PD-MORL: PREFERENCE-DRIVEN MULTI-OBJECTIVE REINFORCEMENT LEARNING ALGORITHM

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective reinforcement learning (MORL) approaches have emerged to tackle many real-world problems with multiple conflicting objectives by maximizing a joint objective function weighted by a preference vector. These approaches find *fixed customized policies corresponding to preference vectors specified during training*. However, the design constraints and objectives typically change dynamically in real-life scenarios. Furthermore, storing a policy for each potential preference is not scalable. Hence, obtaining a set of Pareto front solutions for the entire preference space in a given domain with a single training is critical. To this end, we propose a novel MORL algorithm that trains a *single universal network to cover the entire preference space scalable to continuous robotic tasks*. The proposed approach, Preference-Driven MORL (PD-MORL), utilizes the preferences as guidance to update the network parameters. It also employs a novel parallelization approach to increase sample efficiency. We show that PD-MORL achieves up to 25% larger hypervolume for challenging continuous control tasks compared to prior approaches using an order of magnitude fewer trainable parameters while achieving broad and dense Pareto front solutions.

1 INTRODUCTION

Reinforcement learning (RL) has emerged as a promising approach to solve various challenging problems including board/video games (Silver et al., 2016; Mnih et al., 2015; Shao et al., 2019), robotics (Nguyen & La, 2019), smart systems (Gupta et al., 2020; Yu et al., 2021), and chip design/placement (Zheng & Louri, 2019; Mirhoseini et al., 2020). The main objective in a standard RL setting is to obtain a policy that maximizes a single cumulative reward by interacting with the environment. However, many real-world problems involve multiple, possibly conflicting, objectives. For example, robotics tasks should maximize speed while minimizing energy consumption. In contrast to single-objective environments, performance is measured using multiple objectives. Consequently, there are multiple Pareto-optimal solutions as a function of the preference between objectives (Navon et al., 2020). Multi-objective reinforcement learning (MORL) approaches (Hayes et al., 2022) have emerged to tackle these problems by maximizing a vector of rewards depending on the preferences.

Existing approaches for multi-objective optimization generally transform the multidimensional objective space into a single dimension by statically assigning weights (preferences) to each objective (Liu et al., 2014). Then, they use standard RL algorithms to obtain a policy optimized for the given preferences. These approaches suffer when the objectives have widely varying magnitudes since setting the preference weights requires application domain expertise. More importantly, they can find only a single solution for a given set of goals and constraints. Thus, they need to repeat the training progress when the constraints or goals change. However, repetitive retraining is impractical since the constraints and design can change frequently depending on the application domain. Therefore, obtaining a set of Pareto front solutions that covers the entire preference space with a single training is critical (Xu et al., 2020; Yang et al., 2019; Abdolmaleki et al., 2020).

This paper presents a novel multi-objective reinforcement learning algorithm *using a single policy network that covers the entire preference space* in a given domain. At its core, it uses a multi-objective version of Q-Learning, where we approximate the Q-values with a neural network. This network takes the states and preferences as inputs during training. Making the preferences input parameters allows the trained model to produce the optimal policy for any user-specified preference at run-time. Since the user-specified preferences effectively drive the policy decisions, it is called preference-driven (PD) MORL. For each episode during training, we randomly sample a preference vector

($\boldsymbol{\omega} \in \Omega : \sum_{i=0}^{L} \omega_i = 1$) from a uniform distribution. Since the number of collected transitions by interacting with the environment for some preferences may be underrepresented, we utilize hindsight experience replay buffer (HER) (Andrychowicz et al., 2017). In our implementation of HER, every transition is also stored with randomly sampled preferences ($\boldsymbol{\omega}' \in \Omega : \sum_{i=0}^{L} \omega'_i = 1$) different than the original preference of the transition. As a key insight, we observe that the preference vectors have similar directional angles to the corresponding vectorized Q-values for a given state. Using the insight, we utilize the cosine similarity between the preference vector and vectorized Q-values in the Bellman's optimality operator to guide the training. However, not every Pareto front perfectly aligned with the preference vectors. To mitigate this adverse effect, we fit a multi-dimensional interpolator to project the original preference vectors ($\boldsymbol{\omega} \in \Omega$) to normalized solution space to align preferences with the multi-objective solutions. The projected preference vectors are used in our novel preference-driven optimality operator to obtain the target Q-values. Additionally, to increase the sample efficiency of the algorithm, we divide the preference space into sub-spaces and assign a child process to these sub-spaces. Each child process is responsible for its own preference sub-space to collect transitions. This parallelization provides efficient exploration during training, assuring that there is no bias towards any preference sub-space.

PD-MORL can be employed in any off-policy RL algorithm. We develop a multi-objective version of the double deep Q-network algorithm with hindsight experience replay buffer (MO-DDQN-HER) (Schaul et al., 2015) for problems with discrete action spaces and evaluate PD-MORL's performance on two commonly used MORL benchmarks: Deep Sea Treasure (Hayes et al., 2022) and Fruit Tree Navigation Task (Yang et al., 2019). We specifically choose these two benchmarks to make a fair comparison with the prior approach (Yang et al., 2019) that also aims to achieve a unified policy network and to show that PD-MORL scales to more than two objectives. Additionally, we develop a multi-objective version of the Twin Delayed Deep Deterministic policy gradient algorithm with hindsight experience replay buffer (MO-TD3-HER) (Fujimoto et al., 2018a) for problems with continuous action spaces. Using MO-TD3-HER, we evaluate PD-MORL on the multi-objective continuous control tasks such as MO-Walker2d-v2, MO-HalfCheetah-v2, MO-Ant-v2, MO-Swimmer-v2, MO-Hopper-v2 that are presented by Xu et al. (2020). With the combination of the use of the cosine similarity term, the HER, and the parallel exploration, PD-MORL achieves up to 78% larger hypervolume for simple benchmarks and 25% larger hypervolume for continuous control tasks compared to prior approaches while achieving broad and dense Pareto front solutions. We emphasize that it achieves these results with a single policy network.

2 Related Work

Existing MORL approaches can be classified as single-policy, multi-policy, and meta-policy approaches. The main difference among them is the number of policies learned during training. Single-policy approaches transform a multi-objective problem into a single-objective problem by combining the rewards into a single scalar reward using a scalarization function. Then, they use standard RL approaches to maximize the scalar reward (Roijers et al., 2013). Most of the previous studies find the optimal policy for *a given preference* between the objectives using scalarization (Van Moffaert et al., 2013). Additionally, recent work takes an orthogonal approach and encodes preferences as constraints instead of scalarization (Abdolmaleki et al., 2020). These approaches have two primary drawbacks: they require domain-specific knowledge, and preferences must be set beforehand.

Multi-policy approaches aim to obtain a set of policies that approximates the Pareto front of optimal solutions. The most widely used approach is to repeatedly perform a single-policy algorithm over various preferences (Roijers et al., 2014; Mossalam et al., 2016; Zuluaga et al., 2016). However, this approach suffers from a large number of objectives and dense Pareto solutions for complex control problems. In contrast, recent approaches aim to learn a set of policies with a single training. Abels et al. (2019) and Yang et al. (2019) use a multi-objective Q-learning approach that simultaneously learns a set of policies over multiple preferences. These studies use a single network that takes preferences as inputs and uses vectorized value function updates in contrast to standard scalarized value function updates. An orthogonal approach by Chen et al. (2019) is the meta-policy approach that frames MORL as a meta-learning problem using a task distribution given by a distribution over the preference space. The authors first train a meta-policy to approximate the Pareto front implicitly. Then, they obtain the Pareto optimal solution of a given preference by only fine-tuning the meta-policy with a few gradient updates. A more recent study proposes an efficient evolutionary learning algorithm

to update a population of policies simultaneously in each run to improve the approximation of the Pareto optimal solutions (Xu et al., 2020). The first generation of the policy population is obtained by optimizing n randomly initialized policies using n evenly distributed non-negative weights. It then uses a prediction model to guide the selection of new n policy-weight pairs, which are expected to improve the quality of the Pareto front. These selected pairs are optimized by a multi-objective policy gradient to produce new offspring policies used to update the policy population. All non-dominated policies for each generation of policies are stored as an external Pareto front archive during the training. Finally, it outputs this Pareto front archive as approximated Pareto front solutions. However, this approach obtains a policy network for each solution on the Pareto front. Moreover, there is no specific knowledge of correspondence between preference vectors and solutions in this Pareto front.

To the best of our knowledge, none of the prior approaches on MORL achieves dense Pareto front solutions for continuous control tasks using a single universal network which yields the optimal policy for any user-specific preference at run-time. PD-MORL utilizes the relation between the state-action values and the preferences and proposes a novel update rule including the cosine similarity between the Q-values and preference vectors. Additionally, PD-MORL increases its sample efficiency by using novel parallelization approach with HER and provides efficient exploration. The combination of the use of the cosine similarity term, the HER, and the parallel exploration achieves the state-of-the-art results for continuous controls tasks as well as problems with discrete action-space.

3 BACKGROUND

In standard RL, the agent learns optimal or near-optimal policies by interacting with the environment to maximize the reward. These rewards can be represented as state-action values, $Q^\pi(s,a)$. In contrast, MORL requires learning several tasks with different rewards simultaneously. Each objective has an associated reward signal, transforming the reward from a scalar to a vector, $\mathbf{r} = [r_1, r_2, ..., r_L]^T$, where L is the number of objectives. This vectorized reward can be represented as vectorized state-action values $MQ^\pi(s,a)$, as shown in Figure 1(a). A multi-objective problem can be formulated as a multi-objective Markov decision process (MOMDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \Omega, f_\omega \rangle$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}(s'|s,a)$, $R(s,a)$, and Ω represents state space, action space, transition distribution, reward vector, and preference space, respectively. The function $f_\omega(\mathbf{r})$ yields a scalarized reward using a preference of $\omega \in \Omega$. *Scalarization* is the most common approach to solve multi-objective optimization problem in RL domain (Roijers et al., 2013). It transforms the reward vector into a single scalar, $f_\omega(r) = \omega^T \mathbf{r}$. If ω is taken as a fixed vector, the MOMDP boils down to a standard MDP, which can be solved using standard RL techniques. However, if we consider all possible returns from a MOMDP and all possible preferences in Ω, a set of non-dominated policies called the Pareto front can be obtained, as shown in Figure 1(b). A policy π is Pareto optimal if there is no other policy π' that improves its expected return for an objective without degrading the expected return of any other objective. For complex problems, such as continuous control tasks, obtaining an optimal Pareto front using an RL setting is an NP-hard problem (Xu et al., 2020). Hence, the main goal of the MORL algorithms is to obtain an approximation of the Pareto front. The quality of the approximated Pareto front is typically measured by two metrics: (i) hypervolume and (ii) sparsity (Hayes et al., 2022).

Definition 1 (Hypervolume Indicator). For the same P be a Pareto front approximation in an L-dimensional objective space and contains N solutions. Let $\mathbf{r}_0 \in \mathbb{R}^L$ be the reference point. Then, the hypervolume indicator is defined as:

$$I_H(P) := \Lambda(H(P, \mathbf{r}_0)) \tag{1}$$

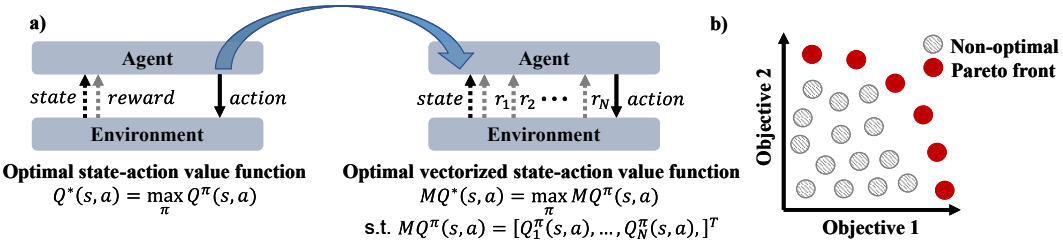

Figure 1: (a) RL to MORL transformation. (b) Pareto front example.

where $H(P, \mathbf{r}_0) = \{\mathbf{z} \in \mathbb{R}^L | \exists\, 1 \leq i \leq N : \mathbf{r}_0 \preceq \mathbf{z} \preceq P_i\}$ with P_i being the ith solution in P and $\preceq$ is the relation operator of multi-objective dominance. $\Lambda(\cdot)$ denotes the Lebesgue measure with $\Lambda(H(P, \mathbf{r}_0)) = \int_{\mathbb{R}^L} \mathbb{1}_{H(P,\mathbf{r}_0)}(\mathbf{z})d\mathbf{z}$ and $\mathbb{1}_{H(P,\mathbf{r}_0)}$ being the characteristic function of $H(P, \mathbf{r}_0)$.

Definition 2 (Sparsity). For the same P, sparsity is defined as:

$$Sp(P) := \frac{1}{N-1} \sum_{j=1}^{L} \sum_{i=1}^{N-1} (P_{i_j} - P_{i+1_j})^2 \quad (2)$$

where P_i is the ith solution in P and P_{i_j} is the sorted set for the jth objective.

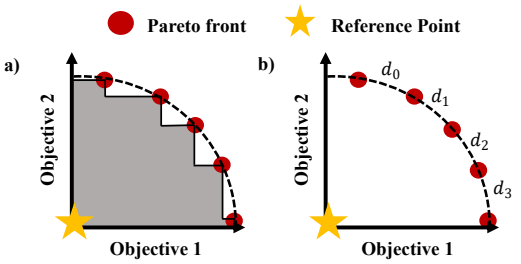

Figure 2: (a) Hypervolume Indicator. The hypervolume between a reference point and solutions in Pareto front is shown with the shaded area. (b) Sparsity. The sparsity is the average square distance ($d_i, i = \{0, \ldots, 3\}$) between consecutive solutions in the Pareto front.

Improvement in any objective manifests as an increase in the hypervolume, as illustrated in Figure 2. However, the hypervolume alone is insufficient to assess the quality of the Pareto front. For instance, hypervolume may increase due to an increase in only one of the objectives which indicates that the algorithm does not improve other objectives. The sparsity metric is introduced to assess whether the Pareto front solutions are dense or sparse. A larger hypervolume indicates that a more desired Pareto front approximation is achieved. Lower sparsity values imply that a dense set of solutions is achieved.

4 PD-MORL: Preference Driven MORL Algorithm

This section introduces the proposed preference-driven MORL algorithm. We first provide a theoretical analysis of PD-MORL based on the multi-objective version of Q-learning (Watkins & Dayan, 1992). The proofs for this theoretical analysis are available in Appendix A. We then introduce multi-objective double deep Q-networks with a preference-driven optimality operator for problems with discrete action space. Finally, we extend the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm (Fujimoto et al., 2018b) to a multi-objective version utilizing our proposed approach for problems with continuous action space.

4.1 Theoretical Analysis of PD-MORL

In standard Q-learning, the value space is defined as $\mathcal{Q} \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, containing all bounded functions $Q(s, a)$ which are the estimates of the total expected rewards when the agent is at state s, taking action a. We extend it to a multi-objective value space by defining the value space as $\mathcal{Q} \in \mathbb{R}^{L^{\mathcal{S} \times \mathcal{A}}}$, containing all bounded functions $\mathbf{Q}(s, a, \boldsymbol{\omega})$ which are the estimates of expected total rewards under preference $\boldsymbol{\omega} \in \mathbb{R}^L_{\geq 0} : \sum_{i=0}^{L} \omega_i = 1$. We then define a metric in this value space as:

$$d(\mathbf{Q}, \mathbf{Q}') := \sup_{s \in \mathcal{S}, a \in \mathcal{A}, \boldsymbol{\omega} \in \Omega} |\boldsymbol{\omega}^T (\mathbf{Q}(s, a, \boldsymbol{\omega}) - \mathbf{Q}'(s, a, \boldsymbol{\omega}))|. \quad (3)$$

This metric gives the distance between $\mathbf{Q}$ and $\mathbf{Q}'$ as the supremum norm of the scalarized distance between these two vectors where the identitity of indiscernables ($d(\mathbf{Q}, \mathbf{Q}') = 0 \Leftrightarrow \mathbf{Q} = \mathbf{Q}'$) does not hold and thus, makes metric d a pseudo-metric. Further details for the axioms of this pseudo-metric are given in Appendix A. In the following Theorem, we showed that the metric space equipped with this metric is complete since the limit point of the sequence of operators is required to be in this space.

Theorem 1 (Multi-objective Metric Space $(\mathcal{Q}, d)$ is Complete)**.** *The metric space $(\mathcal{Q}, d)$ is complete and every Cauchy sequence $\mathbf{Q}_n(s, a, \boldsymbol{\omega})$ is convergent in metric space $(\mathcal{Q}, d)$ $\forall\, s, a, \boldsymbol{\omega} \in \mathcal{S}, \mathcal{A}, \Omega$.*

Given a policy π and sampled transition τ, we can define multi-objective Bellman's evaluation operator $\mathcal{T}_\pi$ using the metric space $(\mathcal{Q}, d)$ as:

$$(\mathcal{T}_\pi \mathbf{Q})(s, a, \boldsymbol{\omega}) := \mathbf{r}(s, a) + \gamma \mathbb{E}_{\mathcal{T} \sim (\mathcal{P}, \pi)} \mathbf{Q}(s', a', \boldsymbol{\omega}) \quad (4)$$

where $(s', a', \boldsymbol{\omega})$ denotes the next state-action-preference pair and $\gamma \in (0, 1)$ is the discount factor.

We define a *preference-driven optimality operator* $\mathcal{T}$ by adding a cosine similarity term between preference vectors and state-action values to the multi-objective Bellman's optimality operator as:

$$(\mathcal{T}\mathbf{Q})(s,a,\boldsymbol{\omega}) := \mathbf{r}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} \mathbf{Q}(s', \sup_{a' \in \mathcal{A}} (S_c(\boldsymbol{\omega}, \mathbf{Q}(s',a',\boldsymbol{\omega})) \cdot (\boldsymbol{\omega}^T \mathbf{Q}(s',a',\boldsymbol{\omega}))), \boldsymbol{\omega}) \tag{5}$$

where $S_c(\boldsymbol{\omega}, \mathbf{Q}(s',a',\boldsymbol{\omega}))$ denotes the cosine similarity between preference vector and Q-value. This term enables our optimality operator to choose actions that align the preferences with the Q-values and maximize the target value, as elaborated in Section 4.2 under the preference alignment subtitle.

Theorem 2 (Multi-objective Bellman's Evaluation Operator is Contraction)**.** *Let $(\mathcal{Q}, d)$ be a complete metric space (as in Theorem 1). Let $\mathbf{Q}$ and $\mathbf{Q}'$ be any two multi-objective Q-value functions in this space. The multi-objective Bellman's evaluation operator is a contraction and $d(\mathcal{T}_\pi \mathbf{Q}, \mathcal{T}_\pi \mathbf{Q}') \leq \gamma d(\mathbf{Q}, \mathbf{Q}')$ holds for the Lipschitz constant γ (the discount factor).*

Theorem 3 (Preference-driven Multi-objective Bellman's Optimality Operator is Contraction)**.** *Let $(\mathcal{Q}, d)$ is a complete metric space. Let $\mathbf{Q}$ and $\mathbf{Q}'$ be any two multi-objective Q-value functions in this space. The preference-driven multi-objective Bellman's optimality operator is a contraction and $d(\mathcal{T}\mathbf{Q}, \mathcal{T}\mathbf{Q}') \leq \gamma d(\mathbf{Q}, \mathbf{Q}')$ holds for the Lipschitz constant γ (the discount factor).*

Theorem 2 and Theorem 3 state that multi-objective evaluation and optimality operators are contraction. They ensure that we can apply our optimality operator in Equation 5 iteratively to obtain the optimal multi-objective value function given by Theorem 4 and Theorem 5 below, respectively.

Theorem 4 (Preference-driven Multi-objective Optimality Operator Converges to a Fixed-Point)**.** *Let $(\mathcal{Q}, d)$ is a complete metric space which is proved above and let $\mathcal{T} : \mathcal{Q} \to \mathcal{Q}$ be a contraction on $\mathcal{Q}$ with modulus γ. Then, $\mathcal{T}$ has a unique $\mathbf{Q}^* \in \mathcal{Q}$ such that $\mathcal{T}(\mathbf{Q}) = \mathbf{Q}^*$.*

Theorem 5 (Optimal Fixed Point of Optimality Operator)**.** *Let $\mathbf{Q}^* \in \mathcal{Q}$ be the optimal multi-objective value function, such that it takes multi-objective Q-value corresponding to the supremum of expected discounted rewards under a policy π then $\mathbf{Q}^* = \mathcal{T}(\mathbf{Q}^*)$.*

These two theorems state that applying our multi-objective optimality operator T iteratively on any multi-objective Q-value function $\mathbf{Q}$ converges to the optimal $\mathbf{Q}^*$ under metric d and the Bellman equation holds at the same fixed-point. Hence, we use this optimality operator to obtain target values when optimizing the loss function in Algorithm 1.

4.2 Preference Driven MO-DDQN-HER and MO-TD3-HER

The main objective of this work is to obtain a single policy network to approximate the Pareto front that covers the entire preference space. For this purpose, we extend double deep Q-network (DDQN) (Van Hasselt et al., 2016) to a multi-objective version (MO-DDQN) with the preference-driven optimality operator to obtain a single parameterized function represents the $\mathcal{Q} \in \mathbb{R}^{L^{\mathcal{S} \times \mathcal{A}}}$ with parameters θ. This network takes $s, \boldsymbol{\omega}$ as input and outputs $|\mathcal{A}| \times L$ Q-values, as described in Algorithm 1. To pull $\mathbf{Q}$ towards $T(\mathbf{Q})$, MO-DDQN minimizes the following loss function at each step k:

$$L_k(\theta) = \mathbb{E}_{(s,a,\mathbf{r},s',\boldsymbol{\omega}) \sim \mathcal{D}} \Big[\Big(\mathbf{y} - \mathbf{Q}(s,a,\boldsymbol{\omega};\theta) \Big)^2 \Big] \tag{6}$$

where $\mathcal{D}$ is the experience replay buffer that stores transitions $(s, a, \mathbf{r}, s', \boldsymbol{\omega})$ for every time step, $\mathbf{y} = \mathbf{r} + \gamma \mathbf{Q}(s', \sup_{a' \in A}(S_c(\boldsymbol{\omega}, \mathbf{Q}(s',a',\boldsymbol{\omega})) \cdot (\boldsymbol{\omega}^T \mathbf{Q}(s',a',\boldsymbol{\omega}))); \theta')$ denotes the preference-driven target value which is obtained using target Q-network's parameters θ'.

For each episode during training, we randomly sample a preference vector ($\boldsymbol{\omega} \in \Omega : \sum_{i=0}^{L} \omega_i = 1$) from a uniform distribution. For example, assume this randomly sampled preference vector ($\boldsymbol{\omega}$) favors energy efficiency rather than speed in multi-objective continuous control tasks. Increasing energy efficiency (i.e., lower speed) will increase the number of transitions before reaching a terminal condition since it decreases the risk of reaching a terminal state (e.g., fall). Hence, transitions that favor speed instead of energy efficiency may be underrepresented in the experience replay buffer in this example. Therefore, we employ hindsight experience replay buffer (HER) (Andrychowicz et al., 2017), where every transition is also stored with N_ω randomly sampled preferences ($\boldsymbol{\omega}' \in \Omega : \sum_{i=0}^{L} \omega'_i = 1$) different than the original preference of the transition. *To increase the sample efficiency of our algorithm*, we also divide the preference space into C_p sub-spaces ($\tilde{\Omega}$), where C_p denotes the number of child processes.

Each child process is responsible for its own preference sub-space. The agent, hence the network, is shared among child processes and the main process. These C_p child processes run in parallel to collect transitions, within its preference sub-space $\tilde{\Omega}$. These transitions are stored inside the HER in the main process. Network architectures and implementation details are available in the supplementary material. This parallelization provides efficient exploration during training.

Algorithm 1: Preference Driven MO-DDQN-HER

Input: Minibatch size N_m, Number of time steps N, Discount factor γ, Target network update coefficient τ, Multi-dimensional interpolator $I(\boldsymbol{\omega})$.
Initialize: Replay buffer $\mathcal{D}$, Current $\mathbf{Q}_\theta$ and target network $\mathbf{Q}_{\theta'} \leftarrow \mathbf{Q}_\theta$ with parameters θ and θ'.
for n = 0: N **do**
  `// Child Process`
  Initialize $t = 0$ and $done = False$.
  Reset the environment to randomly initialized state s_0.
  Sample a preference vector $\boldsymbol{\omega}$ from the subspace $\tilde{\Omega}$.
  while $done = False$ **do**
    Observe state s_t and select an action a_t ϵ-greedily:
    $$a_t = \begin{cases} a \in A & w.p. \quad \epsilon \\ \max_{a \in \mathcal{A}} \boldsymbol{\omega}\mathbf{Q}(s_t, a, \boldsymbol{\omega}; \theta), & w.p. \quad 1-\epsilon \end{cases}$$
    Observe $\mathbf{r}$, s', and $done$.
    Transfer $(s_t, a_t, \mathbf{r}_t, s', \boldsymbol{\omega}, done)$ to main process.
  `// Main Process`
  Sample N_ω preferences $\boldsymbol{\omega}'$
  for j = 1: N_ω **do**
    Store transition $(s_t, a_t, \mathbf{r}_t, s', \boldsymbol{\omega}'_j, done)$ in $\mathcal{D}$
  Sample N_m transitions from $\mathcal{D}$.
  $\boldsymbol{\omega}_p \leftarrow I(\boldsymbol{\omega})$
  $\mathbf{y} \leftarrow \mathbf{r} + \gamma \mathbf{Q}(s', \sup_{a' \in A}(S_c(\boldsymbol{\omega}_p, \mathbf{Q}(s', a', \boldsymbol{\omega})) \cdot (\boldsymbol{\omega}^T \mathbf{Q}(s', a', \boldsymbol{\omega})); \theta')$
  $L_k(\theta) = \mathbb{E}_{(s,a,\mathbf{r},s',\boldsymbol{\omega}) \sim \mathcal{D}}\Big[\Big(\mathbf{y} - \mathbf{Q}(s, a, \boldsymbol{\omega}; \theta)\Big)^2\Big]$
  Update θ_k by applying SGD to $L_k(\theta)$.
  Update target network parameters $\theta'_k \leftarrow \tau\theta_k + (1-\tau)\theta'_k$

Preference alignment: A solution in the Pareto front may not align perfectly with its preference vector. To mitigate this adverse effect, we fit a multi-dimensional interpolator to project the original preference vectors ($\boldsymbol{\omega} \in \Omega$) to normalized solution space to align preferences with the multi-objective solutions. We identify the key preferences and obtain solutions for each of these preferences without HER. We use these solutions first to obtain a normalized solution space, as shown in Figure 3(b). Here, normalization is the process of obtaining unit vectors for these solutions. Then, we use the normalized solution space and key preference point to fit a multi-dimensional interpolator $I(\boldsymbol{\omega})$ to project the original preference vectors ($\boldsymbol{\omega} \in \Omega$) to the normalized solution space. As a result, we obtain projected preference vectors ($\boldsymbol{\omega}_p$) that are incorporated in the cosine similarity term of the preference-driven optimality operator, as illustrated in Figure 3 for the MO-Swimmer-v2 problem. The interpolator is also updated during training as PD-MORL may find new non-dominated solutions for identified key preferences.

Extension to continuous action space: Finally, we extend the TD3 algorithm to a multi-objective version using PD-MORL for problems with continuous action space. In this case, the target values are no longer computed using the optimality operator. Instead, the actions for the target Q-value are determined by the target actor network. The actor network is updated to obtain a policy that maximizes the Q-values generated by the critic network. This update rule eventually boils down to obtaining a policy that maximizes the expected discounted rewards since, by definition, this expectation is the Q-value itself. To incorporate the relation between preference vectors and Q-values for the multi-objective version of TD3, we include a directional angle term to both actor's and critic's loss function. This term computes the directional angle between the preference vector and the

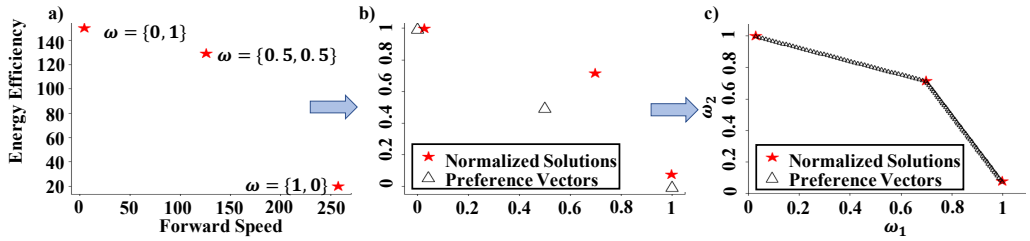

Figure 3: Overview of the interpolation scheme for the MO-Swimmer-v2 problem. (a) Obtained solutions for key reference points. (b) Normalized solution space and corresponding preference vectors. (c) Interpolated preference space.

corresponding Q-value and provides an alignment between preferences and the Q-values. Algorithm, network architectures, and implementation details are available in the supplementary material.

5 EXPERIMENTS

This section extensively evaluates the proposed PD-MORL technique using commonly used MORL benchmarks with discrete state-action spaces (Section 5.1) and complex MORL environments with continuous state-action spaces (Section 5.2). Our results are compared to state-of-the-art techniques in terms of hypervolume, sparsity, and the Pareto front plots.

5.1 MORL BENCHMARKS WITH DISCRETE STATE-ACTION SPACES

This section illustrates the efficacy of PD-MORL in *obtaining a single universal network that covers the entire preference space* in discrete state-action environments. We use the following commonly used benchmarks to (i) make a fair comparison with prior work (Yang et al., 2019), and (ii) show that PD-MORL scales to more than two objectives:

- **Deep Sea Treasure (DST)** has discrete state ($\mathcal{S} \subseteq \mathbb{N}^2$) and action spaces ($\mathcal{A} \subseteq \mathbb{N}^4$). It has two competing objectives: time penalty and treasure value.
- **Fruit Tree Navigation (FTN)** has discrete state ($\mathcal{S} \subseteq \mathbb{N}^2$) and action spaces ($\mathcal{A} \subseteq \mathbb{N}^2$) with six objectives: different nutrition facts of the fruits on the tree: {Protein, Carbs, Fats, Vitamins, Minerals, Water}.

Many prior studies considered DST (Hayes et al., 2022; Van Moffaert et al., 2013; Mossalam et al., 2016) by training a dedicated policy for each specific preference vector. Unlike them, PD-MORL learns a single universal policy that can work with any preference vector specified at run-time. For example, Figure 4 illustrates our results for three preference vectors capture corner cases. Although our policy is *not specific* to any preference, it achieves optimal results in all corner cases. For example, it finds the best trade-off within the shortest time when we only value the time penalty ($w = \{0, 1\}$). As the importance of the treasure value increases ($w =$\{0.5,0.5\}, $w = \{1, 0\}$), it spends more time to find a deeper treasure. Figure 5 provides further insights into the progression of the policies found by PD-MORL. Let the first objective is the treasure value and the second objective is the time penalty in this figure. At the beginning of the training, each child process starts collecting transitions within their own preference sub-space $\tilde{\Omega}$. Since the agent acts ϵ-greedily at early phases and cannot reach the higher treasure values due to terminal condition (time limit), we observe that most of the solutions are stuck in the first-subspace ($\tilde{\Omega}_1$) The transitions collected from child processes are stored in the hindsight experience replay buffer with different preferences other than their original preferences. This buffer enables extra exploration for the agent and leads the agent to different sub-spaces, as illustrated in the middle figure. As the training progresses, HER and our novel optimality operator with the angle term guide the search to expand the Pareto front and cover the entire preference space with a single network.

Among existing algorithms that learns a unified policy (Yang et al., 2019; Abels et al., 2019), the Envelope algorithm (Yang et al., 2019) is the superior and a more recent approach. Hence, we compare and evaluate PD-MORL and the Envelope algorithm on DST and FTN by obtaining a set of preference vectors that covers the entire preference space. PD-MORL achieves both 6.1% larger hypervolume and 56% lower sparsity than the Envelope algorithm for DST problem, as shown in Table 1. Similarly, PD-MORL generates up to 78% larger hypervolume than the Envelope algorithm for FTN.

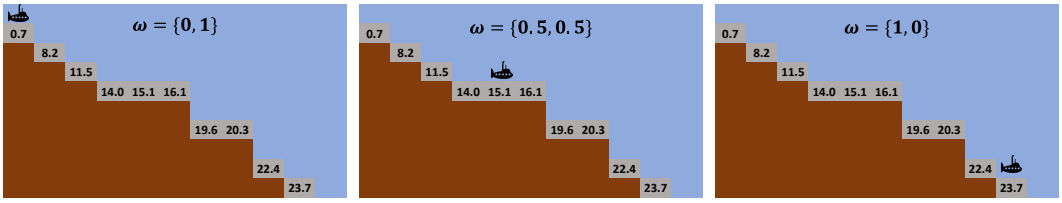

Figure 4: Optimal policies obtained by the trained agent for three corner preferences.

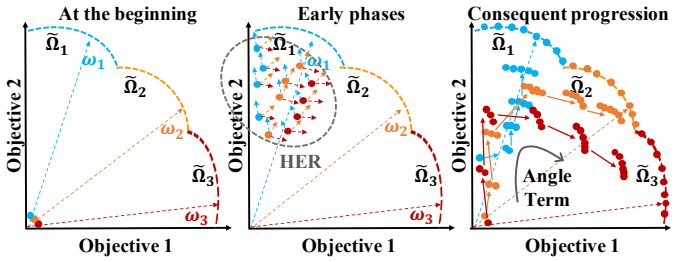

Figure 5: Policy progression during training. HER, the angle term, and the child processes enable PD-MORL align the Pareto front with the preference vector as the training progresses.

Since FTN is a binary tree, the distance between Pareto solutions is the same, and hence, the sparsity metric is the same between the two approaches. Our experiments show that the improvement of PD-MORL compared to the Envelope algorithm increases with the problem complexity (i.e., increasing tree depth), as shown in Table 1. PD-MORL provides efficient and robust exploration using HER and our novel optimality operator together. Further comparisons against the envelope algorithm (Yang et al., 2019), including the Coverage Ratio F1 (CRF1) metric defined by the authors, are provided in the supplementary material.

Table 1: Comparison of our approach with prior work (Yang et al., 2019) using two simple MORL benchmarks in terms of hypervolume and sparsity metrics.

	Deep Sea Treasure		Fruit Tree Navigation (d=6)		Fruit Tree Navigation (d=7)	
	Hypervolume	**Sparsity**	**Hypervolume**	**Sparsity**	**Hypervolume**	**Sparsity**
Envelope (Yang et al., 2019)	227.89	2.62	8427.51	N/A	6395.27	N/A
PD-MORL	241.73	1.14	9299.15	N/A	11419.58	N/A

5.2 MORL Benchmarks with Continuous Control Tasks

This section evaluates PD-MORL on popular multi-objective continuous control tasks based on MuJoCo physics engine (Xu et al., 2020; Todorov et al., 2012). The environment details of these benchmarks are given in Table 2. The Envelope algorithm (Yang et al., 2019) is not evaluated using these benchmarks. This and the fact that PD-MORL outperforms the Envelope algorithm for simpler discrete problems, we compare the performance of PD-MORL against two state-of-the-art approaches (Xu et al., 2020; Chen et al., 2019) that use these benchmarks. We emphasize that *these approaches must learn a different policy network for every solution in the Pareto front*. In contrast, PD-MORL learns *a single universal network* that covers the entire preference space. Training details are reported in the supplementary material.

Table 2: Environment details of continuous control benchmarks. Number of objectives for these benchmarks is 2.

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

We first compare PD-MORL to prior work using hypervolume and sparsity metrics. Since META (Chen et al., 2019) and PG-MORL (Xu et al., 2020) report the average of six runs, we also ran each benchmark six times with PD-MORL. The average metrics are reported in Table 3, while the standard deviations and results of individual runs are given in the supplementary material.

The desired Pareto front approximation should have high hypervolume and low sparsity metrics. PD-MORL outperforms the current state-of-the-art techniques both in terms of hypervolume and sparsity on every benchmark except MO-Hopper-v2, as summarized in Table 3. We emphasize that our *PD-MORL technique trains only a single network, while the other methods use customized policies network for each Pareto point.* For example, we achieve 12% higher hypervolume and 25% better sparsity than the most competitive prior work PG-MORL (Xu et al., 2020) on the Walker environment. They reported 263 Pareto front solutions for this environment. This corresponds to, in total, 2.8×10^6 trainable parameters since they have different policy network for each solution. In contrast, PD-MORL achieves better or comparable result using only 3.4×10^5 trainable parameters. Similarly, we outperform it by as much as 25% in hypervolume (for the MO-Swimmer-v2 environment). We also plot Pareto front plots for all algorithms in Figure 6(a)-(e). Figure 6 shows that the proposed PD-MORL technique achieves a broader and denser Pareto front than the state-of-the-art approaches despite using a single unified network. This feature enables our algorithm to yield an optimal policy for any user-specified preference and adapt to dynamic changes in an application domain.

Table 3: Performance comparison of the proposed approach and state-of-the-art algorithms on the continuous control benchmarks in terms of hypervolume and sparsity metrics. Reference point for hypervolume calculation is set to (0,0) point. HV*: Hypervolume

	MO-Walker2d-v2		MO-HalfCheetah-v2		MO-Ant-v2		MO-Swimmer-v2		MO-Hopper-v2	
	HV*	Sparsity	HV*	Sparsity	HV*	Sparsity	HV*	Sparsity	HV*	Sparsity
PG-MORL (Xu et al., 2020)	4.82×10^6	0.04×10^4	5.77×10^6	$\mathbf{0.44 \times 10^3}$	6.35×10^6	$\mathbf{0.37 \times 10^4}$	2.57×10^4	9.9	$\mathbf{2.02 \times 10^7}$	0.5×10^4
META (Chen et al., 2019)	2.10×10^6	2.10×10^4	5.18×10^6	2.13×10^3	2.40×10^4	1.56×10^4	1.23×10^4	24.4	1.25×10^7	4.84×10^4
PD-MORL (Ours)	$\mathbf{5.41 \times 10^6}$	$\mathbf{0.03 \times 10^4}$	$\mathbf{5.89 \times 10^6}$	0.49×10^3	$\mathbf{7.48 \times 10^6}$	0.78×10^4	$\mathbf{3.21 \times 10^4}$	**5.7**	1.88×10^7	$\mathbf{0.3 \times 10^4}$

Finally, Figure 6(f) plots the progression of the hypervolume of the Pareto front for the environments with similar scales. The plots show that PD-MORL effectively pushes the hypervolume by discovering new Pareto solutions with the help of efficient and robust exploration. We ensure this by using HER, our novel directional angle term, and dividing the preference space into sub-spaces to collect transitions in parallel. The progression of the Pareto front, sparsity, and hypervolume of all benchmarks are provided in the supplementary material.

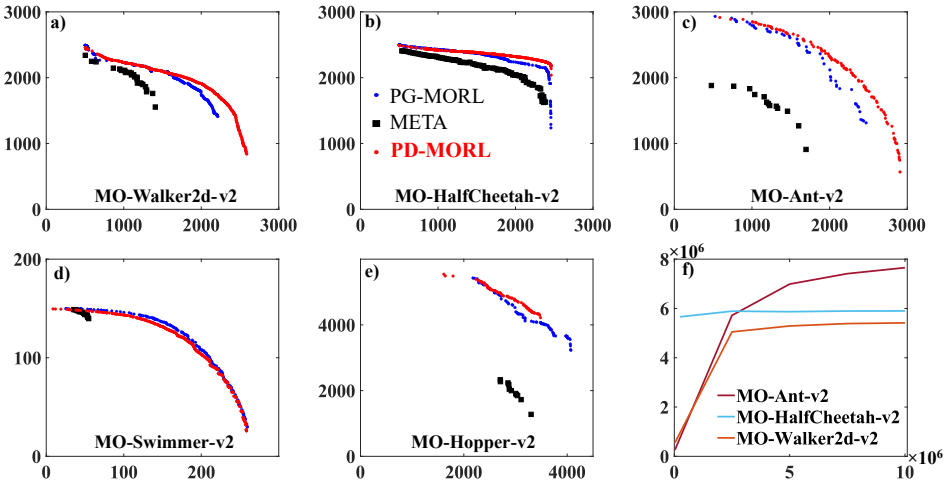

Figure 6: Pareto front comparison for (a) MO-Walker-v2, (b) MO-HalfCheetah-v2, (c) MO-Ant-v2, (d) MO-Swimmer-v2, (e) MO-Hopper-v2. Results for META and PG-MORL are obtained from (Xu et al., 2020) and code base of PG-MORL. (f) Progression of hypervolume of the Pareto front (x-axis shows the time steps). Results for MO-Walker-v2, MO-HalfCheetah-v2, and MO-Ant-v2 are given since they have similar scales.

6 Conclusions and Limitations

Many real-life applications have multiple conflicting objectives that can change dynamically. Since standard RL techniques with scalar rewards do not extend to multi-objective problems, MORL techniques have been proposed in the literature. These techniques use a preference vector to determine the importance of each objective. However, existing techniques either learn a different policy for each preference or fail to extend to complex continuous state-action benchmarks. This paper presented a novel MORL technique that addresses both of these limitations. It trains a universal network to cover the entire preference space using the preferences as guidance to update the network, hence the name preference-driven MORL (PD-MORL). We evaluated PD-MORL using both discrete and continuous benchmarks and demonstrated that PD-MORL outperforms the state-of-the-art approaches.

PD-MORL aims to obtain a generalized network that covers the entire preference space. It updates its network parameters by sampling transitions from HER. Since these transitions can represent anywhere in the preference space, the updates should be conservative. Therefore, it may fail to obtain a dense Pareto front on problems with massive action, state, and preference spaces which is the primary concern of any MORL algorithm.

REFERENCES

Abbas Abdolmaleki, Sandy Huang, Leonard Hasenclever, Michael Neunert, Francis Song, Martina Zambelli, Murilo Martins, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. A distributional view on multi-objective policy optimization. In *International Conference on Machine Learning*, pp. 11–22. PMLR, 2020.

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*, pp. 11–20. PMLR, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

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

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018a.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018b.

Saurabh Gupta, Siddhant Bhambri, Karan Dhingra, Arun Balaji Buduru, and Ponnurangam Kumaraguru. Multi-objective reinforcement learning based approach for user-centric power optimization in smart home environments. In *2020 IEEE International Conference on Smart Data Services (SMDS)*, pp. 89–96. IEEE, 2020.

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): 1–59, 2022.

Chunming Liu, Xin Xu, and Dewen Hu. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398, 2014.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*, 2020.

Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 590–595. IEEE, 2019.

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.

Diederik M Roijers, Shimon Whiteson, Frans A Oliehoek, et al. Linear support for multi-objective coordination graphs. In *AAMAS'14: Proceedings of the 2014 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1297–1304, 2014.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 191–199. IEEE, 2013.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

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*, pp. 10607–10616. PMLR, 2020.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in Neural Information Processing Systems*, 32:14636–14647, 2019.

Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. A review of deep reinforcement learning for smart building energy management. *IEEE Internet of Things Journal*, 2021.

Hao Zheng and Ahmed Louri. An energy-efficient network-on-chip design using reinforcement learning. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

Marcela Zuluaga, Andreas Krause, and Markus Püschel. ε-pal: an active learning approach to the multi-objective optimization problem. *The Journal of Machine Learning Research*, 17(1):3619–3650, 2016.