Supplementary Material of "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

A Benchmark Problems

To test the performance of Hyper-MORL, we use a continuous robot control benchmark [Xu *et al.*, 2020]. As shown in Table 1, the benchmark contains six two-objective problems and one three-objective problem. Among them, MO-Humanoid-v2 has high-dimensional state space and action space.

B Network Architecture of Hyper-MORL

We show the network architecture of Hyper-MORL in Figure 1. Given a preference as input, a hypernet in Hyper-MORL outputs network parameters, which are then assigned to an actor network and a critic network. The obtained networks represent the optimal policy corresponding to the given preference.

The hypernet in Figure 1 first maps the preference $\boldsymbol{\omega}$ into a d-dimensional embedding vector. The mapping is represented by a fully connected neural network with two hidden layers. Each hidden layer has 128 units and activated by ReLU function. Then, the d-dimensional embedding vector is linearly transformed into a $(n_a + n_c)$-dimensional parameter vector, where n_a, n_c are the number of required parameters (i.e., weights and biases) for the actor network and the critic network, respectively.

In Figure 1, the actor network is a fully connected neural network with two hidden layers. The actor network inputs a state vector from the robot and outputs a vector whose length is twice the action space's size. The output vector represents the mean and the standard deviation of the action. The critic network is also a fully connected neural network with two hidden layers. The critic network inputs a state vector from the robot and outputs a vector whose length is the same as the number of objec-

Environment	m	State space	Action space	Objectives
MO-Swimmer-v2	2	$\mathcal{S} \subseteq \mathbb{R}^8$	$\mathcal{A} \subseteq \mathbb{R}^2$	forward speed, energy efficiency
MO-HalfCheetah-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{17}$	$\mathcal{A} \subseteq \mathbb{R}^6$	forward speed, energy efficiency
MO-Walker2d-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{17}$	$\mathcal{A} \subseteq \mathbb{R}^6$	forward speed, energy efficiency
MO-Ant-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{27}$	$\mathcal{A} \subseteq \mathbb{R}^8$	x-axis speed, y-axis speed
MO-Hopper-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{11}$	$\mathcal{A} \subseteq \mathbb{R}^3$	forward speed, jumping height
MO-Hopper-v3	3	$\mathcal{S} \subseteq \mathbb{R}^{11}$	$\mathcal{A} \subseteq \mathbb{R}^3$	forward speed, jumping height, energy efficiency
MO-Humanoid-v2	2	$\mathcal{S} \subseteq \mathbb{R}^{376}$	$\mathcal{A} \subseteq \mathbb{R}^{17}$	forward speed, energy efficiency

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

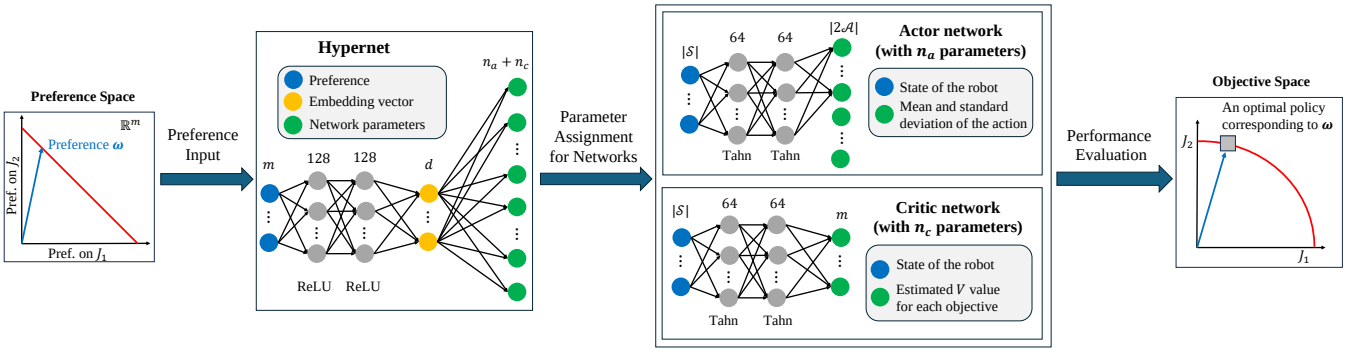


Figure 1: Network architecture of Hyper-MORL.

tives. The output vector represents the estimated values for different objectives. For both two networks, the size of the hidden layer is 64, and Tanh function is used for the activation.

C Experiment Setup Details

The experiments are conducted on a virtual machine equipped with four ADM EPYC 7702 240-Core CUP@2.4GHz, 256GB RAM and Ubuntu Operating System. Since the randomness affects the performance of MORL algorithms, we train neural networks nine runs with different random seeds for each algorithm. PG-MORL and PD-MORL use the default settings in their original papers, respectively. For Hyper-MORL, we use the same neural networks (i.e., the actor network and the critic network shown in Figure 1) as in PG-MORL to represent a policy. To optimize the parameters of the hypernet, we use the PPO [Schulman *et al.*, 2017] algorithm. Our PPO algorithm is implemented based on the codebase [Kostrikov, 2018] and the parameters of PPO are shown in Table 2. The hyper-parameters of Hyper-MORL include:

- α: the percentage of the used environment steps for the warm-up stage.
- K: the number of the sampled preferences in each iteration.
- d: the dimensionality of the reduced parameter space.

The settings for these parameters are shown in Table 3.

parameter name	value
learning rate	5×10^{-5}
timestesp per actorbatch	2048
num processes	4
gamma	0.995
gae lambda	0.95
ppo epoch	10
entropy coefficient	0
value loss coefficient	0.5

Table 2: Parameters of PPO

Environment	α	K	d
MO-Swimmer-v2	15%	6	10
MO-Walker-v2	15%	6	10
MO-HalfCheetah-v2	15%	6	10
MO-Ant-v2	15%	6	10
MO-Hopper-v2	15%	6	10
MO-Hopper-v3	15%	15	10
MO-Humanoid-v2	15%	6	10

Table 3: Hyper parameters for Hyper-MORL

D Additional Results

In this section, the full experimental results are given for all test problems, including the investigation of the learned Pareto set and the parameter analysis for the dimensionality of the reduced parameter space.

D.1 Investigation of the Learned Pareto Set

We investigate the relation between the input preferences and the Pareto set learned by Hyper-MORL. We use t-SNE to visualize the Pareto set in the high-dimensional parameter space. The perplexity of t-SNE is set as 50. Since t-SNE has randomness, we run t-SNE ten times and select the best run (i.e., the run with the lowest KL divergence). Figure 2 shows the results on seven test problems. We can observe a clear relation between the preferences in the preference space and the obtained policies in the objective space.

D.2 Dimensionality of the Reduced Parameter Space

In Hyper-MORL, the Pareto set is represented in a reduced parameter space (i.e., d in Figure 1). We further examine the effects of the dimensionality of the reduced parameter space on the hypervolume performance of Hyper-MORL. Figure 3 shows the results on seven problems. As we can see, the Pareto set of most problems are well approximated by Hyper-MORL in low-dimensional parameter spaces.

E Hypervolume Indicator

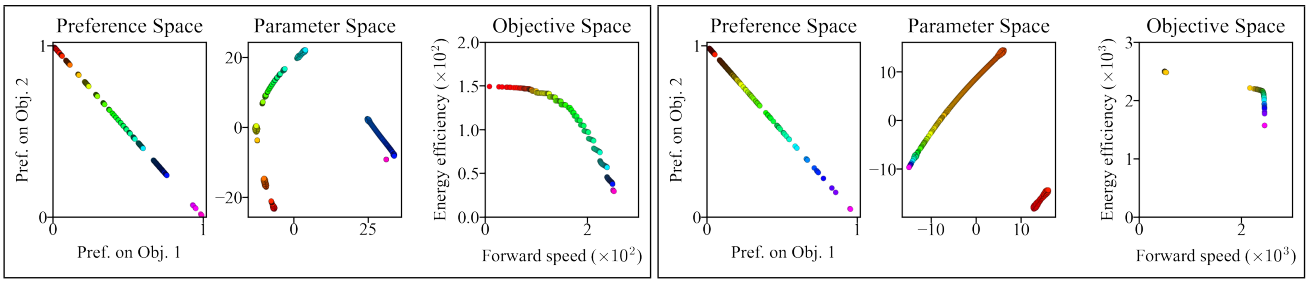
Mathematically, the hypervolume (HV) indicator is defined as follows.

Definition1. *Given a reference point $\boldsymbol{r} \in \mathbb{R}^m$, the hypervolume (i.e., HV) of a point set $S \subset \mathbb{R}^m$ is defined as*

$$HV(S, \boldsymbol{r}) = \mathcal{L}\left(\bigcup_{\boldsymbol{s} \in S} \{\boldsymbol{a} | \boldsymbol{s} \succeq \boldsymbol{a} \succeq \boldsymbol{r}\}\right), \quad (1)$$

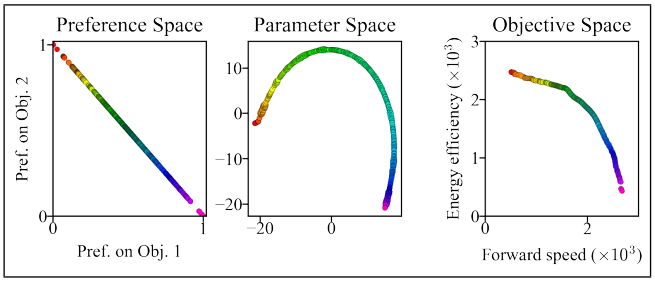
where $\mathcal{L}(\cdot)$ is the Lebesgue measure of a set, and $\boldsymbol{s} \succeq \boldsymbol{a}$ denotes $\boldsymbol{s}$ dominates $\boldsymbol{a}$ (i.e., $s_i \geq a_i \; \forall i \in \{1, 2, .., m\}$ and $s_j > a_j \; \exists j \in \{1, 2, ..., m\}$ in the maximization case).

Figure 4 illustrates the hypervolume of a solution set $\{\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}\}$ in a two-objective case.

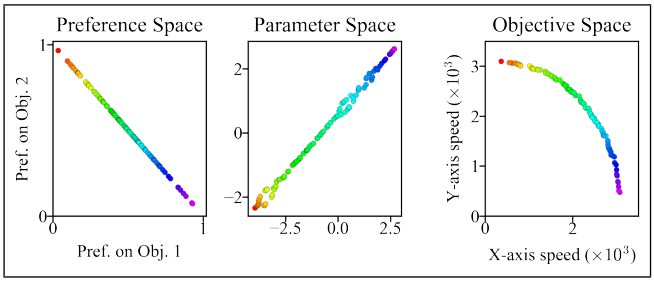


(a) MO-Swimmer-v2

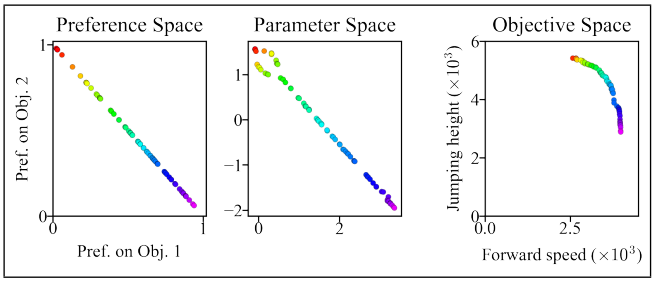
(b) MO-HalfCheetah-v2



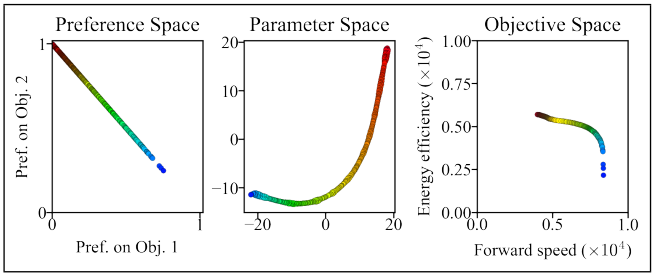
(c) MO-Walker2d-v2



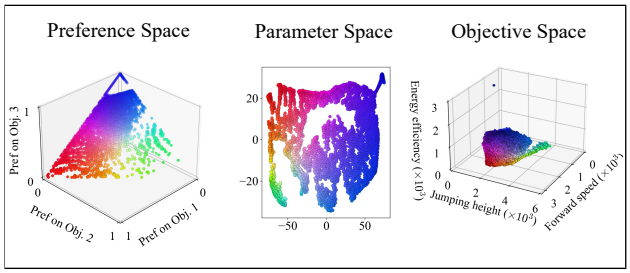
(d) MO-Ant-v2



(e) MO-Hopper-v2

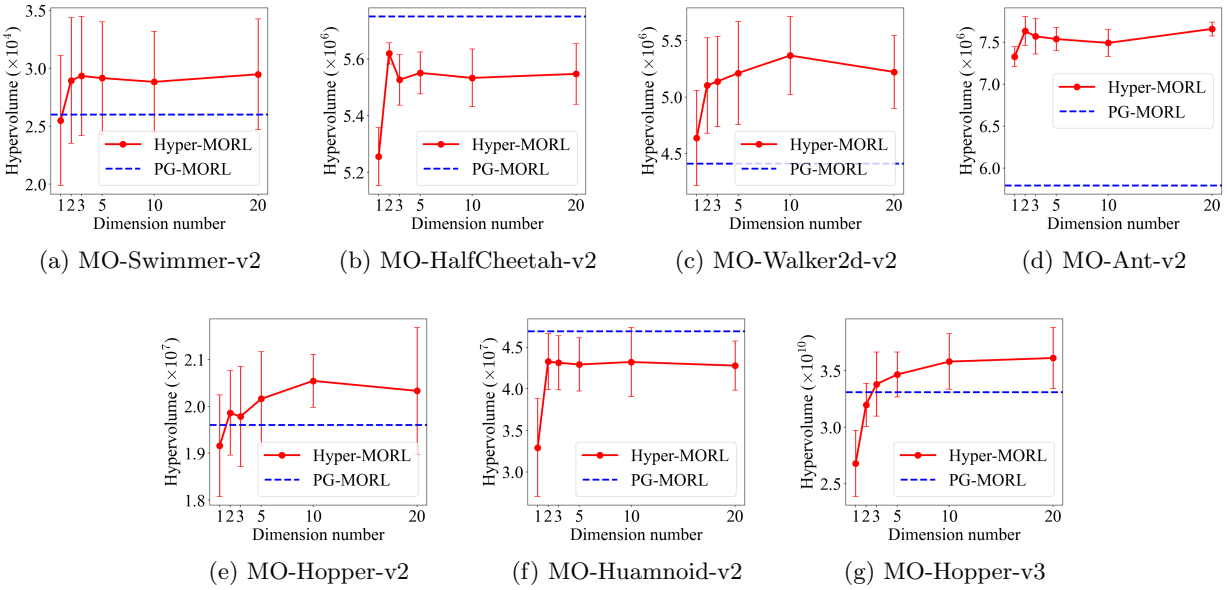


(f) MO-Humanoid-v2



(g) MO-Hopper-v3

Figure 2: Visualization of the nondominated policies obtained by Hyper-MORL in the parameter space (b) and objective space (c), and their corresponding input to the hypernet in the preference space (a) for each test problem. Each policy is plotted with the same color as its corresponding input preference. t-SNE is used for visualization in the high-dimensional parameter space.



(a) MO-Swimmer-v2 (b) MO-HalfCheetah-v2 (c) MO-Walker2d-v2 (d) MO-Ant-v2

(e) MO-Hopper-v2 (f) MO-Huamnoid-v2 (g) MO-Hopper-v3

Figure 3: 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.

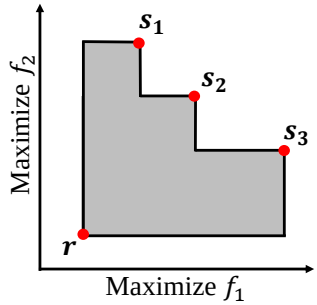


Figure 4: Illustration of the hypervolume indicator. The grey area is the hypervolume of the solution set $\{\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}\}$ with the reference point $\boldsymbol{r}$.

References

[Kostrikov, 2018] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[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.

[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.