WAE-PCN: Wasserstein-autoencoded Pareto Conditioned Networks

Florent Delgrange*
AI Lab, Vrije Universiteit Brussel
University of Antwerp
Belgium
florent.delgrange@ai.vub.ac.be

Mathieu Reymond*
AI Lab, Vrije Universiteit Brussel
Belgium
mathieu.reymond@vub.be

Ann Nowé
AI Lab, Vrije Universiteit Brussel
Belgium

Guillermo A. Pérez
University of Antwerp - Flanders Make
Belgium

ABSTRACT

In real-world problems, decision makers often have to balance multiple objectives, which can result in trade-offs. One approach to finding a compromise is to use a multi-objective approach, which builds a set of all optimal trade-offs called a Pareto front. Learning the Pareto front requires exploring many different parts of the state-space, which can be time-consuming and increase the chances of encountering undesired or dangerous parts of the state-space. In this preliminary work, we propose a method that combines two frameworks, Pareto Conditioned Networks (PCN) and Wasserstein auto-encoded MDPs (WAE-MDPs), to efficiently learn all possible trade-offs while providing formal guarantees on the learned policies. The proposed method learns the Pareto-optimal policies while providing safety and performance guarantees, especially towards unexpected events, in the multi-objective setting.

1 INTRODUCTION

Decision makers acting in real-world problems often have to take into account multiple objectives. When maximizing one objective comes at the cost of another, the objectives are in conflict, and the decision maker must find a compromise between them. For example, maximizing the electricity output of a hydroelectric power plant comes at the expense of increased flooding risks downstream as well as irrigation deficits [1].

We can directly learn the best compromises by using an explicitly multi-objective approach. Assuming that improving an objective is always preferable, we can build a set of all optimal trade-offs called a *Pareto front*. The decision maker can then use the Pareto front to review all available policies, and use this knowledge to select their preferred one [15]. However, learning the Pareto front means exploring many different parts of the state-space, which requires to accumulate more experience than when optimizing a single objective. This also increases the chances of encountering undesired or dangerous parts of the state-space.

In this preliminary work, we propose to learn all possible trade-offs efficiently, while at the same time providing formal guarantees (e.g., performance, safety) on the learned policies. To do so, we build upon two successful frameworks for respectively learning the policies of such a Pareto front, and providing such guarantees: *Pareto Conditioned Networks* (PCN) [14] and *Wasserstein auto-encoded MDPs* (WAE-MDPs) [2]. The former is a goal-conditioned *reinforcement learning* (RL) algorithm which learns the set of *Pareto-optimal* policies. While PCN has proven to be effective in various environments, it struggles to cope with unexpected events. WAE-MDPs allows to *distill single-objective* RL policies, learn a latent space model, and enable their formal verification via model checking tools as well as *bisimulation* guarantees between the original and latent environments (in a nutshell, an agent executing a policy in the two models is guaranteed to behave the same way). However, those guarantees have only be proven to hold in single-objective settings. Moreover, they assume a fixed policy from which behavior is distilled. We propose to combine the strengths of both approaches to provide an algorithm that not only naively combines the two methods but also mutually assist each other to overcome their respective challenges. This results in a method that learns the Pareto front, while providing safety and performance guarantees, e.g., towards unexpected events and the multi-objective setting.

2 BACKGROUND

In the following, we write $\Delta(\mathcal{X})$ for the set of measures over (complete, separable metric space) $\mathcal{X}$ and $[N]$ for $\{n \in \mathbb{N} \mid 1 \leq n \leq N\}$.

2.1 Multi-Objective Decision Making

Markov Decision Processes. In scenarios where the agent must achieve multiple, potentially conflicting goals while interacting with an unpredictable, unknown environment, *multi-objective Markov decision processes* (MOMDPs) provide a valuable tool for making decisions under uncertainty. An MOMDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, s_I, N, \gamma \rangle$ where: $\mathcal{S}$ is a set of *states*; $\mathcal{A}$, a set of *actions*; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, a *probability transition function* that maps the current state and action to a *distribution* over the next states; $\boldsymbol{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$, a *multi-dimensional reward function*; $s_I \in \mathcal{S}$, the *initial state*; $N \in \mathbb{N}$, the number of agent's objectives; and $\gamma \in [0, 1)$, a *discount factor*. An agent interacting in $\mathcal{M}$ produces *trajectories*, i.e., sequences of states, actions, and rewards $\tau = \langle s_{0:T}, a_{0:T-1}, \boldsymbol{r}_{0:T-1} \rangle$ where $s_0 = s_I$, $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$, and $\boldsymbol{r}_{t+1} = \boldsymbol{\mathcal{R}}(s, a)$ for $t < T$. The set of infinite trajectories of $\mathcal{M}$ is $Traj_{\mathcal{M}}$.

Policies. A *(stationary) policy* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ prescribes which action to choose at each step of the interaction. The set of stationary policies of $\mathcal{M}$ is Π. The MOMDP $\mathcal{M}$ and π induce a unique

*Both authors contributed equally to this research, order decided on a coin flip.

Proc.of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 9-10, 2023, Online, https://ala2023.github.io/. 2023.

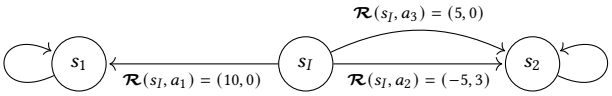


Figure 1: Simple MOMDP with two states, three actions. All transitions (depicted by edges) have probability one.

probability measure $\mathbb{P}_\pi^{\mathcal{M}}$ on the Borel σ-algebra over (measurable) infinite trajectories [13]. We write $\mathbb{P}_\pi(\cdot \mid s_I = s)$ for the probability measure over events of the MDP $\mathcal{M}$ with $s \in \mathcal{S}$ as initial state. The unique *stationary distribution* $\xi_\pi \in \Delta(\mathcal{S})$ is the distribution over states assigning to each state the probability of encountering that state when the agent follows π, at the limit: $\xi_\pi(s) = \lim_{t\to\infty} \mathbb{P}_\pi(\{s_{0:\infty}, a_{0:\infty}, \boldsymbol{r}_{0:\infty} \mid s_t = s\})$. Such a unique stationary distribution is guaranteed to exist in *episodic RL processes* [10].

Value functions. We formalize the agent's behavior through *value functions*: given a policy π, the *value* of a state s is the expected value of a random variable obtained by running π from s. We consider the following value functions.

(i) *Discounted return*: we write $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}_t \mid s_I = s\right]$ for the expected discounted rewards accumulated along trajectories and $\mathbf{V}^\pi$ for the vector resulting of $V^\pi(s_I)$. When the context is clear, we may omit the superscript;

(ii) *Reachability*: we define the *reachability* to a *goal state* as the discounted probability of visiting this state *for the first time*. This is done by defining a binary reward signal capturing this event and computing the return of this reward signal. Let $\mathrm{G} \subseteq \mathcal{S}$ be the set of goal states, the *reachability* event is $\Diamond\mathrm{G} = \{\, s_{0:\infty}, a_{0:\infty}, \boldsymbol{r}_{0:\infty} \mid \exists i \in \mathbb{N}, s_i \in \mathrm{G} \,\} \subseteq \mathit{Traj}_{\mathcal{M}}$. We write $V^\pi_{\Diamond\mathrm{G}}(s) = \mathbb{E}_\pi\left[\gamma^{t^\star} \mathbb{1}_{\tau \in \Diamond\mathrm{G}} \mid s_I = s\right]$ for the *discounted reachability* to G, where $t^\star$ is the length of the shortest trajectory prefix hitting G. Notice that $V^\pi_{\Diamond\mathrm{G}}(s) = \mathbb{P}_\pi(\Diamond\mathrm{G} \mid s_I = s)$ for $\gamma = 1$.

Pareto-optimality. When $N = 1$, the typical goal of an RL agent is to learn a policy $\pi^\star$ that maximizes the single-objective value $V^{\pi^\star}$ by interacting with $\mathcal{M}$. In contrast, when $N > 1$, there is no *optimal* policy as such, the multidimensional rewards may lead to trade-offs between the different objectives formalized through the N dimensions of the value function.

For instance, take the MDP of Fig. 1 and consider two policies, π_1 and π_2, so that $\pi_1(a_1 \mid s_I) = \pi_2(a_2 \mid s_I) = 1$, the returns yielded by the two policies are respectively $\mathbf{V}^{\pi_1} = (10, 0)$ and $\mathbf{V}^{\pi_2} = (-5, 3)$; so there is not clearly one policy better than the other.

A policy π is *dominated* by another policy π' if its value from the initial state is lower on all the dimensions than those of the other. Formally, for two policies π, π', we write $\mathbf{V}^\pi \prec \mathbf{V}^{\pi'}$ iff $\mathbf{V}^\pi_i \leq \mathbf{V}^{\pi'}_i$ for all $i \in [N]$ and $\mathbf{V}^\pi_j < \mathbf{V}^{\pi'}_j$ for some $j \in [N]$, where $\prec$ is the Pareto-dominance operator. For instance, take π_3 so that $\pi_3(a_3 \mid s_I) = 1$, then π_3 is dominated by π_1 since $\mathbf{V}^{\pi_3} = (5, 0) \prec \mathbf{V}^{\pi_1}$ with $5 < 10$.

Therefore, the typical goal of a multi-objective RL agent is to find the set of *Pareto-optimal* policies, i.e., those that are not dominated by any other policies: $\Pi^\star = \left\{\pi \in \Pi \mid \nexists \pi' \text{ such that } \mathbf{V}^{\pi'} \prec \mathbf{V}^\pi\right\}$. There is a surjective mapping from $\Pi^\star$ to the *Pareto front* $\mathcal{F} = \left\{\mathbf{V}^\pi \mid \pi \in \Pi^\star\right\}$.

Learning the full set of Pareto-efficient policies $\Pi^\star$ requires that the policies $\pi^\star \in \Pi^\star$ are deterministic stationary policies [15]. This is useful in settings where stochastic policies are not desired, such as the management of a hydroelectric power plant. In that scenario, the decision maker does not want to be presented with a policy that has a probability of completely draining the water reservoir even if that policy is optimal, as it would have catastrophic consequences for nearby towns [9].

As concrete example, the set of non-dominated policies from the MDP of Fig. 1 is $\Pi^\star = \left\{\pi_1, \pi_2\right\}$, for which the corresponding Pareto front is $\mathcal{F} = \left\{(10, 0), (-5, 3)\right\}$.

In general, we call the images of any surjective mapping from Π to values a *solution set*, and any solution set mapped from non-dominated policies a *coverage set*. The Pareto front is thus the optimal coverage set of the problem.

2.2 Pareto Conditioned Networks

Using neural networks as function approximators in RL comes with challenges. One is that the target (e.g., the policy's optimal action) is not known in advance — as opposed to classical supervised learning where the ground-truth target is provided. As the behavior of the agent improves over time, the action used as target can change, often leading to hard-to-tune and brittle learners [6, 12].

The key idea behind Pareto Conditioned Networks (PCN) [14] is to use supervised learning techniques to improve the policy instead of resorting to temporal-difference learning. This is done by conditioning the policy on a *desired return* that should be achieved. Since we know the return — be it high or low — achieved by a trajectory, we know the sequence of optimal actions needed to reach said return. We can thus train a policy that, conditioned on a desired return, provides the optimal action to reach said return. By leveraging the generalization properties of neural networks, we can accumulate incrementally better experience by conditioning on increasingly higher reward-goals.

Architecture. PCN uses a single neural network that takes a tuple $\langle s, \hat{h}, \hat{\mathbf{R}}\rangle$ as input. They represent, for state s, the return $\hat{\mathbf{R}}$ that PCN should reach at the end of the episode, i.e. the *desired return*. The *desired horizon* $\hat{h}$ says how many timesteps should be executed before reaching $\hat{\mathbf{R}}$. At execution time, both $\hat{h}$ and $\hat{\mathbf{R}}$ are chosen by the decision maker for s_I. PCN's neural network has a separate output for each action $a_i \in \mathcal{A}$. Each output represents the confidence the network has that, by taking the corresponding action, the desired return will be reached in the desired number of timesteps. We can draw an analogy with a classification problem where the network should learn to classify $\langle s, \hat{h}, \hat{\mathbf{R}}\rangle$ to its corresponding label a_i.

Dataset. As for classification, PCN needs a labeled dataset with training examples to learn a mapping from input to label. PCN collects data from the trajectories experienced while exploring the environment. For each timestep t of the trajectory, we know the episode's horizon $h_t = T - t$. We can also compute the cumulative reward obtained from t onward, i.e., $\mathbf{R}_t = \sum_{i=t}^{T} \boldsymbol{r}_i$. Since for this trajectory executing a_t in s_t resulted in $\mathbf{R}_t$ in h_t timesteps, we add a datapoint with input $\langle s, \hat{h}, \hat{\mathbf{R}}\rangle = \langle s_t, h_t, \mathbf{R}_t\rangle$ and output $a = a_t$ to

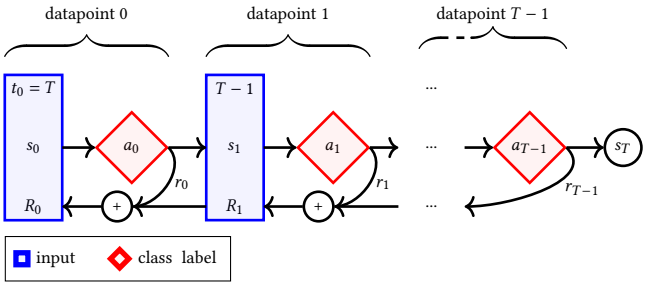


Figure 2: Conversion from a trajectory to labeled datapoints. For each timestep, we extract a single datapoint. The input (blue) is composed of the state at that timestep, and total return and number of timesteps until the end of the episode. The label (red) is the action taken at that timestep.

the dataset. In other words, when the observed return corresponds to the desired return in that state, then a_t is the optimal action to take. Fig. 2 shows how a trajectory is decomposed into datapoints.

One issue arises with this approach. In MORL, we aim to learn the set of Pareto dominating policies, expressed in Values. A policy's performance is thus measured in expectation. In contrast, PCN optimizes its policy on single trajectories, no matter how improbable the trajectory. For (MO)MDPs with a stochastic transition function, exploration can lead to high but improbable returns. When these returns Pareto-dominate other solutions, they are incorporated in PCN's solution set and proposed as target to the decision maker. However, upon selecting this target, the policy executions will, on average, result in lower returns.

We aim to tackle this issue by learning a model of the MOMDP, including its transition function. Using this model, we can evaluate the probability of a trajectory, which can help us determine if it worth incorporating it in the dataset. More importantly, we can simulate our policy on the learned model and estimate its value.

To this end, we take inspiration from Wasserstein Autoencoders (WAE). WAEs learn a latent model of the original MOMDP, conditioned on a policy π. This model presents several advantages. First, it learns a discretization of the original MOMDP. This allows us to execute Value Iteration (VI) on the model, providing us with accurate estimates for the policy's value. Second, WAEs provide theoretical bi-similarity guarantees allowing us to estimate the quality of the learned model and thus the computed values. Finally, we can apply formal methods on WAEs, providing us with reachability guarantees on the targets that make up the coverage set, improving the decision maker's understanding of the proposed policies.

2.3 Latent Space Modeling

Latent MDPs. Given the original (continuous or very large, possibly unknown) environment $\mathcal{M}$, a *latent space model* is another (tractable, explicit) MDP $\overline{\mathcal{M}} = \langle \overline{\mathcal{S}}, \mathcal{A}, \overline{\mathcal{T}}, \overline{\mathcal{R}}, \bar{s}_I, \gamma \rangle$ with state space linked to the original one via a *state embedding function*: $\phi \colon \mathcal{S} \to \overline{\mathcal{S}}$. We write $\overline{V}^{\bar{\pi}}$ for the value function derived from the execution of a latent policy $\bar{\pi} \colon \overline{\mathcal{S}} \to \Delta(\mathcal{A})$ in $\overline{\mathcal{M}}$. Furthermore, such policies can also be executed in the real model $\mathcal{M}$ via ϕ: intuitively, every time a state $s \in \mathcal{S}$ is visited, it is mapped to the latent space via $\bar{s} = \phi(s)$, and then $\bar{\pi}$ prescribes which action to choose via $a \sim \bar{\pi}(\cdot \mid \bar{s})$.

Optimal transport. Let $P, Q \in \Delta(\mathcal{X})$ be two distributions, their divergence can be measured according to the solution of the *optimal transport problem* (OT), which is intuitively the *minimum cost of changing P into Q* [17]. Formally, the OT is defined as $\mathcal{W}_c(P, Q) = \inf_{\lambda \in \Lambda(P,Q)} \mathbb{E}_{x,y\sim\lambda}\, c(x, y)$ where $c \colon \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a cost function and $\Lambda(P, Q)$ is the set of all *couplings* of P and Q. If c is equal to a distance metric d over $\mathcal{X}$, then $\mathcal{W}_d$ is the *Wasserstein distance* between the two distributions.

Wasserstein Auto-encoded MDPs (WAE-MDPs) [2] are parameterized latent space models that are trained based on the OT from trajectory distributions resulting from the execution of the RL agent's policy in the real environment $\mathcal{M}$ to that reconstructed from the latent model $\overline{\mathcal{M}}_\theta$. The optimization process relies on a temperature $\lambda \in [0, 1)$ that controls the continuity of the latent space learned, the zero-temperature corresponding to a *discrete latent space*. This procedure guarantees $\overline{\mathcal{M}}_\theta$ to be probably approximately *bisimilarly close* [3, 8, 11] to $\mathcal{M}$ as $\lambda \to 0$: in a nutshell, *bisimulation metrics* imply the closeness of the two models in terms of probability measures and value functions [4, 5]. In particular, a WAE-MDP learns the following components:

a *state embedding function* $\quad \phi_\iota \colon \mathcal{S} \to \overline{\mathcal{S}}$,

a *latent transition function* $\quad \overline{\mathcal{T}}_\theta \colon \overline{\mathcal{S}} \times \mathcal{A} \to \Delta(\overline{\mathcal{S}})$, and

a *latent reward function* $\quad \overline{\boldsymbol{\mathcal{R}}}_\theta \colon \overline{\mathcal{S}} \times \mathcal{A} \to \mathbb{R}^N$.

The objective function of WAE-MDPs — derived from the OT — incorporates *local losses* [7] that minimize the expected distance between the original and latent reward and transition functions:

$$L^{\mathcal{D}}_{\boldsymbol{\mathcal{R}}} = \mathop{\mathbb{E}}_{s,a\sim\mathcal{D}} \left\| \boldsymbol{\mathcal{R}}(s, a) - \overline{\boldsymbol{\mathcal{R}}}_\theta(\phi_\iota(s), a) \right\|_1,$$
$$L^{\mathcal{D}}_{\mathcal{T}} = \mathop{\mathbb{E}}_{s,a\sim\mathcal{D}} \mathcal{W}_{\bar{d}} \left(\phi_\iota \mathcal{T}(\cdot \mid s, a), \overline{\mathcal{T}}_\theta(\cdot \mid \phi_\iota(s), a) \right) \tag{1}$$

where $\mathcal{D} \in \Delta(\mathcal{S} \times \mathcal{A})$ is the distribution of experiences gathered by the RL agent when it interacts with $\mathcal{M}$, $\phi_\iota \mathcal{T}(\cdot \mid s, a)$ is the distribution of transitioning to $s' \sim \mathcal{T}(\cdot \mid s, a)$, then embedding it to the latent space $\bar{s}' = \phi_\iota(s')$, and $\bar{d}$ is a metric on $\overline{\mathcal{S}}$. To obtain the guarantees, $\mathcal{D}$ generally corresponds to the stationary distribution over states likely to be seen when a latent policy is executed.

3 A JOURNEY INTO THE LATENT SPACE

3.1 Multi-Objective WAE-MDPs

While WAE-MDPs have proven to learn accurate representations of single-objective MDPs, their properties might not hold for MOMDPs. In this section, we focus on two key questions: *"How well can WAE-MDPs model MDPs with multiple objectives?"* and *"Does the WAE-MDP latent space provide an effective representation for multi-objective RL?"*. We present theoretical bounds that are directly aimed at addressing these concerns. First, we show that when the local losses are minimized, the values of states in the original MDP and those of their embeddings in the latent MDP are close in average, demonstrating the quality of the latent model learned: not only the dynamics of the latent model are close to those of the original one, but the behaviors of the agent operating under any latent policy are

close (in terms of values) in the two models. Second, we show that in consequence, states that are mapped to the same latent representation have in fact close values. Therefore, the states clustering resulting from the embedding function is a suitable representation for optimizing (multi-objective) value functions.

Theorem 3.1 (Latent model quality). *Let $\mathcal{M}$ be an MO-MDP with state space $\mathcal{S}$, $\overline{\mathcal{M}}_\theta$ be a latent space model of $\mathcal{M}$ with state space $\overline{\mathcal{S}}$ and embedding function ϕ_ι, as well as $\overline{\pi}$ be a latent policy for $\overline{\mathcal{M}}_\theta$. Assume that the WAE-MDP is at the zero-temperature limit (i.e., $\lambda \to 0$) and let $\xi_{\overline{\pi}}$ be a stationary distribution of $\mathcal{M}$ as well as $K_{\overline{V}} = \sup_{s,a,i} \left|\overline{\boldsymbol{\mathcal{R}}}(s,a)_i\right| / 1-\gamma$, then*

$$\mathop{\mathbb{E}}_{s\sim\xi_{\overline{\pi}}} \left\| V^{\pi}(s) - \overline{V}^{\overline{\pi}}(\phi_\iota(s)) \right\|_1 \leq \frac{L^{\xi_{\overline{\pi}}}_{\boldsymbol{\mathcal{R}}} + \gamma N K_{\overline{V}} L^{\xi_{\overline{\pi}}}_{\mathcal{T}}}{1-\gamma}. \tag{2}$$

Notice that this upper bound goes to zero as $L^{\xi_{\overline{\pi}}}_{\boldsymbol{\mathcal{R}}}$ and $L^{\xi_{\overline{\pi}}}_{\mathcal{T}}$ go to zero.

Furthermore, let $\mathrm{G} \subseteq \mathcal{S}$ and $\overline{\mathrm{G}} \subseteq \overline{\mathcal{S}}$ be measurable sets of goal states so that, for all $s \in \mathrm{G}$, $\phi_\iota(s) = \bar{s} \implies \bar{s} \in \overline{\mathrm{G}}$. Then,

$$\mathop{\mathbb{E}}_{s\sim\xi_{\overline{\pi}}} \left| V^{\pi}_{\Diamond \mathrm{G}}(s) - \overline{V}^{\overline{\pi}}_{\Diamond \overline{\mathrm{G}}}(\phi_\iota(s)) \right| \leq \frac{\gamma L^{\xi_{\overline{\pi}}}_{\mathcal{T}}}{1-\gamma}. \tag{3}$$

Proof. The second inequality (Eq. 3) is a direct application of [3, Eq. 3 and Lem. B.3]. Concerning the first inequality (Eq. 2), let $L^{\xi_{\overline{\pi}}}_{\boldsymbol{\mathcal{R}}}(i) = \mathbb{E}_{s,a\sim\xi_{\overline{\pi}}} \left| \boldsymbol{\mathcal{R}}(s,a)_i - \overline{\boldsymbol{\mathcal{R}}}_\theta(\phi_\iota(s),a)_i \right|$ for all $i \in [N]$. Then:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s\sim\xi_{\overline{\pi}}} \left\| V^{\overline{\pi}}(s) - \overline{V}^{\overline{\pi}}(\phi_\iota(s)) \right\|_1 \\
&= \mathop{\mathbb{E}}_{s\sim\xi_{\overline{\pi}}} \left[\sum_{i=1}^{N} \left| V^{\overline{\pi}}(s)_i - \overline{V}^{\overline{\pi}}(\phi_\iota(s))_i \right| \right] && \text{(by def. of the } L_1 \text{ norm)} \\
&= \sum_{i=1}^{N} \mathop{\mathbb{E}}_{s\sim\xi_{\overline{\pi}}} \left| V^{\overline{\pi}}(s)_i - \overline{V}^{\overline{\pi}}(\phi_\iota(s))_i \right| \\
&\leq \sum_{i=1}^{N} \frac{L^{\xi_{\overline{\pi}}}_{\boldsymbol{\mathcal{R}}}(i) + \gamma K_{\overline{V}} L^{\xi_{\overline{\pi}}}_{\mathcal{T}}}{1-\gamma} && \text{(by [3, Eq. 3 and Lem. B3])} \\
&= \frac{L^{\xi_{\overline{\pi}}}_{\boldsymbol{\mathcal{R}}} + \gamma N K_{\overline{V}} L^{\xi_{\overline{\pi}}}_{\mathcal{T}}}{1-\gamma}. && \text{(by Eq. 1 and definition of the } L_1 \text{ norm)}
\end{aligned}$$

□

Corollary 3.1.1 (Representation quality). *For any two states $s_1, s_2 \in \mathcal{S}$ with $\phi_\iota(s_1) = \phi_\iota(s_2)$,*

$$\left\| V^{\overline{\pi}}(s_1) - V^{\overline{\pi}}(s_2) \right\|_1 \leq \frac{L^{\xi_{\overline{\pi}}}_{\boldsymbol{\mathcal{R}}} + \gamma N K_{\overline{V}} L^{\xi_{\overline{\pi}}}_{\mathcal{T}}}{1-\gamma} \left(\xi^{-1}_{\overline{\pi}}(s_1) + \xi^{-1}_{\overline{\pi}}(s_2) \right),$$

$$\left| V^{\overline{\pi}}_{\Diamond \mathrm{G}}(s_1) - V^{\overline{\pi}}_{\Diamond \mathrm{G}}(s_2) \right| \leq \frac{\gamma L^{\overline{\pi}}_{\mathcal{T}}}{1-\gamma} \left(\xi^{-1}_{\overline{\pi}}(s_1) + \xi^{-1}_{\overline{\pi}}(s_2) \right).$$

This claim can be proven by mimicking the proof for the one of [3, Eq. 4]. Furthermore, those value difference bounds can be PAC-learned, in the same fashion as those of [3] to assess the model quality resulting from the WAE-MDP procedure.

3.2 The Target-Augmented MDP

As explained in Sec. 2.2, PCN conditions its network on a target return $\hat{\mathbf{R}}$ and target horizon $\hat{h}$. Implicitly, this network holds infinitely many policies, as we can choose any target, and PCN chooses the action accordingly. In comparison, a WAE-MDP learns a model based on the traces of a single policy. To reconcile the differences between these two approaches, i.e., the multiple policies that PCN learns and the single policy the WAE-MDP depends on, we propose a mapping of the MOMDP to another MOMDP such that all of PCN's different policies of the original MOMDP are encompassed into a single policy of the transformed MOMDP.

Concretely, we enable optimizing a WAE-MDP through traces generated via a latent policy, solely conditioned on latent states, by incorporating the conditioning on target returns and horizons straight to the latent space. This is done by augmenting the original MOMDP's state-space with the target returns and horizons. Finally, we define a new start-state, s_{reset}, from which we can transition to the original MOMDP's start-state augmented with a $\langle \hat{\mathbf{R}}, \hat{h} \rangle$ pair. Intuitively, every time the environment is reset, a target is sampled from some target distribution. Then, $\mathcal{M}^{\uparrow}$ keeps track of the current target in its state space: it is merely the MDP encoding the process of Fig. 2. Finally, when the horizon is zero, it means that a terminal state is reached, and the environment is reset.

Formally, given an MOMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, s_I, N, \gamma \rangle$ and a target set $\mathbb{T} \subseteq \mathbb{N} \times \mathbb{R}^N$, we learn the WAE-MDP from the MOMDP $\mathcal{M}^{\uparrow} = \langle \mathcal{S}^{\uparrow}, \mathcal{A}, \mathcal{T}^{\uparrow}, \boldsymbol{\mathcal{R}}^{\uparrow}, s_{\text{reset}}, N, \gamma \rangle$ so that the augmented state is $\mathcal{S}^{\uparrow} = (\mathcal{S} \times \mathbb{N} \times \mathbb{R}^N) \cup \{s_{\text{reset}}\}$, where a special *reset state* s_{reset} is embedded which indicates that the environment has been reset. The transitions to that reset state are

$$\mathcal{T}^{\uparrow}(s_{\text{reset}} \mid s^{\uparrow}, a) = 1 \text{ for any } s^{\uparrow} \in \{\langle s, h, \mathbf{R} \rangle \mid h = 0\} \subseteq \mathcal{S}^{\uparrow},$$

transitions from the reset state are

$$\mathcal{T}^{\uparrow}(\{\langle s_I, h, \mathbf{R} \rangle \mid \langle h, \mathbf{R} \rangle \in \mathbb{T}\} \mid s_{\text{reset}}, a) = 1,$$

and transitions from one augmented state to another are

$$\begin{aligned}
&\mathcal{T}^{\uparrow}\left(\left\langle s', h-1, \mathbf{R}' \right\rangle \mid \langle s, h, \mathbf{R} \rangle, a\right) \\
&= \mathop{\mathbb{E}}_{s_I, h_I, \mathbf{R}_I \sim \mathcal{T}^{\uparrow}(\cdot \mid s_{\text{reset}}, a)} \left[\mathcal{T}\left(s' \mid s, a\right) \cdot \mathbb{1}_{=}\left(\mathbf{R}', \mathbf{R} - \gamma^{h_I - h} \cdot \boldsymbol{\mathcal{R}}(s,a)\right) \right].
\end{aligned}$$

Finally, the augmented reward function is

$$\boldsymbol{\mathcal{R}}^{\uparrow}(\langle s, h, \mathbf{R} \rangle, a) = \boldsymbol{\mathcal{R}}(s,a) \text{ for all } \langle s, h, \mathbf{R} \rangle \in \mathcal{S}^{\uparrow}, a \in \mathcal{A}.$$

Playing around value functions. The bounds derived in Sec. 3.1 are also valid for $\mathcal{M}^{\uparrow}$: the embedding function ϕ_ι maps states from $\mathcal{S}^{\uparrow}$ to $\overline{\mathcal{S}}$ while $\overline{\mathcal{M}}_\theta$ models the dynamics of $\mathcal{M}^{\uparrow}$. In particular, Corollary 3.1.1 ensures that the representation obtained by minimizing the local losses is suited to optimize the multi-objective Values during any MORL process, and Thm 3.1 enables the computation of the multi-objective return in the latent space model to check the behaviors of the agent operating under the latent policy. In particular, in the zero-temperature limit, VI can be used to compute the values of any state of $\overline{\mathcal{M}}_\theta$. Furthermore, the goal set G of interest is the set of *achieved targets*, i.e., augmented states of the form $\langle s, h, \mathbf{R} \rangle$ where $h = 0$ *and* $\mathbf{R} = 0$: such states indicate that a target has been achieved in $\mathcal{M}^{\uparrow}$. We reserve a special bit in the latent space $\overline{\mathcal{S}}$ to enforce ϕ_ι to deviate all states from G to a part of the latent space $\overline{\mathrm{G}} \subseteq \overline{\mathcal{S}}$ dedicated to achieved targets: for any goal state $s \in \mathrm{G}$, this

ensures that, whenever $\phi_\iota(s) = \bar{s}$, $\bar{s} \in \overline{\mathrm{G}}$. Therefore, for any target $g = \langle \hat{h}, \hat{\mathbf{R}} \rangle \in \mathbb{T}$, we can check *the probability of achieving g in the latent space* by computing the value $\overline{V}^{\overline{\pi}}_{\Diamond \overline{G}}\left(\phi_\iota\left(s_I, \hat{h}, \hat{\mathbf{R}}\right)\right)$.

4 WAE-PCN

We have shown in Sec. 3.1 that the difference between the value function of the policy π that interacts with the original MOMDP and the value function of the policy $\overline{\pi}$ that interacts with the latent model is bounded. In our case, we combine π with the encoder as latent policy, so the bound solely depends on the latent model error.

Thus, the latent model offers representation quality guarantees on the MOMDP. Combined with the augmented MOMDP described in Sec. 3.2, we can encode PCN's non-dominated policies into a single policy, thus adhering to the assumptions of Thm. 3.1.

Theoretically, we can train a set of Pareto-efficient policies using PCN, and estimate their Value and the target's reachability probability. Policies for which the Value does not match the target (due to the stochasticity of the transition function) can be filtered out. However, this does not allow to keep alternative policies, had the mismatching policies been filtered out *during the learning process.*

Our goal is thus to combine the learning of the WAE-MDP with the learning of PCN such that, by using WAE-MDP's estimated Values, mismatching policies are pruned out during PCN's search for non-dominated policies. We call this novel algorithm WAE-PCN.

The key differences of our algorithm with PCN are in blue.

Learning $\overline{\pi}$. To remove potential discrepancies between π (learned by PCN) and its distillation $\overline{\pi}$, *we directly learn* $\overline{\pi}$. PCN uses $\overline{\pi}$ by first encoding s into $\bar{s} = \phi_\iota(s)$. When PCN updates $\overline{\pi}$, we keep ι fixed, as to not modify the latent space during policy updates.

Incorporating s_{reset}. WAE-MDPs assume the original (MO)MDP is ergodic. We can enforce this property through a special s_{reset} state. To include s_{reset} in the latent space, we artificially add a final transition at the end of each trajectory from the final state to s_{reset}.

Keeping separate replay buffers. PCN keeps the best performing trajectories in its buffer (in terms of Pareto dominance), encountered by exploring with a stochastic policy and optimistic targets. PCN learns deterministic policies that should imitate these trajectories. However, the current $\overline{\pi}$ might not have converged to this desired behavior. In contrast, WAE-MDP learns its model based on traces of the final policy. Moreover, since only the best trajectories are kept, the transitions in PCN's buffer do not match the MOMDP's state-transition distribution, which is crucial for WAE-MDPs. Finally, PCN's buffer relabels the targets so they can be used to improve the policy. But executing the current policy on these targets might lead to different trajectories. Thus, WAE-MDP requires the original labels. Thus, after an exploration phase with optimistic targets for PCN, we add an exploitation phase where, for each currently non-dominated target, we execute the deterministic policy. These trajectories are added to a separate WAE-buffer.

Pruning the buffer with $\overline{V}^{\overline{\pi}}$. Since we aim to keep the best policies w.r.t. Values, we change PCN's pruning mechanism to prioritize trajectories from policies with non-dominated Values instead of returns. For this, WAE-PCN estimates the Values of $\overline{\pi}$ conditioned on the trajectory's target and uses it to set the trajectory's priority.

Algorithm 1 WAE-PCN

Require: WAE-MDP $\overline{\mathcal{M}}_\theta$ with buffer $\mathcal{B}_{\text{WAE}}$, PCN network $\overline{\pi}_\psi$ with buffer $\mathcal{B}_{\text{PCN}}$.

1: **for** $n \in [N]$ **do** ▷ fill buffers with random trajectories
2: sample trajectory $\tau = \langle s_{0:T}, a_{0:T-1}, \boldsymbol{r}_{0:T-1} \rangle$ using random policy π_n
3: $\tau_{T+1} \leftarrow \langle s_{\text{reset}}, a_T, \mathbf{0} \rangle$ ▷ terminal to reset
4: **for** $\tau_i \in \tau$ **do**
5: $\hat{h}, \hat{\mathbf{R}} = T - i, \sum_{t=i}^{T} \gamma^{t-i} \boldsymbol{r}_t$ ▷ target horizon, return
6: add $\tau_i, \hat{h}, \hat{\mathbf{R}}$ to $\mathcal{B}_{\text{PCN}}$, $\mathcal{B}_{\text{WAE}}$
7: **while** True **do**
8: **for** $m \in [M]$ **do** ▷ policy and model updates
9: **for** $w \in [W]$ **do**
10: update $\overline{\mathcal{M}}_\theta$ using $\mathcal{B}_{\text{WAE}}, \overline{\pi}_\psi$
11: update $\overline{\pi}_\psi$ using $\mathcal{B}_{\text{PCN}}$
12: **for** $\tau_i \in \mathcal{B}_{\text{PCN}}$ **do**
13: $\mathbf{V}^{\overline{\pi}i} \leftarrow \overline{V}^{\overline{\pi}\psi}\left(\phi_\iota\left(s_0, \hat{h_0}, \hat{\mathbf{R}_0}\right)\right)$
14: prune $\mathcal{B}_{\text{PCN}}$ using $\mathbf{V}^{\overline{\pi}}$.
15: select $\hat{h}, \hat{\mathbf{R}}$ based on $\mathcal{B}_{\text{PCN}}$ ▷ optimistic targets
16: **for** $n \in [N]$ **do** ▷ exploration
17: sample $\tau = \langle s_{0:T}, a_{0:T-1}, \boldsymbol{r}_{0:T-1} \rangle$ using $\overline{\pi}_\psi$ with $\phi_\iota, \hat{h}, \hat{\mathbf{R}}$
18: $\tau_{T+1} \leftarrow \langle s_{\text{reset}}, a_T, \mathbf{0} \rangle$ ▷ terminal to reset
19: **for** $\tau_i \in \tau$ **do**
20: add $\tau_i, T, \mathbf{R}_0$ to $\mathcal{B}_{\text{PCN}}$
21: **for** $\hat{h}, \hat{\mathbf{R}} \in \mathcal{F}$ **do** ▷ exploitation
22: **for** $n \in [N]$ **do**
23: sample $\tau = \langle s_{0:T}, a_{0:T-1}, \boldsymbol{r}_{0:T-1} \rangle$ using $\overline{\pi}_\psi$ with $\phi_\iota, \hat{h}, \hat{\mathbf{R}}$
24: $\tau_{T+1} \leftarrow \langle s_{\text{reset}}, a_T, \mathbf{0} \rangle$ ▷ terminal to reset
25: **for** $\tau_i \in \tau$ **do**
26: add $\tau_i, \hat{h}, \hat{\mathbf{R}}$ to $\mathcal{B}_{\text{WAE}}$

We depict the algorithm in Algorithm 1. We highlight in blue the additions to the original PCN algorithm to incorporate the WAE-MDP. In lines 1-9, the buffers are initialized with random trajectories. After this initial stage, we enter a loop that repeats three processes. First, we update the models (lines 11-16). Secondly, we estimate the Values of the policy responsible for each trajectory in the replay buffer and prune the buffer based on these Values (lines 17-20). Thirdly, we select an optimistic target based on the current Values from the buffer and we execute the updated PCN policy conditioned on this target (lines 21-28). Finally, we execute each currently non-dominated policy to fill the WAE-buffer (lines 29-37).

5 EXPERIMENTS

To understand the properties of our proposed algorithm, we devise a simple tabular MOMDP with a stochastic transition function, such that we can encounter trajectories with high returns with low probability. Trying to imitate these trajectories results in a worse expected return. Concretely, we inspire ourselves from Deep-Sea-Treasure [16], a classic benchmark MOMDP. In this environment, a submarine seeks for treasure at the bottom of the sea. Each timestep it consumes 1 unit of fuel. Further away treasures have a higher worth, leading to different fuel/treasure trade-offs.

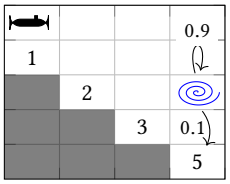
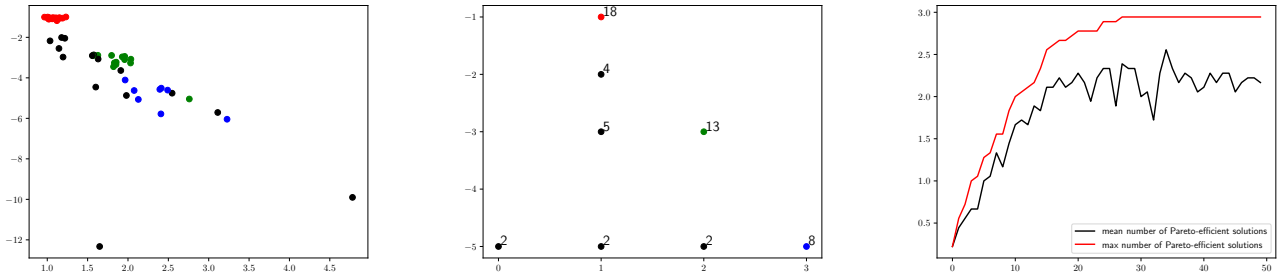


Figure 3: Modified DST.

For this environment, we change the dynamics of a single state. At one location of the ocean, there is a whirlpool. Entering the



(a) Estimated V for all the policies across all the performed experiments.

(b) Returns obtained by executing each policy, for each run. Each number denotes the number of policies reaching the matching point.

(c) Number of Pareto efficient policies over training time. Red illustrates that each run learns the full Pareto front during training.

Figure 4: All policies obtained across all experiments. We performed 18 experiments in total. The Pareto front contains 3 solutions, depicted in red, green and blue. The black solutions depict sub-optimal policies. For clarity, we omit the 5 policies that are unable to discover a return, wandering forever in the environment.

whirlpool results in a 90% of damaging the submarine, forcing it to return empty handed, and thus ending the episode. However, with a 10% probability, the submarine is transported to the most valuable treasure of the environment. The environment is depicted in Fig. 3. There exist 4 different treasures, leading to 4 different trade-offs: $(-1, 1)$, $(-3, 2)$, $(-5, 3)$, $(-5, 5)$, with corresponding policies $\pi_1, \pi_2, \pi_3, \pi_4$ respectively. We note that we can only reach $(-5, 5)$ through the whirlpool. When only considering the trajectories, the 3 non-dominated trade-offs are $(-1, 1)$, $(-3, 2)$, $(-5, 5)$. However, due to the whirlpool, $\mathbf{V}^{\pi_1} = (-5, 0.5)$, in which case the Pareto front is $(-1, 1)$, $(-3, 2)$, $(-5, 3)$. While PCN learns the first coverage set, we aim for WAE-PCN to learn the second.

5.1 Results

We perform 18 experiments on our modified DST environment. In Fig. 4, we plot the learned policies for each experiment. On the left, we plot the latent Values for each learned policy. The center plot shows the returns obtained after executing the policy. The solutions that are part of $\mathcal{F}$ are plotted in a separate colors, while the other ones are plotted in black.

First, we see that 8/18 experiments discover the full Pareto front, since 8 policies reach $(-5, 3)$. However, due to inaccuracies in latent Value, some of these experiments learn to keep more than 3 non-dominated policies. For example, 2 experiments learn a fourth policy that aims for $(-5, 5)$, even though they already have a policy going to $(-5, 3)$. Still, not a single experiment learns to aim for $(-5, 5)$ while discarding $(-5, 3)$, which is what vanilla PCN learns.

Next, we analyze the latent Values. We observe that, for $(-1, 1)$ and $(-3, 2)$, estimates are accurate. For $(-5, 3)$ they are less so: they correctly estimate the -5 fuel consumption, but are pessimistic about the treasure value, with estimates in range $[2, 2.5]$. This is close to the 2nd treasure but with worse fuel consumption, which might explain why WAE-PCN has difficulties in keeping π_3.

Finally, Fig. 4c shows the evolution of the learned coverage set over time. On average, the size of the learned coverage set is higher than 2, but does not reach 3. This is because, even after learning $\mathcal{F}$, WAE-PCN forgets policies and has to relearn them. With this insight, we plot in red the best learned coverage set over time for each experiment. This curve, which converges to 2.95, shows that almost all experiments learn the full Pareto front during training.

6 DISCUSSION & CONCLUSION

We show that WAE-PCN can learn the Pareto front and produce reliable latent Values. However, this currently applies on 8/18 experiments. Many experiments learn a subset of the Pareto front and consider some sub-optimal policies as optimal, due to distant latent Values. E.g., often the policy that leads to $(-2, 1)$ is considered non-dominated since it produces Values close to $(-1, 1)$.

Moreover, most of the experiments learn all the Pareto-efficient policies sometime during the training procedure, but unlearn them as they are considered sub-optimal. Without this instability, 17/18 experiments would have fully succeeded.

We believe this instability results from the dual dependency between PCN and WAE-MPDs, which can lead to conflicting learning behavior. PCN learns $\overline{\pi}$ in the WAE-MDP's latent state-space. When we update the WAE-MDP, the latent representation changes, which impacts $\overline{\pi}$. With a changed policy, the traces that PCN produces (lines 22-27) might become unreliable. This results in a change in training data in $\mathcal{B}_{\text{WAE}}$, which modifies the learned transition function. This in turn impacts the estimated Values, which alters the priorities of trajectories in $\mathcal{B}_{\text{PCN}}$ and thus its pruning. Finally, this results in a different set of targets that PCN learns to imitate.

We cope with this instability by intertwining the updates of both components. Moreover, we keep low learning rates and use decaying learning rate to avoid drastic changes between each update.

The clear future direction of our work is stabilizing the learning procedure. We foresee three possible avenues of work: (i) we aim to stabilize the pruning process by including reachability probabilities; (ii) we take inspiration from adverserial learning; (iii) we cope with the stabilization problem with RL solutions for the moving target problem (e.g., Polyak averaging).

REFERENCES

[1] A Castelletti, Francesca Pianosi, and Marcello Restelli. 2013. A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research* 49, 6 (2013), 3476–3486.

[2] Florent Delgrange, Ann Nowe, and Guillermo Perez. 2023. Wasserstein Auto-encoded MDPs: Formal Verification of Efficiently Distilled RL Policies with Many-sided Guarantees. In *International Conference on Learning Representations*. https://openreview.net/forum?id=JLLTtEdh1ZY

[3] Florent Delgrange, Ann Nowé, and Guillermo A. Pérez. 2022. Distillation of RL Policies with Formal Guarantees via Variational Abstraction of Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 6 (Jun. 2022), 6497–6505. https://doi.org/10.1609/aaai.v36i6.20602

[4] Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. 2004. Metrics for labelled Markov processes. *Theor. Comput. Sci.* 318, 3 (2004), 323–354. https://doi.org/10.1016/j.tcs.2003.09.013

[5] Norm Ferns, Prakash Panangaden, and Doina Precup. 2011. Bisimulation Metrics for Continuous Markov Decision Processes. *SIAM J. Comput.* 40, 6 (2011), 1662–1714. https://doi.org/10.1137/10080484X

[6] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. 2019. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*. PMLR, 2021–2030.

[7] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2170–2179. http://proceedings.mlr.press/v97/gelada19a.html

[8] Robert Givan, Thomas L. Dean, and Matthew Greig. 2003. Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.* 147, 1-2 (2003), 163–223. https://doi.org/10.1016/S0004-3702(02)00376-4

[9] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. 2021. A practical guide to multi-objective reinforcement learning and planning. *arXiv preprint arXiv:2103.09568* (2021).

[10] Bojun Huang. 2020. Steady State Analysis of Episodic Reinforcement Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/69bfa2aa2b7b139ff581a806abf0a886-Abstract.html

[11] Kim Guldstrand Larsen and Arne Skou. 1989. Bisimulation Through Probabilistic Testing. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*. ACM Press, 344–352. https://doi.org/10.1145/75277.75307

[12] 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. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[13] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley. https://doi.org/10.1002/9780470316887

[14] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. 2022. Pareto Conditioned Networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1110–1118.

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

[16] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning* 84, 1-2 (2011), 51–80.

[17] Cédric Villani. 2009. *Optimal Transport: Old and New*. Springer Berlin Heidelberg, Berlin, Heidelberg. 93–111 pages. https://doi.org/10.1007/978-3-540-71050-9_6