforcement learning [Sarafian *et al.*, 2021]. A task-conditioned hypernet is used for generalization across tasks in meta-RL [Beck *et al.*, 2023]. Besides, hypernets have been used in continual RL [Huang *et al.*, 2021] and zero-shot learning [Rezaei-Shoshtari *et al.*, 2023] in RL.

Recently, hypernets have been used to learn the Pareto set for many multi-objective problems such as multi-objective combinatorial optimization [Lin *et al.*, 2022] and multi-task learning [Navon *et al.*, 2021; Lin *et al.*, 2019]. In this paper, we use a hypernet to learn the Pareto set $PS(\Theta)$ of an MOMDP.

3 Proposed Method: Hyper-MORL

To solve a multi-objective RL problem, we propose an efficient manifold-based algorithm called Hyper-MORL.

3.1 Problem Decomposition

For multi-objective optimization, decomposition is a commonly-used idea [Zhang and Li, 2007]. A multi-objective RL (MORL) problem can be decomposed into several single-objective subproblems using a preference set $\{\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, ..., \boldsymbol{\omega}^{(k)}\}$ and a scalarization function f. Each subproblem with preference $\boldsymbol{\omega}$ is a single-objective RL problem $SORL(\boldsymbol{\omega})$ which aims to maximize a scalar expected return $f(\boldsymbol{J}, \boldsymbol{\omega})$.

The scalarization function f has two types: linear and non-linear. The linear scalarization function $f_{LS}(\boldsymbol{J}, \boldsymbol{\omega}) = \boldsymbol{\omega}^T \boldsymbol{J}$ is straightforward and widely used in many MORL algorithms [Parisi *et al.*, 2014; Xu *et al.*, 2020]. However, one limitation of f_{LS} is that it can only find a convex hull, which is a subset of the Pareto set [Hayes *et al.*, 2022]. Non-linear scalarization functions can find the whole Pareto set regardless of the Pareto front shape. However, the use of non-linear scalarization functions in RL is difficult since the assumed additive property in the Bellman equation does not hold for the non-linearly scalarized returns [Roijers *et al.*, 2018; Hayes *et al.*, 2022].

In this paper, we consider the linear scalarization function since Lu et al. [2023] theoretically revealed that a MORL

problem is convex (i.e., the range of the value function is convex) when all stationary policies are considered. Given a preference space $\Omega = \{\boldsymbol{\omega} | \boldsymbol{\omega} \in (\{0\} \cup \mathbb{R}^+)^m, \sum_{i=1}^m \omega_i = 1\}$, the following proposition holds.

Proposition1 [Lu *et al.*, 2023]**.** *For a convex MORL problem, a policy $\pi \in PS(\Theta)$ iff $\exists \boldsymbol{\omega} \in \Omega$, π is optimal for the problem $SORL(\boldsymbol{\omega})$ which maximizes $\boldsymbol{\omega}^T \boldsymbol{J}(\pi)$.*

Proposition 1 says that we can find the whole Pareto set by solving the problem $SORL(\boldsymbol{\omega})$ for each preference $\boldsymbol{\omega} \in \Omega$.

3.2 Pareto Set Representation via Hypernet

The basic idea of Hyper-MORL is shown in Figure 1. To represent the Pareto set, we train a model to map each preference $\boldsymbol{\omega} \in \Omega$ to the corresponding Pareto-optimal policy parameter $\boldsymbol{\theta} \in \Theta$ which maximizes $\boldsymbol{\omega}^T \boldsymbol{J}(\boldsymbol{\theta})$. To reduce the number of the required parameters, we consider a reduced d-dimensional subspace Θ_{sub} instead of the original n-dimensional parameter space Θ to represent the Pareto set ($d \ll n$). The value of d is specified as 10 in this paper (and some other values are also examined for sensitivity analysis). A hypernet $\mathcal{H}_{\boldsymbol{\varphi}}: \Omega \to \Theta$ is defined as:

$$\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega}) = \boldsymbol{W} f_{\boldsymbol{\mu}}(\boldsymbol{\omega}) + \boldsymbol{b}. \tag{3}$$

Here $f_{\boldsymbol{\mu}}$ maps the preference $\boldsymbol{\omega}$ to a d-dimensional vector. The parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ are an $n \times d$ matrix and an n-dimensional vector, respectively. The row vectors of $\boldsymbol{W}$ spans a d-dimensional parameter subspace Θ_{sub}. The vector $f_{\boldsymbol{\mu}}(\boldsymbol{\omega})$ is linearly transformed from Θ_{sub} to Θ. The learning parameters $\boldsymbol{\varphi}$ of the hypernet $\mathcal{H}_{\boldsymbol{\varphi}}$ are $\{\boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{b}\}$. A detailed illustration of $\mathcal{H}_{\boldsymbol{\varphi}}$ is included in Appendix[1] B.

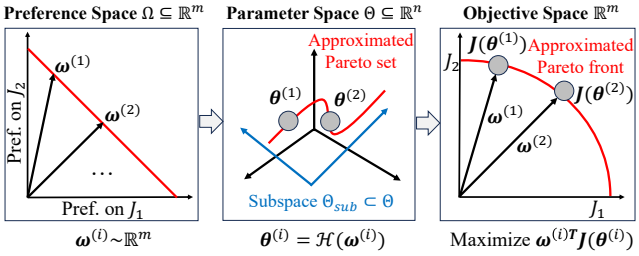


Figure 1: Basic idea of Hyper-MORL to approximate the Pareto set in the n-dimensional parameter space.

3.3 Pareto Set Learning via Policy Gradient

To efficiently learn the Pareto set, a gradient-based algorithm is proposed for Hyper-MORL. As shown in Algorithm 1, the learning algorithm contains warm-up stage and Pareto set learning stage. As shown in lines 1-2, $\alpha \times 100\%$ computation load is assigned to the warm-up stage where α is a parameter. In the following part, we will describe these two stages.

[1]https://github.com/HisaoLabSUSTC/Hyper-MORL/blob/main/Appendix.pdf

Algorithm 1: Learning Pareto Set via Policy Gradient

Input: Hypernet $\mathcal{H}_{\boldsymbol{\varphi}}: \Omega \to \Theta$ with parameters $\boldsymbol{\varphi} = \{\boldsymbol{W}, \boldsymbol{\mu}, \boldsymbol{b}\}$, the number of sampled preferences K, the number of available environment steps T, parameter α

1 $G_W = \lfloor \frac{\alpha T}{T_{tra}} \rfloor$;
2 $G_{PSL} = \lfloor \frac{(1-\alpha)T}{K T_{tra}} \rfloor$; // T_{tra} is the number of consumed environment steps to collect a trajectory
3 // Warm-up Stage
4 $\boldsymbol{W} \leftarrow \boldsymbol{0}$;
5 Set $\boldsymbol{\mu}$ and $\boldsymbol{b}$ by an arbitrary initialization scheme;
6 $\boldsymbol{\omega} = (\frac{1}{m}, ..., \frac{1}{m})$;
7 **for** $g \leftarrow 1$ **to** G_W **do**
8 Run policy $\pi_{\boldsymbol{b}}$ in environment and collect a trajectory τ;
9 Calculate gradients $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{b}}[\boldsymbol{\omega}^T \boldsymbol{J}(\boldsymbol{b})]$;
10 Update parameters $\boldsymbol{b} \leftarrow \textbf{ADAM}(\boldsymbol{b}, \boldsymbol{g})$;
11 Reset exploration-related parameters in $\boldsymbol{b}$;
12 // Pareto Set Learning Stage
13 **for** $g \leftarrow 1$ **to** G_{PSL} **do**
14 Sample K preferences $\boldsymbol{\omega}^{(1)}, ..., \boldsymbol{\omega}^{(K)}$ from Ω;
15 **for** $i \leftarrow 1$ **to** K **do**
16 Run policy $\pi_{\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega}^{(i)})}$ in environment and collect a trajectory $\tau^{(i)}$;
17 **for** $i \leftarrow 1$ **to** K **do**
18 Calculate gradients $\boldsymbol{g}^{(i)} \leftarrow \nabla_{\boldsymbol{\varphi}}[\boldsymbol{\omega}^{(i)T} \boldsymbol{J}(\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega}^{(i)}))]$
19 Update parameters $\boldsymbol{\varphi} \leftarrow \textbf{ADAM}(\boldsymbol{\varphi}, \frac{1}{K} \sum_{i=1}^K \boldsymbol{g}^{(i)})$;

Warm-up Stage

The warm-up stage is to find a good initial policy close to the Pareto front. By starting from a good initial policy instead of a random policy, the Pareto set learning stage can be accelerated. In Algorithm 1, the warm-up stage is in lines 3-11. In lines 4-5, we use the Bias-HyperInit method [Beck *et al.*, 2023] to initialize the hypernet parameters. In lines 6-10, we update parameters $\boldsymbol{b}$ to optimize all objectives simultaneously with the preference $(\frac{1}{m}, ..., \frac{1}{m})$ (m is the number of objectives). The multi-objective policy gradient method [Xu *et al.*, 2020] is used as follows:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \left[\boldsymbol{\omega}^T \boldsymbol{J}(\boldsymbol{\theta})\right] &= \sum_{i=1}^m \left[\omega_i \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta})\right] \\ &= \mathbb{E}\left[\sum_{t=0}^T \left[\boldsymbol{\omega} \boldsymbol{A}(s_t, a_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t)\right]\right], \end{aligned} \tag{4}$$

where $\boldsymbol{A}$ is the advantage function estimated from a sampled trajectory τ.

After G_W iterations, we obtain a policy $\pi_{\boldsymbol{b}}$ from the preference $(\frac{1}{m}, ..., \frac{1}{m})$. In our implementation, the policy $\pi_{\boldsymbol{b}}$ samples an action from a normal distribution. Some values in the n-dimensional vector $\boldsymbol{b}$ represent the standard deviation

of the output action. In line 11, the exploration-related parameters (i.e., the standard deviation of the output action) are reset as one. Since other values in $\boldsymbol{b}$ are not changed, the mean of the output action is preserved. The reset step is to prevent the output action selection from being almost deterministic (i.e., standard deviation from being almost zero) which severely degrades the exploration ability in the Pareto set learning stage. After the warm-up stage, $\boldsymbol{b}$ becomes closer to the Pareto set, and $\boldsymbol{W}$ remains $\mathbf{0}$. From Eq. (3), the hypernet $\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega}) = \boldsymbol{b}$ for $\forall \boldsymbol{\omega} \in \Omega$. This means that the initial values of $\boldsymbol{b}$ are shared among all subproblems.

Pareto Set Learning Stage

Our task is to find the optimal parameters $\boldsymbol{\varphi}$ (i.e., $\boldsymbol{W}$, $\boldsymbol{\mu}$ and $\boldsymbol{b}$) which maximizes $\boldsymbol{\omega}^T \boldsymbol{J}(\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega}))$ for each $\boldsymbol{\omega} \in \Omega$. The objective function can be written as:

$$\mathbb{E}_{\boldsymbol{\omega} \sim \Omega} \left[\boldsymbol{\omega}^T \boldsymbol{J}(\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega})) \right]. \quad (5)$$

It is difficult to directly optimize the parameters $\boldsymbol{\varphi}$ to maximize the exception in Eq. (5). Thus, we use Monte Carlo method to sample the preferences, and use the policy gradient method to optimize the parameters $\boldsymbol{\varphi}$. In Algorithm 1, lines 13-19 show the Pareto set learning stage. In each iteration, K preferences are randomly sampled from the preference space Ω. Then, we obtain K policies by inputting these K preferences into the hypernet. In lines 15-16, each policy interacts with the environment and generates a trajectory. In lines 17-18, the gradient $\boldsymbol{g}^{(i)}$ is calculated for each preference based on the corresponding trajectory. Then, the parameters $\boldsymbol{\varphi}$ are updated based on the average gradient over all sampled preferences (i.e., $\frac{1}{K} \sum_{i=1}^{K} \boldsymbol{g}^{(i)}$).

The remaining question is how to calculate the gradient $\boldsymbol{g}^{(i)}$ in line 18. By the chain rule, we have

$$\nabla_{\boldsymbol{\varphi}} \left[\boldsymbol{\omega}^T \boldsymbol{J}(\mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega})) \right] = \nabla_{\boldsymbol{\varphi}} \left[\nabla_{\boldsymbol{\theta}} \left[\boldsymbol{\omega}^T \boldsymbol{J}(\boldsymbol{\theta}) \right] \cdot \mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega}) \right], \quad (6)$$

where $\boldsymbol{\theta} = \mathcal{H}_{\boldsymbol{\varphi}}(\boldsymbol{\omega})$ and $\cdot$ means dot product. As we can see, the term $\nabla_{\boldsymbol{\theta}} \left[\boldsymbol{\omega}^T \boldsymbol{J}(\boldsymbol{\theta}) \right]$ can be calculated by Eq. (4). Thus, the gradient $\boldsymbol{g}^{(i)}$ can be calculated by Eq. (4) and Eq. (6). After the Pareto set learning stage, the whole Pareto set $PS(\Theta)$ is obtained from the hypernet $\mathcal{H}_{\boldsymbol{\varphi}}$.

4 Experiments

4.1 Benchmark Problems

To test the performance of Hyper-MORL, we use seven problems in Table 1 from a multi-objective robot control benchmark suite [Xu *et al.*, 2020]. For more details, please refer to Appendix A.

4.2 Compared MORL Algorithms

We compare the proposed Hyper-MORL algorithm with the following two state-of-the-art MORL algorithms.

PG-MORL [Xu *et al.*, 2020]

Prediction-guided MORL (i.e., PG-MORL) algorithm is an evolutionary learning algorithm. In each generation, PG-MORL selects a set of policies from the current population based on a prediction model, and optimizes them with a multi-objective policy gradient method. All Pareto-optimal policies are stored in an unbounded external archive. We use the default settings in the original paper for PG-MORL[2].

Problem	m	State space	Action space
MO-Swimmer-v2	2	$\mathcal{S} \subseteq \mathbb{R}^8$	$\mathcal{A} \subseteq \mathbb{R}^2$
MO-HalfCheetah-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{17}$	$\mathcal{A} \subseteq \mathbb{R}^6$
MO-Walker2d-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{17}$	$\mathcal{A} \subseteq \mathbb{R}^6$
MO-Ant-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{27}$	$\mathcal{A} \subseteq \mathbb{R}^8$
MO-Hopper-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{11}$	$\mathcal{A} \subseteq \mathbb{R}^3$
MO-Humanoid-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{376}$	$\mathcal{A} \subseteq \mathbb{R}^{17}$
MO-Hopper-v3	3	$\mathcal{S} \subseteq \mathbb{R}^{11}$	$\mathcal{A} \subseteq \mathbb{R}^3$

Table 1: Seven test problems in a multi-objective robot control benchmark [Xu *et al.*, 2020].

Problem	Hyper-MORL	PG-MORL	PD-MORL
MO-Swimmer-v2	1.2×10^7	1.2×10^7	1×10^7
MO-Walker2d-v2	3×10^7	3×10^7	1×10^7
MO-HalfCheetah-v2	3×10^7	3×10^7	1×10^7
MO-Ant-v2	4.8×10^7	4.8×10^7	1×10^7
MO-Hopper-v2	4.8×10^7	4.8×10^7	1×10^7
MO-Humanoid-v2	1.2×10^8	1.2×10^8	1×10^7
MO-Hopper-v3	1.2×10^8	1.2×10^8	1×10^7

Table 2: The number of environment steps used by each algorithm for each problem. PG-MORL and PD-MORL follow the settings in their original papers [Xu *et al.*, 2020; Basaklar *et al.*, 2023].

PD-MORL [Basaklar *et al.*, 2023]

Preference-driven MORL (i.e., PD-MORL) algorithm learns a generalized policy by combining the preference into the Bellman equation. We use the default settings in the original paper for PD-MORL[3]. Note that the number of environment steps used by PD-MORL is smaller than that used by PG-MORL as shown in Table 2. However, PD-MORL needs more computation time than PG-MORL, which we will discuss in Section 4.3. We find it impractical for PD-MORL to use the same number of environment steps as in PG-MORL.

Hyper-MORL (Our Algorithm)

In our algorithm, we use the same neural network architecture as in PG-MORL. Thus, the parameter spaces of Hyper-MORL and PG-MORL are the same. For a fair comparison, Hyper-MORL uses the same policy gradient method (i.e., PPO [Schulman *et al.*, 2017]) as in PG-MORL. The optimizer is ADAM with learning rate $\eta = 5 \times 10^{-5}$. In Hyper-MORL, the parameters α and d are set as 0.15 (i.e., 15%) and 10, respectively. The number of sampled preferences K is set as 6 and 15 for two-objective and three-objective problems, respectively. For more details, please refer to Appendix C. All codes of Hyper-MORL are available from https://github.com/HisaoLabSUSTC/Hyper-MORL.

[2]https://github.com/mit-gfx/PGMORL

[3]https://github.com/tbasaklar/PDMORL-Preference-Driven-Multi-Objective-Reinforcement-Learning-Algorithm

Algorithm	Hyper-MORL (Ours)	PG-MORL	PD-MORL
MO-Swimmer-v2	$2.88\pm0.44 \times 10^4$ (2)	$2.60\pm0.71 \times 10^4$ (3)	$\mathbf{3.13 \pm 0.14 \times 10^4 (1)}$
MO-HalfCheetah-v2	$5.53\pm0.10\times10^6$ (3)	$5.75\pm0.02\times10^6$ (2)	$\mathbf{5.89 \pm 0.01 \times 10^6 (1)}$
MO-Walker2d-v2	$\mathbf{5.37 \pm 0.35 \times 10^6 (1)}$	$4.41\pm0.74\times10^6$ (2)	$5.08\pm0.23\times10^6$ (3)
MO-Ant-v2	$\mathbf{7.49 \pm 0.16 \times 10^6 (1)}$	$5.79\pm0.25\times10^6$ (3)	$7.05\pm1.13\times10^6$ (2)
MO-Hopper-v2	$\mathbf{2.05 \pm 0.06 \times 10^7 (1)}$	$1.96\pm0.17\times10^7$ (2)	$5.84\pm8.12\times10^6$ (3)
MO-Humanoid-v2	$4.32\pm0.42\times10^7$ (2)	$\mathbf{4.69 \pm 0.29 \times 10^7 (1)}$	$1.66\pm0.60\times10^7$ (3)
MO-Hopper-v3	$\mathbf{3.58 \pm 0.25 \times 10^{10} (1)}$	$3.31\pm0.23\times10^{10}$ (2)	$1.14\pm1.24\times10^{10}$ (3)
Average Rank	**1.57**	2.29	2.14

Table 3: HV-based performance comparison of Hyper-MORL with two state-of-the-art algorithms on a multi-objective continuous robot control benchmark [Xu *et al.*, 2020]. The reference point for HV calculation is set as (0, ..., 0). The average values and standard deviations over nine runs are reported. The rank of each method is shown in the parenthesis, and a small value means a better rank.

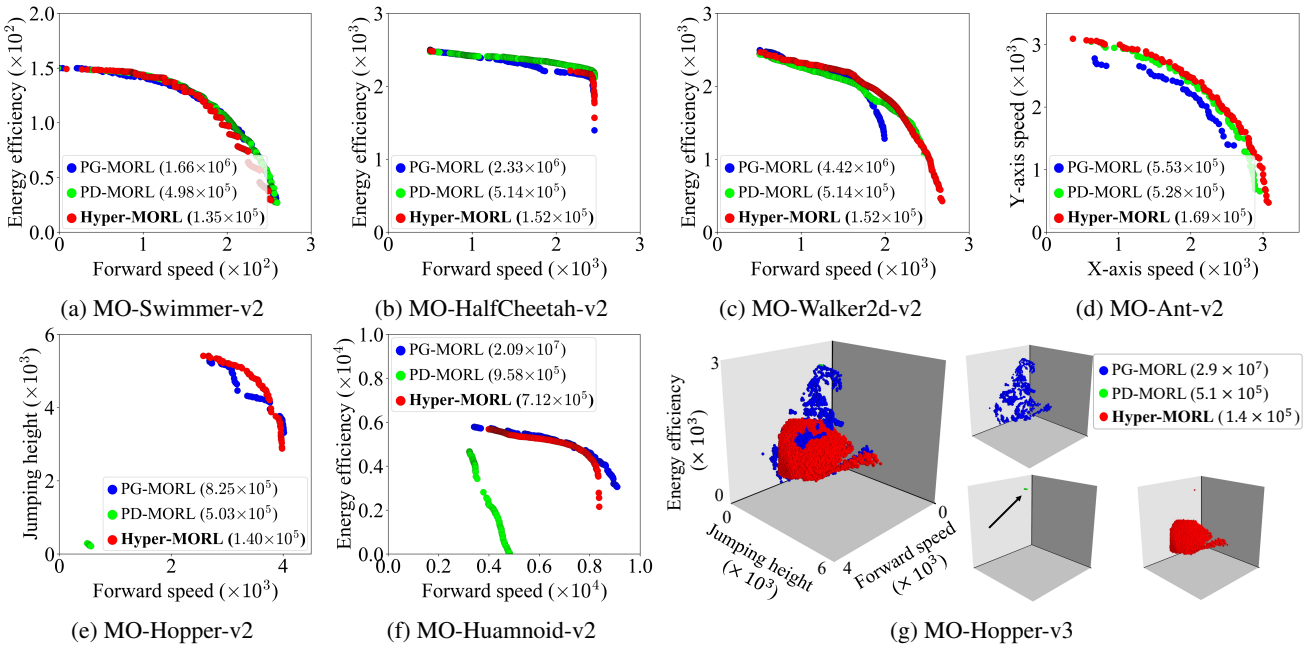


Figure 2: Visualization of all Pareto-optimal policies obtained by each MORL algorithm on each of six two-objective problems (a)-(f) and one three-objective problem (g). The total number of required parameters to represent these policies is shown in the parenthesis for each algorithm. A specific run with the median HV value among nine runs is shown for each algorithm.

4.3 Performance Comparison

We first compare the quality of the policy sets learned by the three algorithms. A widely-used indicator hypervolume (HV) [Zitzler *et al.*, 2003; Shang *et al.*, 2021] is used for the quality assessment, which is explained in Appendix E. For the evolutionary learning algorithm (i.e., PG-MORL), we evaluate the final population with population sizes 200 and 420 for 2-objective and 3-objective problems, respectively. For a fair comparison, we sample the same number of uniformly distributed preferences for each of the single model-based algorithms (i.e., PD-MORL and Hyper-MORL), and evaluate the output policies. Table 3 shows the HV results of each algorithm on the seven test problems. As we can see, Hyper-MORL has the best HV performance on 4 out of 7 problems and the worst only on MO-HalfCheetah-v2. Thus, Hyper-MORL has the best average rank in Table 3.

To test the ability of each algorithm to generate the continuous Pareto front, we use a single run with the median HV value for each algorithm. For PG-MORL, we plot all Pareto-optimal policies stored in the unbounded external archive. For Hyper-MORL and PD-MORL, we input a number of uniform preferences and plot all Pareto-optimal policies corresponding to these preferences. The number of input preferences is set as 2,000 and 200,100 for 2-objective and 3-objective problems, respectively. Figure 2 shows all Pareto-optimal policies from each algorithm on each test problem. Except for MO-HalfCheetah-v2, Hyper-MORL generates a continuous approximation of the Pareto front with high qual-

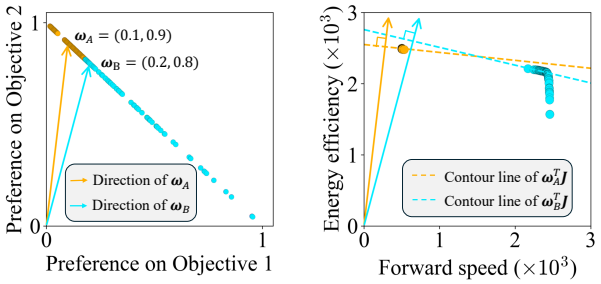


Figure 3: Explanation of the poor performance of Hyper-MORL on MO-HalfCheetah-v2 in Figure 2 (b). The input preferences (Left) and their corresponding Pareto-optimal policies (Right) are plotted in two colors. The preferences in yellow (blue) correspond to the policies in yellow (blue). Two similar preferences $\boldsymbol{\omega}_A$ and $\boldsymbol{\omega}_B$ lead to clearly two different policies in the right figure.

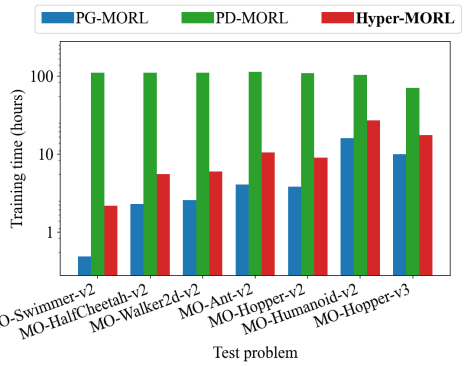
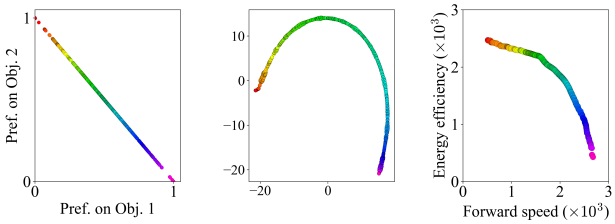


Figure 4: The average training time required for each algorithm on each test problem. The termination condition is shown in Table 2.



(a) Preference space (b) Parameter space (c) Objective space

Figure 5: Visualization of the Pareto-optimal policies obtained by Hyper-MORL for MO-Walker2d-v2 problem in the parameter space (b) and objective space (c), and their corresponding input to the hypernet in the preference space (a). Each policy is plotted with the same color as its corresponding input preference. t-SNE is used for visualization in the 11,150-dimensional parameter space.

ity for each problem. Especially for MO-Hopper-v3 in Figure 2 (g), Hyper-MORL generates a much more dense approximation than the other two methods. As shown in parenthesis in Figure 2, Hyper-MORL has the smallest number of parameters for each problem among three algorithms.

Hyper-MORL shows poor performance only on MO-HalfCheetah-v2 in Figure 2 (b). The reason is explained in Figure 3. As we can see, many preferences are mapped to the top left corner of the objective space by Hyper-MORL. This is because the linear scalarization function cannot find solutions on flat and concave regions of the Pareto front. The search for those regions by non-linear scalarization function is a future research topic.

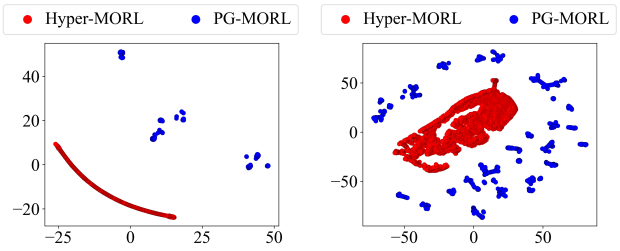
From Table 3 and Figure 2, we can see that PD-MORL (green points in Figure 2) has poor performance on MO-Hopper-v2, MO-Humanoid-v2, and MO-Hopper-v3, which could be due to two reasons. One reason is that the performance of PD-MORL highly relies on an interpolation procedure with some key solutions, as mentioned in the original paper [Basaklar *et al.*, 2023]. However, PD-MORL could not find any good key solutions for these problems. The other reason is that the number of environment steps (shown in Table 2) in PD-MORL may be insufficient. Figure 4 shows the training time of each algorithm. As we can see, PD-MORL needs the longest training time. For MO-Humanoid-v2, the training in PD-MORL for 1×10^7 environment steps needs about four days, while the training in Hyper-MORL and PG-MORL for 1.2×10^8 environment steps needs less than one day. If we use the same number of environment steps in all three algorithms, the training in PD-MORL will take about 40 days.

5 Further Studies

5.1 Investigation of the Learned Pareto Set

We further investigate the relation between the input preferences and the Pareto sets learned by Hyper-MORL. For better visualization, each preference is plotted with a different color in Figure 5 (a). Then, each policy in the learned Pareto set is plotted with the same color as its corresponding input preference in Figure 5 (b) and (c) for MO-Walker2d-v2. Dominated policies and the related preferences are not shown in Figure 5. We show the learned Pareto set in both the parameter space and the objective space. Since the dimension of the parameter space is high, we use t-SNE [Van der Maaten and Hinton, 2008] to visualize the high-dimensional parameter space in a two-dimensional space. Visualization results for the other problems are included in Appendix D.1. In Figure 5, we can observe a clear relation between the preferences in (a) and the Pareto front in (c). For example, the top-left solution in Figure 5 (c) with the largest second objective value is obtained from the preference (0, 1).

One interesting observation from Figure 5 (b) is that the shape of the Pareto set learned by Hyper-MORL is a simple curve after the dimensionality reduction from the 11,150-dimensional parameter space. This observation is different from reported results on PG-MORL [Xu *et al.*, 2020] where it was shown that a single continuous policy family cannot represent the Pareto set of these continuous robot control problems well. Thus, we further compare the two policy sets learned by Hyper-MORL and PG-MORL in the parameter space. PD-MORL is not considered since it has a different parameter space from Hyper-MORL and PG-MORL. Figure 6 shows the comparison results on the two-objective MO-Humanoid-v2 and three-objective MO-Hopper-v3 problems. As we can see, Hyper-MORL learns a single continuous pol-

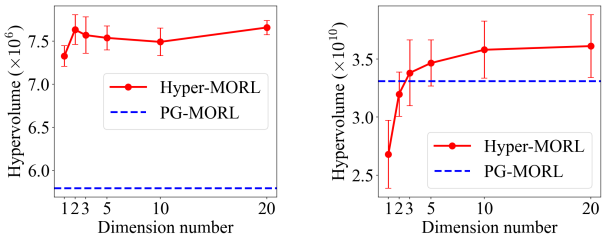


(a) MO-Humanoid-v2 (b) MO-Hopper-v3

Figure 6: Comparison of the policy sets learned by Hyper-MORL and PG-MORL in the parameter space on MO-Humanoid-v2 and MO-Hopper-v3. Figure 2 (f) and Figure 2 (g) are the corresponding visualization in the objective space for (a) and (b).

icy family in Figure 6 (a) and closely related policy families in Figure 6 (b) while PG-MORL learns a set of disjoint policy families.

As discussed in Section 3.2, the Pareto set learned by Hyper-MORL is located in a d-dimensional parameter subspace where d is a parameter for the hypernet used in Hyper-MORL. We examine the effects of d on the hypervolume performance of Hyper-MORL using six values of d: d =1, 2, 3, 5, 10, 20. Figure 7 shows the results of the two-objective MO-Ant-v2 and three-objective MO-Hopper-v3 problems. As a baseline, the hypervolume performance of PG-MORL is shown by the blue dotted line. In Figure 7 (a), the hypervolume performance of Hyper-MORL is better than PG-MORL even when $d = 1$. That means the Pareto set of MO-Ant-v2 is well approximated by Hyper-MORL with a curved line in the parameter space. For the three-objective MO-Hopper-v3 problem in Figure 7 (b), good results are obtained when $d > 2$. The results for the other problems are included in Appendix D.2. Our experimental results show that the Pareto sets of most problems are well approximated by Hyper-MORL in low-dimensional parameter subspace.



(a) MO-Ant-v2 (b) MO-Hopper-v3

Figure 7: Effects of the dimensionality d of the reduced parameter space on the hypervolume performance of Hyper-MORL. The red vertical line shows the standard deviation among nine runs.

5.2 Effects of the Warm-Up Stage

To examine the effects of the warm-up stage, we test four specifications of α: 0%, 5%, 10%, and 20%. $\alpha = 0\%$ means that the warm-up stage is not performed, and the hypernet is initialized by the Bias-HyperInit [Beck *et al.*, 2023]. All the other settings except for α are the same as in the previous experiments. For each specification (i.e., $\alpha = x\%$), we calculate the hypervolume improvement percentage (HVIP) from the results with $\alpha = 0\%$ as follows:

$$\text{HVIP} = \frac{\text{HV}_{\alpha=x\%} - \text{HV}_{\alpha=0\%}}{\text{HV}_{\alpha=0\%}} \times 100. \tag{7}$$

Here positive and negative HVIP values mean positive and negative effects of the warm-up stage with $\alpha = x\%$. The HVIP results are shown in Figure 8. We can see that the warm-up stage has a large positive effect on the hypervolume performance for MO-Swimmer and MO-Humanoid-v2 (i.e., more than 40% improvement). For the other problems, the positive and negative effects of the warm-up stage are small (less than 10%). Figure 8 also shows that too small value (e.g., 5%) or too large value (e.g., 20%) for the parameter α is not good. Our recommended value for α is 15%.

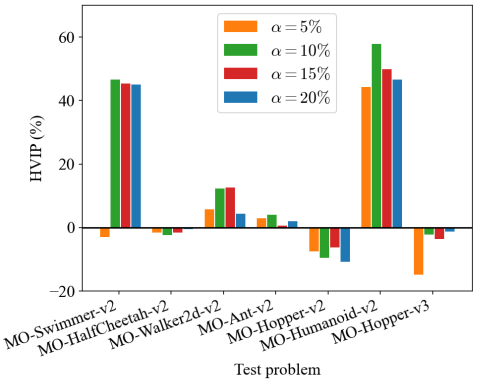


Figure 8: The hypervolume improvement percentage (HVIP) of Hyper-MORL by the warm-up stage with different α values.

6 Conclusions

In this paper, we proposed an MORL algorithm called Hyper-MORL, which learns a hypernet to approximate the Pareto set in a high-dimensional parameter space of deep policies by using a low-dimensional reduced parameter space. We compared Hyper-MORL with two state-of-the-art algorithms on seven multi-objective robot control problems. Our results showed that Hyper-MORL has the best overall performance and the least training parameters among compared algorithms. By the investigation of the Pareto set learned by Hyper-MORL, we found that the Pareto set of these multi-objective objective robot control problems can be well approximated by a manifold in a low-dimensional (e.g., 3-dimensional) subspace of the original high-dimensional (e.g., 11,150-dimensional) parameter space. Our results provide valuable insights for researchers to design new MORL algorithms. In the future, we try to combine a non-linear scalarization function into our method.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No. 62250710163, 62376115), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001).

References

[Abels *et al.*, 2019] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*, volume 97, pages 11–20, Jun 2019.

[Alegre *et al.*, 2023] Lucas N. Alegre, Ana L. C. Bazzan, Diederik M. Roijers, Ann Nowé, and Bruno C. da Silva. Sample-efficient multi-objective learning via generalized policy improvement prioritization. In *Autonomous Agents and Multi-Agent Systems*, page 2003–2012, 2023.

[Barreto *et al.*, 2020] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. In *Proceedings of the National Academy of Sciences*, 2020.

[Basaklar *et al.*, 2023] Toygun Basaklar, Suat Gumussoy, and Umit Ogras. PD-MORL: Preference-driven multi-objective reinforcement learning algorithm. In *International Conference on Learning Representations*, 2023.

[Beck *et al.*, 2023] Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Proceedings of The 6th Conference on Robot Learning*, volume 205, pages 1478–1487, 2023.

[Chauhan *et al.*, 2023] Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *arXiv preprint arXiv:2306.06955*, 2023.

[Chen *et al.*, 2019] Xi Chen, Ali Ghadirzadeh, Mårten Björkman, and Patric Jensfelt. Meta-learning for multi-objective reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 977–983. IEEE, 2019.

[Galanti and Wolf, 2020] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. In *Advances in Neural Information Processing Systems*, volume 33, pages 10409–10419, 2020.

[Hayes *et al.*, 2022] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

[Hillermeier, 2001] Claus Hillermeier. *Nonlinear multiobjective optimization: a generalized homotopy approach*, volume 135. Springer Science & Business Media, 2001.

[Huang *et al.*, 2021] Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 799–805, 2021.

[Lin *et al.*, 2019] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[Lin *et al.*, 2022] Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022.

[Lu *et al.*, 2023] Haoye Lu, Daniel Herman, and Yaoliang Yu. Multi-objective reinforcement learning: Convexity, stationarity and Pareto optimality. In *International Conference on Learning Representations*, 2023.

[Navon *et al.*, 2021] Aviv Navon, Aviv Shamsian, Ethan Fetaya, and Gal Chechik. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021.

[Owen and Zhou, 2000] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

[Parisi *et al.*, 2014] Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. Policy gradient approaches for multi-objective sequential decision making. In *International Joint Conference on Neural Networks*, pages 2323–2330, 2014.

[Parisi *et al.*, 2017] Simone Parisi, Matteo Pirotta, and Jan Peters. Manifold-based multi-objective policy search with sample reuse. *Neurocomputing*, 263:3–14, 2017.

[Pirotta *et al.*, 2015] Matteo Pirotta, Simone Parisi, and Marcello Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[Rame *et al.*, 2023] Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. In *Advances in Neural Information Processing Systems*, 2023.

[Rezaei-Shoshtari *et al.*, 2023] Sahand Rezaei-Shoshtari, Charlotte Morissette, Francois R. Hogan, Gregory Dudek, and David Meger. Hypernetworks for zero-shot transfer in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9579–9587, Jun. 2023.

[Roijers *et al.*, 2013] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[Roijers *et al.*, 2018] Diederik M Roijers, Denis Steckelmacher, and Ann Nowé. Multi-objective reinforcement

learning for the expected utility of the return. In *Proceedings of the Adaptive and Learning Agents Workshop at FAIM*, 2018.

[Sarafian *et al.*, 2021] Elad Sarafian, Shai Keynan, and Sarit Kraus. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*, volume 139, pages 9301–9312, 2021.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shang *et al.*, 2021] Ke Shang, Hisao Ishibuchi, Linjun He, and Lie Meng Pang. A survey on the hypervolume indicator in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):1–20, 2021.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

[Van Moffaert and Nowé, 2014] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[von Oswald *et al.*, 2020] Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020.

[Xu *et al.*, 2020] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International Conference on Machine Learning*, pages 10607–10616, 2020.

[Yang *et al.*, 2019] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

[Zitzler *et al.*, 2003] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.