First-Order Methods for Wasserstein Distributionally Robust MDPs

Julien Grand-Clément [1] Christian Kroer [1]

Abstract

Markov decision processes (MDPs) are known to be sensitive to parameter specification. Distributionally robust MDPs alleviate this issue by allowing for *ambiguity sets* which give a set of possible distributions over parameter sets. The goal is to find an optimal policy with respect to the worst-case parameter distribution. We propose a framework for solving Distributionally robust MDPs via first-order methods, and instantiate it for several types of Wasserstein ambiguity sets. By developing efficient proximal updates, our algorithms achieve a convergence rate of $O\left(NA^{2.5}S^{3.5}\log(S)\log(\epsilon^{-1})\epsilon^{-1.5}\right)$ for the number of kernels N in the support of the nominal distribution, states S, and actions A; this rate varies slightly based on the Wasserstein setup. Our dependence on N, A and S is significantly better than existing methods, which have a complexity of $O\left(N^{3.5}A^{3.5}S^{4.5}\log^2(\epsilon^{-1})\right)$. Numerical experiments show that our algorithm is significantly more scalable than state-of-the-art approaches across several domains.

1. Introduction

In many applications of sequential decision-making problems, the dynamics of the environment can only be partially modeled, because of statistical errors and inaccurate distributional information regarding the parameters of the model. This occurs, for example, in healthcare applications (Grand-Clément et al., 2020; Steimle et al., 2018) and vehicle routing (Miao et al., 2017). In *Markov Decision Processes* (MDPs), this can be addressed using robust formulations, where the transition probabilities belong to a safety region called the *uncertainty set* (Iyengar, 2005; Nilim & Ghaoui, 2005; Wiesemann et al., 2013; Goyal & Grand-Clément, 2018). However, robust MDPs often compute conservative policies, as they optimize only for the *worst-case* kernel realization, without incorporating *distributional* information about uncertainties.

Distributionally Robust MDPs (DR-MDPs) (Xu & Mannor, 2010; Yu & Xu, 2015) attempt to overcome the conservative nature of robust MDPs. In DR-MDPs the goal is to maximize the worst-case *expected* reward, assuming that the distribution over the set of possible transition kernels is not known, but belongs to a so-called *ambiguity set* consisting of all the possible measures over transition kernels. Robust MDPs can be viewed as a special case of DR-MDP, where the distribution over the set of possible kernels is restricted to Dirac masses. Yang (2017) introduces a Wasserstein ball formulation for ambiguity sets, shows the existence of an optimal policy that is Markovian, and gives a Value Iteration (VI) algorithm based on iterating a Bellman equation. Wasserstein distances have been shown to be particularly useful when the data is too sparse to use moment-based ambiguity sets (Gao & Kleywegt, 2016; Esfahani & Kuhn, 2018; Zhao & Guan, 2018).

One drawback of the Value Iteration approach to solving DR-MDPs is that every iteration of the algorithm requires solving the associated Bellman equation. Yang (2017) shows that this Bellman equation can be reformulated as a finite-dimensional convex program with a max-min objective. In the special case of DR-MDP policies for Wasserstein balls with a finite number of states and actions and s-rectangular ambiguity sets, it is possible to derive a large conic convex program using standard optimization methods. Letting N be the number of kernels in the support of the nominal distribution over the set of possible kernels, S the number of states, and A the number of actions of the MDP, VI with such a conic convex reformulation (solved using standard interior-point methods) returns an ϵ-optimal policy in $O\left(N^{3.5}A^{3.5}S^{4.5}\log^2(\epsilon^{-1})\right)$ time, for Wasserstein uncertainty based on the ℓ_2-metric. The same complexity results hold for Wasserstein uncertainty based on ℓ_1 and ℓ_∞ metric, see end of Section 2.1. This time complexity is largely due to the expensive per-iteration cost of interior-point methods. This may prove prohibitively slow when the MDP instance or the number of kernels is large.

In this paper, our goal is to design algorithms based on first-order methods (FOMs), which are typically more scalable

*Equal contribution [1]IEOR Department, Columbia University. Correspondence to: Julien Grand-Clément <jg3728@columbia.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

(at the cost of lower precision in the final solution). Recently, Grand-Clément & Kroer (2021) introduced FOMs to solve *robust* MDPs. Their algorithms adapt FOMs for solving static zero-sum games to the dynamic setting of MDP. Interleaving FOM updates with approximate VI updates, the authors obtain an algorithm that improves significantly on VI, in terms of dependence on S and A, at the price of a $O(1/\epsilon)$ convergence rate rather than $O(\log(1/\epsilon))$.

Our contributions

A First-Order Method for Distributionally Robust MDP. We build upon the Wasserstein framework for DR-MDP of Yang (2017) and on the first-order framework of Grand-Clément & Kroer (2021). Our algorithmic framework interleaves first-order steps and approximate Bellman updates. Our algorithm generates a sequence of iterates $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_T, \boldsymbol{y}_T)$, each of which is a policy $\boldsymbol{x}_t$ and an uncertainty instantiation $\boldsymbol{y}_t$. The t'th iterate is generated based on a first-order update on iterate $t-1$. This is achieved by computing the gradients for the first-order updates based on the linear objective arising from a value-vector estimate. By carefully interleaving approximate Bellman updates on this value-vector estimate, we show that the *average* of our generated *policy* iterates constructs a solution to the DR-MDP problem whose duality gap decreases at a rate of $O(1/T^{2/3})$ after T first-order updates. Note that this is different from the usual convergence guarantees for VI, which is on the *last iterate value vector*.

Our algorithmic framework attains a $O(1/T^{2/3})$ convergence rate in terms of the number of FOM steps T. As is expected with FOMs, this is worse than the $\log(1/\epsilon)$ rate achieved by VI. However, our dependence on N, A and S is better than VI by a factor of $O(N^{2.5}AS)$.

Novel proximal setup. A fundamental component in our scheme is to show that the iterate FOM updates can be computed very cheaply (in nearly linear time) for various ambiguity sets of interest. This is crucial in practice, since even a moderate number of states S, actions A, and kernels in the nominal estimate N, leads to a large MDP, whose instance size is $O(NAS^2)$. Since Wasserstein distances rely on a choice of *type* and *metric* (see next section), we show how to instantiate our FOM framework for several such Wasserstein ambiguity sets. We cover metrics based on the norms ℓ_1, ℓ_2, and ℓ_∞, as these are the most common found in the literature on Wasserstein distances. For each of these setups, we give novel algorithms that allow the proximal first-order iterates to be computed in nearly linear time.

Combining these proximal setups with our FOM framework yields an algorithm that, to the best of our knowledge, has the best convergence rates in terms of N, S and A for DR-MDPs with Wasserstein balls for any of the three metrics.

Empirical evaluation. We focus our numerical experiments on ℓ_2-based Wasserstein balls. We consider random MDPs, and applications to machine replacement and forest management. We compare our algorithms to four state-of-the-art Value Iteration algorithms (VI, Gauss-Seidel, Anderson, and Accelerated VI) and show that our algorithm is significantly faster. Even for small instances (e.g. $S = 10, N, A = 30$ or $N = 10$ and $S, A = 30$), our algorithm is at least twice as fast as Value Iteration. As instances get larger (both in terms of states/actions or number of observed kernels), our algorithm becomes much faster than all the VI variants. This is because the VI variants are solving large optimization programs for every state at every iteration, compared to our algorithm which only takes cheap primal-dual proximal steps.

Related works

Faster algorithms for MDPs. Accelerating the convergence rate of VI for regular MDPs has been studied extensively, e.g. in Zhang et al. (2018) and Goyal & Grand-Clément (2019). For robust MDPs, fast Bellman updates can be computed for s, a-rectangular uncertainty sets (Iyengar, 2005; Nilim & Ghaoui, 2005) and s-rectangular uncertainty sets (see Ho et al. (2018) for d_1-based uncertainty set). However, none of these algorithms extend directly to a setup with $N \geq 2$ kernels in the support of the nominal distribution, and they do not modify the Value Iteration algorithm itself. Grand-Clément & Kroer (2021) develop a FOM framework which outperforms value iteration for robust MDPs, when the size of the MDP instance is large. While this improves upon VI for large instances of *robust MDPs*, their methods do no directly extend to $N \geq 2$ (i.e. to distributionally robust MDPs) nor to Wasserstein balls. Exploiting the *linear programming* formulation of *non-robust* MDP, Gong & Wang (2020) and Jin & Sidford (2020) propose to adapt mirror descent algorithms to solve MDPs. There is no known linear programming reformulation for *robust* and *distributionally robust* MDPs. Finally, our work differs from value function approximation (Tsitsiklis & Van Roy, 1997; De Farias & Van Roy, 2003; Petrik, 2010; Tamar et al., 2014) in that we can control the desired accuracy of our inexact updates, contrary to value function approximation once the basis on the chosen subspace of functions is fixed. Additionally, unlike value function approximation, our algorithm improves convergence time even when the number of states and actions remain small, if there is a large number of kernels N.

Distributionally Robust MDPs. DR-MDPs were introduced in Xu & Mannor (2010). Yu & Xu (2015) considerably extend the expressiveness of the ambiguity sets (to e.g. mean absolute deviation and confidence sets) by using lifting methods developed in Wiesemann et al. (2014). Yang (2017) introduces Wasserstein DR-MDPs and presents a reformulation of the robust Bellman update based on Kantorovitch duality; however, the author appeals to general convex programming to solve the resulting min-max problem, which may not be tractable without exploiting further problem structure or reformulation. Our approach builds on the robust Bellman formulation of Yang (2017) by combining it with a tractable first-order setup. The authors in Chen et al. (2019) combine various ambiguity sets (among others moments, ϕ-divergences, and Wasserstein distances) and give a conic formulation for the Bellman equation for this combination of ambiguity sets.

Notation We let $P(X)$ be the set of all Borel probability measures on a set X. For $n \in \mathbb{N}$, $\Delta(n)$ is the probability simplex of dimension n. For $S, A \in \mathbb{N}$, we let $\mathcal{U} = (\Delta(S))^A$ be the Cartesian product of probability simplexes over states.

2. Distributionally Robust MDP

A Distributionally Robust MDP (DR-MDP) is a tuple $(\mathbb{S}, \mathbb{A}, \boldsymbol{c}, \boldsymbol{p}_0, \lambda, \mathbb{D})$; $\mathbb{S}$ is the set of states and $\mathbb{A}$ is the set of actions. We assume a *finite* set of states and actions: $|\mathbb{S}| = S < +\infty, |\mathbb{A}| = A < +\infty$. There is a state-action cost $\boldsymbol{c} \in \mathbb{R}^{|\mathbb{S}|\times|\mathbb{A}|}$, an initial distribution over the set of states $\boldsymbol{p}_0 \in \Delta(S)$ and a discount factor λ. The transition rates $(\boldsymbol{y}_{sa})_{s,a} \in (\Delta(S))^{S\times A}$ are unknown; instead, we assume that they follow a joint probability distribution μ, which is known to belong to an *ambiguity set* $\mathbb{D}$. This distribution μ is typically estimated from historical data (see next section). The goal of the decision maker is to compute a policy $\boldsymbol{x}$ in $\Pi = (\Delta(A))^S$, which maps each state s to a distribution over actions, so as to minimize the worst-case infinite-horizon discounted cost, defined as $C(\boldsymbol{x}, \mu) = \mathbb{E}_{\boldsymbol{x}}\mathbb{E}_{\boldsymbol{y}\sim\mu}[\sum_{t=0}^{+\infty} \lambda^t c_{s_t a_t} | s_0 \sim \boldsymbol{p}_0]$. Specifically, we want to solve

$$\min_{\boldsymbol{x}\in\Pi} \max_{\mu\in\mathbb{D}} C(\boldsymbol{x}, \mu). \tag{1}$$

We focus on the case of *s-rectangular* ambiguity, where the uncertainty about transitions is independent across states. Formally, $\mathbb{D} = \{\mu \mid \mu = \bigotimes \mu_s, \mu_s \in \mathbb{D}_s, \forall s \in \mathbb{S}\}$, where for each state $s \in \mathbb{S}$ the set $\mathbb{D}_s$ is a set of probability distributions over the parameters $(\boldsymbol{y}_{sa})_{a=1}^A \in (\Delta(S))^A$ and $\bigotimes$ stands for the product over measures. This is a standard assumption in the literature, as related transition rates across different states lead to intractable problems in general (Wiesemann et al., 2013).

As detailed in Yu & Xu (2015) and Yang (2017), the *value vector* $\boldsymbol{v}^*$ of a solution $(\boldsymbol{x}^*, \mu^*)$ to (1) satisfies the following Bellman equation:

$$v_s^* = \min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\mu_s\in\mathbb{D}_s} \mathop{\mathbb{E}}_{\boldsymbol{y}_s\sim\mu_s}\left[\sum_{a\in\mathbb{A}} x_{sa}\left(c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}^*\right)\right]. \tag{2}$$

Moreover, $(\boldsymbol{x}^*, \mu^*)$ can be recovered as the optimal solutions in the right-hand min-max problem in (2). Since $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \sum_{a\in\mathbb{A}} \boldsymbol{x}_{sa}\left(c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}^*\right)$ is bilinear, the Bellman equation depends on μ_s only through $\mathbb{E}_{\boldsymbol{y}_s\sim\mu_s}[\boldsymbol{y}_s]$ (Yu & Xu, 2015). By linearity of expectation, we may maximize over the set of possible expected values for $\boldsymbol{y}_s$ instead:

$$v_s^* = \min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\boldsymbol{y}_s\in\mathbb{B}_s} \sum_{a\in\mathbb{A}} x_{sa}\left(c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}^*\right), \tag{3}$$

where $\mathbb{B}_s = \{\boldsymbol{y}_s \mid \exists\, \mu_s \in \mathbb{D}_s \text{ s.t. } \boldsymbol{y}_s = \mathbb{E}_{\hat{\boldsymbol{y}}_s\sim\mu_s}[\hat{\boldsymbol{y}}_s]\}$.

2.1. Wasserstein Distributionally Robust MDP

We will investigate the case where the sets of densities $\mathbb{D}_s$ are defined by Wasserstein distances. For single-state distributionally robust optimization and chance-constrained problems, this distance has proved useful when the number of data points is too small to rely on moment estimation of the underlying distribution (Gao & Kleywegt, 2016; Esfahani & Kuhn, 2018). In particular, a Wasserstein ball contains both continuous and discrete distributions while balls based on ϕ-divergences (e.g. Kullback-Leibler divergence) centered at a discrete distribution do not contain relevant continuous distributions. Additionally, ϕ-divergences do not take into account the closeness of two distributions, contrary to Wasserstein distance. Finally, by choosing a *metric* accordingly (see definition below), the Wasserstein distance can account for the underlying geometry of the space that the distributions are defined on.

Let us define Wasserstein distances and balls. The Wasserstein distance $W_p(\mu, \nu_s)$ between two distributions μ and ν_s is defined with respect to a *metric* d and a type $p \in \mathbb{N}$ as

$$\begin{aligned} W_p(\mu, \nu_s) = \min\ & \left(\mathbb{E}_{(x,y)\sim\kappa}\left[d(x,y)^p\right]\right)^{1/p} \\ & \kappa \in P(\mathcal{U}\times\mathcal{U}), \\ & \Pi_1\kappa = \mu, \Pi_2\kappa = \nu_s. \end{aligned}$$

where $\Pi_1\kappa$ and $\Pi_2\kappa$ are the first and second marginals for a density κ on $\mathcal{U}\times\mathcal{U}$. When $p \to +\infty$, we have the pointwise convergence $W_p \to W_\infty$ (Givens et al., 1984) where

$$\begin{aligned} W_\infty(\mu, \nu_s) = \min\ & \kappa\text{-ess.sup}(d) \\ & \kappa \in P(\mathcal{U}\times\mathcal{U}), \\ & \Pi_1\kappa = \mu, \Pi_2\kappa = \nu_s, \end{aligned}$$

with κ-ess.sup(d) defined as

$$\inf\{c \in \mathbb{R} \mid \kappa\left(\{(x,y) \mid d(x,y)) > c\}\right) = 0\}.$$

We will be interested in the norm-based metrics $d_1 = \ell_1, d_2 = \ell_2$ and $d_\infty = \ell_\infty$.

We assume that we have a nominal estimate $\nu \in \mathbb{D}$ of the distribution over the transition rates. Additionally, we assume that ν has finite support, i.e. for each s, $\nu_s = (1/N)\sum_{i=1}^N \delta_{\hat{\boldsymbol{y}}_{i,s}}$, for some observed kernels

$\hat{\boldsymbol{y}}_{1,s}, ..., \hat{\boldsymbol{y}}_{N,s} \in \mathcal{U} = (\Delta(S))^A$. This occurs, for example, when ν is the empirical distribution over N samples of the transition kernels, obtained from observed, historical data (Yang, 2017). Note that here, each $\hat{\boldsymbol{y}}_{i,s}$ represents the collection $(\hat{\boldsymbol{y}}_{i,s,a})_{a\in\mathbb{A}}$ of distributions over the next states given state and action pairs. The ambiguity set $\mathbb{D}_{p,s}$ will be the set of all measures μ within some Wasserstein distance $W_p(\mu, \nu_s)$ of the nominal estimate:

$$\mathbb{D}_{p,s} = \{\mu \in P(\mathcal{U}) | W_p(\mu, \nu_s) \leq \theta^p\}. \tag{4}$$

In a small abuse of notation, we will let $\mathbb{D}_{\infty,s}$ denote the Wasserstein ball (4) based on W_∞ instead of W_p, with a radius of θ. Given a metric d, and $p \in \mathbb{R} \bigcup \{\infty\}$, the set of expected kernels for the measures μ in the Wasserstein ambiguity sets $\mathbb{D}_{p,s}$ can be described as (Yang, 2017; Bertsimas et al., 2018; Xie, 2020):

$$\mathbb{B}_{p,s} = \{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i | \frac{1}{N}\sum_{i=1}^{N} d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)^p \leq \theta^p, \boldsymbol{y}_i \in \mathcal{U}, \forall\, i\},$$

$$\mathbb{B}_{\infty,s} = \{\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i | d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) \leq \theta, \boldsymbol{y}_i \in \mathcal{U}, \forall i = 1, ..., N\}.$$

Computing an optimal policy Yang (2017) shows that for Wasserstein balls (with $p < +\infty$), there exists an optimal policy which is stationary and Markovian; we present a proof of this result for $p = +\infty$ in our Appendix D. Yang (2017) also gives a *Value Iteration* algorithm to compute an optimal value vector $\boldsymbol{v}^*$ by iterating the Bellman equation. In particular, let $F : \mathbb{R}^S \to \mathbb{R}^S$ be the Bellman operator

$$F(\boldsymbol{v})_s = \min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\boldsymbol{y}_s\in\mathbb{B}_{p,s}} \sum_{a\in\mathbb{A}} x_{sa}\left(c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}\right), \forall\, s \in \mathbb{S}. \tag{5}$$

The Value Iteration (VI) algorithm is defined as follow:

$$\boldsymbol{v}_0 \in \mathbb{R}^S, \boldsymbol{v}_{\ell+1} = F(\boldsymbol{v}_\ell), \forall\, \ell \geq 0. \tag{VI}$$

F is a contraction of factor λ and VI returns a sequence $(\boldsymbol{v}_\ell)_{\ell\geq 0}$ such that $\|\boldsymbol{v}^{\ell+1} - \boldsymbol{v}^*\|_\infty \leq \lambda \cdot \|\boldsymbol{v}^\ell - \boldsymbol{v}^*\|_\infty, \forall\, \ell \geq 0$; an ϵ-optimal policy and distribution over kernels can be computed as the pair attaining the $\min\max$ in $F(\boldsymbol{v})$, if $\|\boldsymbol{v} - F(\boldsymbol{v})\|_\infty < 2\lambda\epsilon(1-\lambda)^{-1}$ (Wiesemann et al., 2013).

In Appendix A, we show that (5) can be reformulated as a convex program by invoking convex duality twice. Thus, using an Interior Point Method (IPM), $F(\boldsymbol{v})$ can be computed in $O(N^{3.5}A^{3.5}S^{3.5}\log(\epsilon^{-1}))$ arithmetic operations (Ben-Tal & Nemirovski (2001), Section 4.6.1-4.6.2), for $d = d_1, d_2, d_\infty$. This leads to an overall complexity for Value Iteration to return an ϵ-optimal policy in $O(N^{3.5}A^{3.5}S^{4.5}\log^2(\epsilon^{-1}))$, which can be prohibitively large when the number of kernels, states, and actions grows.

3. First-Order Methods for Wasserstein DR-MDP

Our algorithm builds upon (VI), but avoids repeatedly solving expensive convex programs. At every VI epoch $\ell \geq 1$ (we refer to VI iterations as *epochs* to distinguish from FOM iterations), we have a value vector $\boldsymbol{v}^\ell$ and we use a FOM to compute an approximation of the Bellman update $F(\boldsymbol{v}^\ell)$. At VI epoch $\ell + 1$, we use our approximate solution to $F(\boldsymbol{v}^\ell)$ to warm-start the computation of an approximation to $F(\boldsymbol{v}^{\ell+1})$. We will show that the (weighted) average of the FOM strategies across *all* epochs converges to a solution to the Distributionally-Robust MDP problem (1).

It is important to note that our scheme is very different from the following simpler approach: run (VI), but use a FOM (instead of interior point methods) to solve each of the Bellman-equation problems. This would only converge in terms of the *value vector*, rather than in terms of the duality gap guarantee that we provide for the average of all pairs of policy-kernel visited (see Theorem 1). In particular, our analysis allows us to construct an average of *all* iterates generated across T FOM iterations and allows us to use this T in our convergence guarantee.

First, we rewrite the strategy space for the $\boldsymbol{y}$ player to explicitly be in terms of the individual components of the averaged vector $\boldsymbol{y} = \dfrac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_{i,s}$. Concretely, we rewrite $F(\boldsymbol{v})_s$ from (5) as

$$\min_{\boldsymbol{x}_s\in\Delta(A)} \max_{(\boldsymbol{y}_{1,s},...,\boldsymbol{y}_{N,s})\in\tilde{\mathbb{B}}_{p,s}} \sum_{a\in\mathbb{A}} \boldsymbol{x}_{sa}\left(c_{sa} + \lambda\sum_{i=1}^{N}\frac{1}{N}\boldsymbol{y}_{i,sa}^\top\boldsymbol{v}\right), \tag{6}$$

for $\tilde{\mathbb{B}}_{p,s} \subset \mathbb{R}^{N\times S\times A}$ defined as

$$\tilde{\mathbb{B}}_{p,s} = \{(\boldsymbol{y}_i)_{i=1}^N | \frac{1}{N}\sum_{i=1}^{N} d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)^p \leq \theta^p, \boldsymbol{y}_i \in \mathcal{U}, \forall\, i\}. \tag{7}$$

As we are now considering elements indexed by $i = 1, ..., N$, for the sake of conciseness we will write $(\boldsymbol{y}_i)_i$ for $(\boldsymbol{y}_i)_{i=1}^N$. This strategy space representation will be easier to design FOMs for.

Proximal Setup for First-Order Methods. Let us fix a state $s \in \mathbb{S}$, for which we solve (5). FOMs such as the one we consider rely on having a *proximal setup* for the convex and compact decision spaces $\Delta(A)$ (referred to as X for simplicity in this section) and $\tilde{\mathbb{B}}_s$ (referred to as Y).

Using ψ_X, we construct the *Bregman divergence* D_X, which measures a (pseudo) distance between any pair $\boldsymbol{x}, \boldsymbol{x}' \in X$ (D_Y is defined analogously):

$$D_X(\boldsymbol{x}, \boldsymbol{x}') = \psi_X(\boldsymbol{x}') - \psi_X(\boldsymbol{x}) - \langle\nabla\psi_X(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x}\rangle,$$

The convergence rate depends on the *set widths* Θ_X, Θ_Y, which are the maxima of D_X and D_Y on $X \times X$ and $Y \times Y$. We will also require the maximum norm-magnitude $R_X = \max_{x \in X} \|x\|_X$, with R_Y defined analogously.

We will pay particular attention to the *Euclidean case*, where $(\|\cdot\|_X, \|\cdot\|_Y) = (\psi_X, \psi_Y) = (\ell_2, \ell_2)$, though Algorithm 1 applies more broadly (for example, a proximal setup with the ℓ_1 norm is also possible). The Bregman divergences are

$$D_X(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2,$$
$$D_Y((\boldsymbol{y}_i)_i, (\boldsymbol{y}'_i)_i) = \sum_{i=1}^{N} \frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{y}'_i\|_2^2. \tag{8}$$

Given a proximal setup, a crucial component of the FOMs we are interested in is the *proximal mapping*, which can effectively be thought of as a generalization of taking a step from the previous iterate in the direction of improvement along the gradient $\boldsymbol{g}$:

$$\text{prox}_x(\boldsymbol{g}_x, \boldsymbol{x}'_s) = \arg\min_{\boldsymbol{x}_s \in X} \langle \boldsymbol{g}_x, \boldsymbol{x} \rangle + D_X(\boldsymbol{x}_s, \boldsymbol{x}'_s),$$
$$\text{prox}_y(\boldsymbol{g}_y, \boldsymbol{y}'_s) = \arg\max_{\boldsymbol{y}_s \in Y} \langle \boldsymbol{g}_y, \boldsymbol{y}_s \rangle - D_Y(\boldsymbol{y}_s, \boldsymbol{y}'_s).$$

These two proximal mapping are computed once per iteration of the algorithm, with varying inputs. A crucial issue for a practical scalable method is therefore whether these proximal mappings can be computed efficiently. As we will show later, this is indeed the case for several types of distributional uncertainty that are of practical interest.

Primal-Dual update for MDP. In this paper we focus on the primal-dual FOM from Chambolle & Pock (2016), which we refer to as PDA. Given the saddle-point formulation of (5), for some step sizes $\tau, \sigma \in \mathbb{R}$ and some vector $\boldsymbol{v} \in \mathbb{R}^S$, the Primal-Dual Algorithm (PDA) repeatedly applies proximal mappings as follows:

$$\boldsymbol{x}_s^{t+1} = \text{prox}_x(\tau \boldsymbol{c}_s^{t\prime}, \boldsymbol{x}_s^t), \tag{9}$$
$$(\boldsymbol{y}_{i,s}^{t+1})_i = \text{prox}_y(\sigma \hat{\boldsymbol{h}}_s^t, (\boldsymbol{y}_{i,s}^t)_i) \tag{10}$$

where $\boldsymbol{c}_s^{t\prime} \in \mathbb{R}^A, c_{sa}^{t\prime} = c_{sa} + \lambda \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_{i,s,a}^{t\top} \boldsymbol{v}$, and $\hat{\boldsymbol{h}}_s^t \in \mathbb{R}^{N \times A \times S}, h_{ias'} = -\frac{\lambda}{N}(2x_{sa}^{t+1} - x_{sa}^t)v_{s'}$ for each i, a and s'. After T iterations, PDA obtains a $O(1/T)$ approximation to a (static) saddle-point problem such as $F(\boldsymbol{v})$ (Chambolle & Pock, 2016). Various weight schemes can be chosen to accelerate the ergodic convegence (Gao et al., 2019). We now show how to combine PDA updates with VI in order to compute a solution to (2).

Algorithm for DR-MDP. Our algorithm builds upon the first-order framework introduced in Grand-Clément & Kroer (2021) for robust MDP. In particular, the horizon T is divided into k *epochs* of lengths $1, ..., k^2$. During epoch ℓ, we perform ℓ^2 PDA *iterations*, starting from the last policy-kernel pair computed at the previous epoch. The average of the policy-kernel pairs visited across *all* epochs converges to an optimal solution of the distributionally robust MDP problem, as shown in Theorem 1. Our Algorithm 1 is different from the algorithm proposed in Grand-Clément & Kroer (2021) for *robust* MDP, which only optimizes for a single kernel. This is because we must iterate over an N-tuple of kernels $(\boldsymbol{y}_1, ..., \boldsymbol{y}_N)$ for the max-player. To better understand the distinction between the algorithms, note that one could apply the algorithm of (Grand-Clément & Kroer, 2021) directly to (5) since that formulation has a single $\boldsymbol{y}$. However, it is not clear how one would set up an appropriate strongly-convex function $\psi_{\mathbb{B}_{p,s}}$ for this space, as it suffers from degeneracy issues where the same average kernel $\boldsymbol{y}$ can be represented by multiple combinations of the samples $\boldsymbol{y}_1, ..., \boldsymbol{y}_N$. In contrast, we will show that there are efficient proximal setups for our representation in terms of $\tilde{\mathbb{B}}_{p,s}$. Our choice of step sizes τ and σ also specifically addresses the dimension imbalance between the min-player decisions $\boldsymbol{x} \in \mathbb{R}^A$ and the max-player decisions $(\boldsymbol{y}_i)_i \in \mathbb{R}^{NAS}$.

Algorithm 1 First-order Method for Wasserstein DR-MDP

```
1: Input A number of epochs $k$.
2: Initialize $\boldsymbol{v}^1, \bar{\boldsymbol{x}}^0, \bar{\boldsymbol{y}}^0$ at random
3: for epoch $\ell = 1, ..., k$ do
4:     for $s \in \mathbb{S}$ do
5:         $\tau = \left(\sqrt{A}\lambda\|\boldsymbol{v}^\ell\|_2\right)^{-1}, \sigma = N\sqrt{A}(\lambda\|\boldsymbol{v}^\ell\|_2)^{-1}$
6:         $\tau_\ell = \sum_{k'=1}^{(\ell-1)^2} k'$
7:         for $t = \tau_\ell, \ldots, \tau_\ell + T_\ell$ do
8:             $\boldsymbol{x}_s^{t+1} = \text{prox}_x(\tau \boldsymbol{c}_s^{t\prime}, \boldsymbol{x}_s^t)$
9:             $(\boldsymbol{y}_{i,s}^{t+1})_i = \text{prox}_y(\sigma \hat{\boldsymbol{h}}_s^t, (\boldsymbol{y}_{i,s}^t)_i)$
10:        $S_\ell = \sum_{t=\tau_\ell}^{\tau_\ell+\ell^2} t$
11:        $(\bar{\boldsymbol{x}}_s^\ell, (\bar{\boldsymbol{y}}_{i,s}^\ell)_i) = \sum_{t=(\ell+1)}^{\tau_\ell+\ell^2} \frac{t}{S_\ell}(\boldsymbol{x}_t, (\boldsymbol{y}_{t,i})_i)$
12:        Compute $\bar{\boldsymbol{y}}_s^\ell \in \mathbb{B}_s$ as $\bar{\boldsymbol{y}}_s^\ell = \frac{1}{N}\sum_{i=1}^{N} \bar{\boldsymbol{y}}_{i,s}^\ell$
13:        Update $v_s^{\ell+1} = F^{\bar{\boldsymbol{x}}_s^\ell, \bar{\boldsymbol{y}}_s^\ell}(\boldsymbol{v}^\ell)_s$
14: Let $S_T = \sum_{t=1}^{T} t$
15: Output $(\bar{\boldsymbol{x}}_s^T, (\bar{\boldsymbol{y}}_{i,s}^T)_i) = \sum_{t=1}^{T} \frac{t}{S_T}(\boldsymbol{x}_t, (\boldsymbol{y}_{t,i})_i)$
```

Algorithm 1 guarantees a bound on the *duality gap* of a policy-kernel pair $(\boldsymbol{x}, \boldsymbol{y})$ defined as

$$\max_{s \in \mathbb{S}} \{ \max_{\boldsymbol{y}' \in \mathbb{B}_s} F^{\boldsymbol{x}, \boldsymbol{y}'}(\boldsymbol{v}^*)_s - \min_{\boldsymbol{x}' \in \Delta(A)} F^{\boldsymbol{x}', \boldsymbol{y}}(\boldsymbol{v}^*)_s \}, \tag{11}$$

where $F^{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{v})_s = \sum_{a \in \mathbb{A}} x_{sa}\left(c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}\right)$. Note that (11) $\leq \epsilon/2$ guarantees that $\boldsymbol{x}$ is a ϵ-optimal policy in (1). We give a detailed proof of our theorem in Appendix B.

Theorem 1. *Let $\boldsymbol{v}^*$ be the value vector for a pair $\boldsymbol{x}^*, \boldsymbol{y}^*$ of optimal solutions to the Bellman equation* (1).

Let $\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T$ the output of Algorithm 1 after T iterations. The duality gap (11) *of $\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T$ is upper bounded by* $O\left(\frac{\sqrt{S}}{\sqrt{N}} R_X R_Y \left(\frac{\Theta_X}{\tau} + \frac{\Theta_Y}{\sigma}\right) \frac{1}{T^{2/3}}\right)$.

Therefore, Algorithm 1 returns a sequence of policies which converges to an optimal solution to the Distributionally Robust MDP over Wasserstein balls. In order to give the number of arithmetic operations for Algorithm 1 before returning an ϵ-optimal policy, there remains to investigate the complexity of the proximal updates (9)-(10).

Remark 2. We could use other FOMs than PDA in Algorithm 1. For example, Mirror Prox would yield a similar rate (Nemirovski, 2004), while Mirror Descent would yield a slower rate. It is also possible to change the proximal setup, e.g. to $\|\cdot\|_X = \|\cdot\|_1, \|\cdot\|_Y = \|\cdot\|_1$. For such a choice of norms, a natural choice of 1-convex function is the *negative entropy*, which leads to the Kullback-Leibler divergence as the Bregman divergence.

Remark 3. FOMs for constrained saddle-point problems can be accelerated (from $1/T$ to $1/T^2$) when the objective is *strongly* convex-concave (Theorem 4 in Section 5 of (Chambolle & Pock, 2016)). Additionally, if the objective is smooth, it is possible to achieve a linear convergence rate (Theorem 5 in Section 6 of (Chambolle & Pock, 2016)). In our setting the objective is a bilinear function (see (6)), and therefore we cannot use accelerated FOMs. Finally, the only known $1/T$ lower bounds for FOMs in stationary settings (Ouyang & Xu, 2021) are very technical and relate to ℓ_2-ball settings, and not on the simplex. It appears hard to extend these results to MDPs.

4. Convergence Rate for Wasserstein Balls.

Note that in Theorem 1, we only provide a convergence rates in term of the number of PD iterations T. In order to obtain our complexity results, we now turn to investigating the complexity of the primal-dual updates (9) and (10). The uncertainty set $\tilde{\mathbb{B}}_{p,s}$ is quite unusual in the first-order methods literature, where most of the updates are computed in closed-form upon the simplex or the non-negative orthant. One of the main contributions of this paper is to design novel efficient algorithms for computing (10) when the metric d is d_1, d_2 or d_∞. In particular in Proposition 4 we show that we can compute (10) in nearly linear time. To the best of our knowledge, we are the first to present efficient algorithms for computing the proximal updates on intersection of simplices and (various) Wasserstein balls.

Proximal setup for $\boldsymbol{x}$ player The proximal update for the $\boldsymbol{x}$ player (9) is the classical proximal update onto the simplex of dimension A, and can be computed in $O(A\log(A))$ operations (Ben-Tal & Nemirovski, 2001).

Proximal setup for $\boldsymbol{y}$ player Since (10) decomposes into independent problems for each state, we drop the index s in our formulation of (10) and assume that we are solving for some arbitrary state s. For $p < +\infty$, the proximal update of the max player (10) from a kernel $\boldsymbol{y'}$ can be reformulated as

$$\begin{aligned} \min \; & \sum_{i=1}^{N} \langle \boldsymbol{y}_i, \boldsymbol{h}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}'_i\|_2^2 \\ & \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in \mathcal{U}, \\ & \frac{1}{N}\sum_{i=1}^{N} d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)^p \le \theta^p. \end{aligned} \tag{12}$$

In the next propositions, we show that (12) can be solved efficiently, for d equal to d_1, d_2 and d_∞. The proof for each case is different, but follows a similar argument:

1. We first introduce a Lagrange multiplier γ for the last constraint. This simplifies the problem of computing (12) to solving N sub-problems over $\mathcal{U}$, each of the form
$$\begin{aligned} \min \; & \langle \boldsymbol{y}_i, \boldsymbol{h}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}'_i\|_2^2 + \gamma \cdot d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)^p \\ & \boldsymbol{y}_i \in \mathcal{U}. \end{aligned} \tag{13}$$
2. We then turn to efficiently solving (13).
   - For $d = d_2, p = 2$, (13) can be rewritten as a series of Euclidean projections onto the simplex $\Delta(S)$, as $\mathcal{U} = (\Delta(S))^A$.
   - For $d = d_1, p = 1$, we introduce Lagrange multipliers $\alpha_{i,s,a}$ for each simplex constraint $\boldsymbol{y}_{i,s,a}^\top \boldsymbol{e} = 1$; we can then solve the resulting problems using the KKT conditions. By carefully inspecting the breakpoints of the Lagrangian for the multipliers $\alpha_{i,s,a}$, we do not need to use bisection to find the multipliers $\alpha_{i,s,a}$; see Appendix C.
   - Finally, for $d = d_\infty, p = 1$, we use bisection to find an optimal α such that $d(\boldsymbol{y}_a, \hat{\boldsymbol{y}}_{i,a}) \le \alpha$, for all $a \in \mathbb{A}$. Then we solve the problem of Euclidean projection onto the simplex $\Delta(S)$ with box constraints.
3. Having designed efficient algorithms for solving (13), we use a bisection method on the multiplier μ and return an optimal solution of (12).

Summarizing the above ideas, we have the following proposition. We present the detailed proof in Appendix C.

Proposition 4. *Let $d = d_2, p = 2$ or $d = d_1, p = 1$. The proximal update* (12) *can be computed in $O\left(NAS\log(S)\log(\epsilon^{-1})\right)$ arithmetic operations.*

Let $d = d_{\infty}, p = 1$. The proximal update (12) can be computed in $O\left(NAS\log(S)\log^3(\epsilon^{-1})\right)$ arithmetic operations.

We can now give the overall convergence rates of our algorithms in the following theorem.

Theorem 5. *The total number of arithmetic operations needed to compute an ϵ-optimal solution to the Distributionally Robust MDP problem (1) using Algorithm 1 is $O\left(NA^{2.5}S^{3.5}\log(S)\log^m(\epsilon^{-1})\epsilon^{-1.5}\right)$, where $m = 1$ for $d = d_2$ and $p \in \{2, +\infty\}$, $d = d_1$ and $p \in \{1, +\infty\}$, and $m = 3$ for $d = d_{\infty}$ and $p \in \{1, +\infty\}$.*

Proof. We show here our proof for $d = d_2$ and $p \in \{2, +\infty\}$, and $d = d_1$ and $p \in \{1, +\infty\}$; the proof for $d = d_{\infty}$ and $p \in \{1, +\infty\}$, follows the same argument.

For our choice of $\|\cdot\|_X, \|\cdot\|_Y$, Bregman divergences and step sizes we have (see Ben-Tal & Nemirovski (2001))

- $R_X = O(1), R_Y = O(\sqrt{NA})$,
- $\Theta_X = O(1), \Theta_Y = O(NA)$,
- $\Theta_X/\tau = \Theta_Y/\sigma = \sqrt{A}\lambda\|\boldsymbol{v}^{\ell}\|_2 = O\left(\sqrt{AS}\right)$,

where we have used the norm equivalence between $\|\cdot\|_2$ and $\|\cdot\|_{\infty}$ in $\mathbb{R}^S$ in the last two lines. Therefore following Theorem 1 we have that the duality gap (11) of the policy returned by Algorithm 1 after T PD iterations is bounded above by $O\left(\frac{SA}{T^{2/3}}\right)$. Each PD iteration for these choices of d and p can be computed in $O\left(NAS\log(S)\log(\epsilon^{-1})\right)$. Note that we have to compute PD iterations for each state $s \in \mathbb{S}$; therefore, Algorithm 1 returns an ϵ-optimal policy to the Distributionally Robust MDP problem in $O\left(NA^{2.5}S^{3.5}\log(S)\log(\epsilon^{-1})\epsilon^{-1.5}\right)$. □

Comparing Algorithm 1 to Value Iteration, we improve upon the dependence on the problem size by a factor of $O(N^{2.5}AS)$, at the cost of a $\epsilon^{-1.5}$ convergence rate in terms of the accuracy ϵ. This is expected, as the advantage of FOMs is that they significantly improve upon the cost of the updates in terms of the dimensions of the problem. This is a well-known, standard tradeoff, and FOMs have proved extremely efficient in other settings than MDPs, e.g., poker AI and equilibrium computation (Kroer et al., 2018). Theoretically, we improve the convergence rate (compared to Value Iteration) by a factor $\Omega(N^{2.5}AS)$, which is large, even for small numbers of kernels N, states S and actions A. The improvement in terms of N is better than in terms of S and A because the number of kernels N only plays a role for the max-player; this is also the reason why we choose different step sizes τ and σ in Algorithm 1.

Remark 6 (Epoch and weight scheme)**.** The above results are for epoch lengths $T_{\ell} = \ell^2$. By choosing larger values $T_{\ell} = \ell^q$ where q tends to infinity, our algorithm approaches a complexity of $O\left(NA^2S^3\log(S)\log^m(\epsilon^{-1})\epsilon^{-1}\right)$. Thus it is possible to improve upon VI by a total factor of $O(N^{2.5}A^{1.5}S^{1.5})$ by choosing a large q. Additionally, we have presented Algorithm 1 with *linear* weights, i.e. the weight is t for the iterate $(\boldsymbol{x}^t, (\boldsymbol{y}_i^t)_i)$. Note that Algorithm 1 can be implemented with any (increasing) weight schemes; we found that for a weight scheme of $t^p, p \geq 0$, the convergence rate of Algorithm 1 *does not depend of* p, even though numerically, $p = 1$ performs better than $p = 0$.

5. Numerical Experiments

In this section we compare the empirical performances of our algorithm with state-of-the-art approaches. We focus on $d = d_2$ and we compare the running time of Algorithm 1 to the classical Value Iteration algorithm VI, Gauss-Seidel VI (*GS-VI*, Puterman (1994)), Anderson VI (*Anderson*, Geist & Scherrer (2018)), and Accelerated VI (*AVI*, Goyal & Grand-Clément (2019)) (see Appendix F for more details).

Empirical setup. We implement our algorithms in Python 3.7.3, using Gurobi 8.1.1 to solve any linear/quadratic optimization program involved. We run our simulations on a laptop with 2.2 GHz Intel Core i7 and 8 GB of RAM. We test our algorithm on three different sets of instances: a machine replacement problem, a forest management problem and some random (Garnet) instances. The discount factor is fixed at $\lambda = 0.8$. For each MDP instance, we generate the sampled kernels $\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_N$ by considering N small random (Garnet) perturbations around the "true" nominal kernel $\boldsymbol{y}^0$ (see Appendix F).

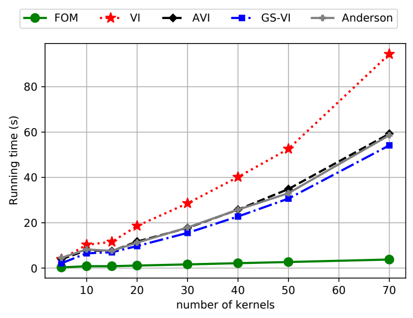
All figures in this section show the running times of the algorithms before returning an ϵ-optimal policy with $\epsilon = 0.1$. We stop Algorithm 1 when (DG) $\leq \epsilon/2$, where

$$\max_{\mu \in \mathbb{D}} C(\boldsymbol{x}, \mu) - \min_{\boldsymbol{x'} \in \Pi} C(\boldsymbol{x'}, \mu) \qquad \text{(DG)}$$

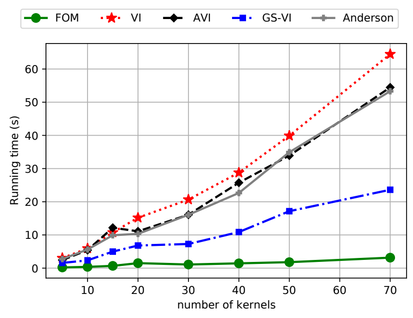
is the *duality gap* of a pair of policy-density (π, μ). Note that (DG) $\leq \epsilon/2$ is enough to ensure that the policy is an ϵ-optimal policy for (1). We stop VI and variants when $\|\boldsymbol{v}^{\ell} - F(\boldsymbol{v}^{\ell})\|_{\infty} < 2\lambda\epsilon(1-\lambda)^{-1}$, which guarantees that the current policy is ϵ-optimal (Puterman, 1994). The running times are averaged across 5 instances by changing the seeds for sampling the N kernels around $\boldsymbol{y}^0$.

Initialization and warm-start. We initialize all algorithms with $\boldsymbol{v}_0 = \boldsymbol{0}$. We evaluate $F(\boldsymbol{v})$ using our convex reformulation (see Appendix A). At epoch ℓ of VI and variants, we warm-start each computation of $F(\boldsymbol{v}^{\ell})$ with the optimal solution obtained from the previous epoch $\ell - 1$. We present details about the computation of (DG) in Appendix E.

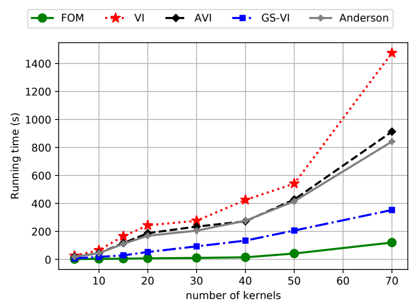
Structured MDP instances. We consider two instances in-



(a) Forest.

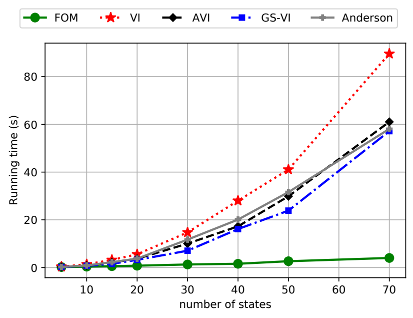


(b) Machine.

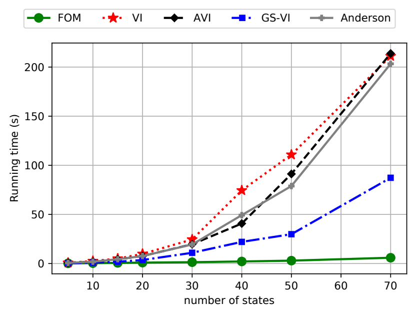


(c) Garnet.

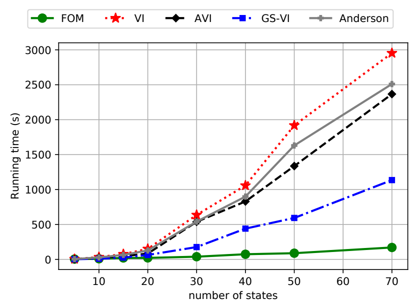
Figure 1: Comparison of Alg. 1 with four variants of Value Iteration on three MDP domains (increasing number of kernels).



(a) Forest.



(b) Machine.



(c) Garnet.

Figure 2: Comparison of Alg. 1 with four variants of Value Iteration on three MDP domains (increasing number of states).

spired from real-world applications, a machine replacement problem studied by Delage & Mannor (2010), Wiesemann et al. (2013) and Goyal & Grand-Clément (2018), and a forest management example from the Python package *pymdptoolbox* (Cordwell et al., 2015) inspired by Possingham & Tuck (1997). In the machine replacement problem, the goal is to design a replacement policy for a line of machines. The states of the MDP represent age phases of the machine and the actions represent different repair or replacement options. In the forest management problem, the forest grows at every period and the goal is to balance the revenue associated with selling cut wood and the risk of wildfire. The transition kernels $\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_N$ represent historical data, obtained from observations from previous years. In both instances, even though the transition parameters can be estimated from retrospective data sets, one often does not have access to enough data to exactly assess the probability of a machine breaking down when in a given condition, the rate of growth of the forest, or the risk of wildfire. Additionally, the historical data may contain errors; this warrants the use of a robust model for finding good, stable machine replacement and forest management policies. We present details on these instances in Appendix G and Appendix H.

Random MDP instances. We also test our algorithm on random, denser MDP instances. We use the Generalized Average Reward Non-stationary Environment Test-bench, or in short, Garnet MDPs (Archibald et al., 1995; Bhatnagar et al., 2007). Garnet MDPs are a class of abstract but representative finite MDPs that are easy to build and for which we can control the connectivity of the underlying Markov chain with a *branching* factor, n_b, which represents the proportion of next states available at every state-action pair (s, a). They are a class of randomly constructed finite MDP's serving as a test-bench for RL algorithms (Tarbouriech & Lazaric, 2019; Piot et al., 2016; Jian et al., 2019). We consider $S = A$, $n_b = 20\%$ and random uniform rewards in $[0, 10]$.

Increasing instance sizes. Our experiments evaluate the performance of all algorithms by running them on increasingly-larger instances. Our problems have three size parameters: S and A, which affect the MDP size, and N, which affects the size of the ambiguity sets. Because the runtimes of the VI algorithms grow quickly in these parameters, we perform our experiments by holding two out of three parameters fixed, while increasing the last one. When we consider an increasing number of kernels (Figure 1), we keep $S = 30$ fixed. When we consider an increasing number of states, we keep $N = 30$ fixed. For all instances, $A = 30$ for Garnet MDPs, $A = 2$ for machine replacement MDPs, and $A = 3$ for forest management MDPs.

Numerical results. We present the results of our numerical study in Figure 1 and Figure 2. For very small instances (e.g. $S = 5$ states, $A = 2$ actions, $N = 30$ observed kernels), Algorithm 1 has similar performance as the other four algorithms. When the number of states or the number of kernels increases, the average convergence times of our algorithm moderately increase, e.g. from 1.6 seconds for $N = 5, S, A = 30$ to 120.2 seconds for $N = 70, S, A = 30$ (Figure 1c). However, Algorithm 1 scales significantly better than the other methods based on IPM, and as the instance sizes increases it outperforms all other methods. We also see that for Garnet instances, the convergences of all the algorithms are slower, since the MDP instances are denser than for more structured examples and there are more actions. As expected from our theoretical results in the previous section, the running time of Algorithm 1 grows linearly with N. Perhaps more surprisingly, the empirical running times of the other algorithms also seem to grow (almost) linearly with N. This may be due to the solver (Gurobi 8.1.1) exploiting the particular problem structure of the robust Bellman update (see Appendix A).

References

Akian, M., Gaubert, S., Qu, Z., and Saadi, O. Multiply accelerated value iteration for non-symmetric affine fixed point problems and application to Markov decision processes. *arXiv preprint arXiv:2009.10427*, 2020.

Archibald, T., McKinnon, K., and Thomas, L. On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46(3):354–361, 1995.

Ben-Tal, A. and Nemirovski, A. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. Siam, 2001.

Bertsimas, D., Shtern, S., and Sturt, B. A data-driven approach for multi-stage linear optimization. *Available at Optimization Online*, 2018.

Bertsimas, D., Shtern, S., and Sturt, B. Two-stage sample robust optimization. *arXiv preprint arXiv:1907.07142*, 2019.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. Natural gradient actor-critic algorithms. *Automatica*, 2007.

Chambolle, A. and Pock, T. On the ergodic convergence rates of a first-order primal–dual algorithm. *Mathematical Programming*, 159(1-2):253–287, 2016.

Chen, Z., Yu, P., and Haskell, W. B. Distributionally robust optimization for sequential decision-making. *Optimization*, 68(12):2397–2426, 2019.

Cordwell, S., Gonzalez, Y., and Tulabandhula, T. Markov Decision Process (MDP) toolbox for python. *https://github.com/sawcordwell/pymdptoolbox*, 2015.

De Farias, D. P. and Van Roy, B. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

Delage, E. and Mannor, S. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research*, 58(1):203 – 213, 2010.

Esfahani, P. M. and Kuhn, D. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1-2):115–166, 2018.

Gao, R. and Kleywegt, A. J. Distributionally robust stochastic optimization with Wasserstein distance. *arXiv preprint arXiv:1604.02199*, 2016.

Gao, Y., Kroer, C., and Goldfarb, D. Increasing iterate averaging for solving saddle-point problems. *arXiv preprint arXiv:1903.10646*, 2019.

Geist, M. and Scherrer, B. Anderson acceleration for reinforcement learning. *arXiv preprint arXiv:1809.09501*, 2018.

Givens, C. R., Shortt, R. M., et al. A class of Wasserstein metrics for probability distributions. *The Michigan Mathematical Journal*, 31(2):231–240, 1984.

Gong, H. and Wang, M. A duality approach for regret minimization in average-award ergodic Markov decision processes. 2020.

Goyal, V. and Grand-Clément, J. Robust Markov decision process: Beyond rectangularity. *arXiv preprint arXiv:1811.00215*, 2018.

Goyal, V. and Grand-Clément, J. A first-order approach to accelerated value iteration. *arXiv preprint arXiv:1905.09963*, 2019.

Grand-Clément, J. and Kroer, C. Scalable first-order methods for robust MDPs. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Grand-Clément, J., Chan, C. W., Goyal, V., and Escobar, G. Robust policies for proactive ICU transfers. *arXiv preprint arXiv:2002.06247*, 2020.

Ho, C., Petrik, M., and Wiesemann, W. Fast Bellman updates for Robust MDPs. *Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm*, 2018.

Iyengar, G. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Jian, Q., Fruit, R., Pirotta, M., and Lazaric, A. Exploration bonus for regret minimization in discrete and continuous average reward MDPs. In *Advances in Neural Information Processing Systems*, pp. 4890–4899, 2019.

Jin, Y. and Sidford, A. Efficiently solving mdps with stochastic mirror descent. In *International Conference on Machine Learning*, pp. 4890–4900. PMLR, 2020.

Kroer, C., Waugh, K., Kılınç-Karzan, F., and Sandholm, T. Faster algorithms for extensive-form game solving via improved smoothing functions. *Mathematical Programming*, pp. 1–33, 2018.

Miao, F., Han, S., Hendawi, A. M., Khalefa, M. E., Stankovic, J. A., and Pappas, G. J. Data-driven distributionally robust vehicle balancing using dynamic region partitions. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pp. 261–271, 2017.

Nemirovski, A. Prox-method with rate of convergence O(1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

Nesterov, Y. A method for solving the convex programming problem with convergence rate O(1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Nilim, A. and Ghaoui, L. E. Robust control of Markov decision processes with uncertain transition probabilities. *Operations Research*, 53(5):780–798, 2005.

Ouyang, Y. and Xu, Y. Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems. *Mathematical Programming*, 185(1):1–35, 2021.

Petrik, M. Optimization-based approximate dynamic programming. 2010.

Piot, B., Geist, M., and Pietquin, O. Difference of convex functions programming applied to control with expert data. *arXiv preprint arXiv:1606.01128*, 2016.

Possingham, H. and Tuck, G. Application of stochastic dynamic programming to optimal fire management of a spatially structured threatened species. In *Proceedings International Congress on Modelling and Simulation, MODSIM*, pp. 813–817, 1997.

Puterman, M. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

Steimle, L. N., Kaufman, D. L., and Denton, B. T. Multi-model Markov decision processes. *Optimization Online URL http://www. optimization-online. org/DB_FILE/2018/01/6434. pdf*, 2018.

Tamar, A., Mannor, S., and Xu, H. Scaling up robust mdps using function approximation. In *International Conference on Machine Learning*, pp. 181–189. PMLR, 2014.

Tarbouriech, J. and Lazaric, A. Active exploration in Markov decision processes. *arXiv preprint arXiv:1902.11199*, 2019.

Tsitsiklis, J. N. and Van Roy, B. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pp. 1075–1081, 1997.

Wiesemann, W., Kuhn, D., and Rustem, B. Robust Markov decision processes. *Operations Research*, 38(1):153–183, 2013.

Wiesemann, W., Kuhn, D., and Sim, M. Distributionally robust convex optimization. *Operations Research*, 62(6): 1358–1376, 2014.

Xie, W. Tractable reformulations of two-stage distributionally robust linear programs over the type-infinity Wasserstein ball. *Operations Research Letters*, 2020.

Xie, W., Zhang, J., and Ahmed, S. Distributionally robust bottleneck combinatorial problems: Uncertainty quantification and robust decision making. *arXiv preprint arXiv:2003.00630*, 2020.

Xu, H. and Mannor, S. Distributionally robust Markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 2505–2513, 2010.

Yang, I. A convex optimization approach to distributionally robust Markov decision processes with wasserstein distance. *IEEE control systems letters*, 1(1):164–169, 2017.

Yu, P. and Xu, H. Distributionally robust counterpart in Markov decision processes. *IEEE Transactions on Automatic Control*, 61(9):2538–2543, 2015.

Zhang, J., O'Donoghue, B., and Boyd, S. Globally convergent type-I Anderson acceleration for non-smooth fixed-point iterations. *arXiv preprint arXiv:1808.03971*, 2018.

Zhao, C. and Guan, Y. Data-driven risk-averse stochastic optimization with Wasserstein metric. *Operations Research Letters*, 46(2):262–267, 2018.

A. Convex Reformulation for Bellman Update

We show here how to reformulate (3) into a convex program, for $\mathbb{B}_s = \mathbb{B}_{p,s}$ (the reformulation for $\mathbb{B}_s = \mathbb{B}_{\infty,s}$ follows directly). At every epoch of Value Iteration VI, we compute $F(\boldsymbol{v})$ for the current value vector $\boldsymbol{v} \in \mathbb{R}^S$, where

$$F(\boldsymbol{v})_s = \min_{\boldsymbol{x}_s \in \Delta(A)} \max_{\boldsymbol{y}_s \in \mathbb{B}_s} \sum_{a=1}^{A} x_{sa} \left(c_{sa} + \lambda \cdot \boldsymbol{y}_{sa}^\top \boldsymbol{v} \right), \forall\, s \in \mathbb{S}.$$

From convex duality we have, for any $s \in \mathbb{S}$,

$$\begin{aligned} F(\boldsymbol{v})_s &= \min_{\boldsymbol{x}_s \in \Delta(A)} \max_{\boldsymbol{y}_s \in \mathbb{B}_s} \sum_{a=1}^{A} x_{sa} \left(c_{sa} + \lambda \cdot \boldsymbol{y}_{sa}^\top \boldsymbol{v} \right) \\ &= \max_{\boldsymbol{y}_s \in \mathbb{B}_s} \min_{\boldsymbol{x}_s \in \Delta(A)} \sum_{a=1}^{A} x_{sa} \left(c_{sa} + \lambda \cdot \boldsymbol{y}_{sa}^\top \boldsymbol{v} \right). \quad (14) \end{aligned}$$

For $\boldsymbol{y} \in \mathbb{B}_s$, we can reformulate the inner minimization as

$$\begin{aligned} \max\ & \gamma \\ & \gamma \in \mathbb{R}, \\ & c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v} \geq \gamma, \forall\, a \in \mathbb{A}. \end{aligned}$$

Overall, we have proved that

$$\begin{aligned} F(\boldsymbol{v})_s = \max\ & \gamma \\ & \gamma \in \mathbb{R}, \boldsymbol{y} \in \mathbb{B}_s, \quad (15) \\ & c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v} \geq \gamma, \forall\, a \in \mathbb{A}. \end{aligned}$$

Replacing $\mathbb{B}_s$ by $\tilde{\mathbb{B}}_{p,s}$ we obtain

$$\begin{aligned} F(\boldsymbol{v})_s = \max\ & \gamma \\ & \gamma \in \mathbb{R}, \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in \mathcal{U}, \\ & c_{sa} + \lambda \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_{i,sa}^\top \boldsymbol{v} \geq \gamma \forall\, a \in \mathbb{A}, \quad (16) \\ & \frac{1}{N} \sum_{i=1}^{N} d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_{i,s})^p \leq \theta^p. \end{aligned}$$

Formulation (16) is a linear program with linear constraints (for $d = d_1, d_\infty$ and $p = 1$), and one additional quadratic constraint (for $d = d_2$ and $p = 2$). Following (Ben-Tal & Nemirovski, 2001), we can solve (16) up to accuracy ϵ in a number of arithmetic operations in $O\left(N^{3.5} S^{3.5} A^{3.5} \log(1/\epsilon)\right)$. We warm-start each of this optimization problem with the optimal solution found in the previous epoch of VI.

B. Proof of Theorem 1

We present here the detailed proof for Theorem 1. We proceed in three steps:

- We justify the choice of the step-sizes σ, τ as

$$\tau = \left(\sqrt{A} \lambda \|\boldsymbol{v}^\ell\|_2 \right)^{-1}, \sigma = N\sqrt{A}(\lambda \|\boldsymbol{v}^\ell\|_2)^{-1}.$$

- We prove upper bounds on the duality gap (11).
- We finally combine these upper bounds to obtain the convergence rate of Theorem 1.

Choice of step-sizes. We define

$$L = \sup_{\|\boldsymbol{x}\|_2 \leq 1, \|(\boldsymbol{y})_i\|_2 \leq 1} \sum_{a \in \mathbb{A}} \boldsymbol{x}_{sa} \lambda \sum_{i=1}^{N} \frac{1}{N} \boldsymbol{y}_{i,sa}^\top \boldsymbol{v}^\ell.$$

At epoch ℓ we choose step sizes σ, τ such that

$$\frac{1}{\sqrt{\sigma \tau}} = L. \quad (17)$$

From Chambolle & Pock (2016), this is enough to ensure that $\bar{\boldsymbol{x}}_s^\ell, (\bar{\boldsymbol{y}}_{i,s}^\ell)_i$ are $O(1/\ell^2)$-optimal in computing $F(\boldsymbol{v}^\ell)$, where $\bar{\boldsymbol{x}}_s^\ell, (\bar{\boldsymbol{y}}_{i,s}^\ell)_i$ are the weighted averages for the iterates

$$(\boldsymbol{x}_{\tau_\ell+1}, (\boldsymbol{y}_{\tau_\ell+1,i})_i), ..., (\boldsymbol{x}_{\tau_\ell+\ell^2}, (\boldsymbol{y}_{\tau_\ell+\ell^2,i})_i),$$

with weights $\tau_\ell + 1, ..., \tau_\ell + \ell^2$. Now note that, by using Cauchy-Schwarz twice, we have

$$L = \frac{\lambda}{\sqrt{N}} \|\boldsymbol{v}^\ell\|_2. \quad (18)$$

Note that we could simply choose $\sigma = \tau = \sqrt{N}\left(\lambda \|\boldsymbol{v}^\ell\|_2\right)^{-1}$. However, since our convergence rate will involve the term $\Theta_X/\tau + \Theta_Y/\sigma$, we try to equalize these two terms. Under the condition (17), the best choice of step sizes is therefore $\tau = \left(\sqrt{\Theta_X/\Theta_Y}\right) L^{-1}$. Recall that Θ_X, Θ_Y are the maximum of the respective Bregman divergences (squared norm two) onto $\Delta(A)$ and $\tilde{B}_{p,s}$. Therefore, $\Theta_X = O(1), \Theta_Y = O(NA)$.This leads to

$$\tau = \left(\sqrt{A} \lambda \|\boldsymbol{v}^\ell\|_2 \right)^{-1}, \sigma = N\sqrt{A}(\lambda \|\boldsymbol{v}^\ell\|_2)^{-1}.$$

Note that we are essentially adjusting the step sizes, taking into account the difference of dimensions between $\Delta(A)$, the decision space of the min-player, and $\tilde{B}_{p,s} \subset \mathbb{R}^{N \times A \times S}$, the decision space of the max-player.

Upper bounds on duality gap (11) Note that Theorem 3.1 in Grand-Clément & Kroer (2021) only gives an upper bound on (11) when $N = 1$, which reduces to the case of robust MDP. However, note that we can extend this result to distributionally robust MDPs by considering that Algorithm 1 is running N instances of the same algorithm for robust MDPs, one instance per kernel $\boldsymbol{y}_i$. Here it is crucial to reckon that:

- This scales the constants R_Y (maximum of $\|\cdot\|_Y$ on Y) and Θ_Y (maximum of D_Y on $Y \times Y$). This is because for distributionally robust MDPs with nominal distribution supported on N kernels, Y is now contained in $(\Delta(S))^{N \times A}$, compared to Y contained in $(\Delta(S))^A$ for robust MDPs; here recall that we denote by Y the decision space of the max-player.
- This leaves unchanged the convergence rate of Algorithm 1 *in terms of number of PD iterations* T, as this convergence rate only depends (in terms of transition kernels) of the *expected value* $\boldsymbol{y}_s = \frac{1}{N}\sum_{i=1}^N \boldsymbol{y}_{i,s}$.

Therefore, after T PD iterations of Algorithm 1, the duality gap (11) is upper bounded by

$$O\left(R_X R_Y \left(\frac{\Theta_X}{\tau} + \frac{\Theta_Y}{\sigma}\right)\frac{\sqrt{S}}{\sqrt{N}}\left(\frac{\lambda^{T^{1/3}}}{T^{1/3}} + \frac{1}{T^{2/3}}\right)\right).$$

Note the additional $1/\sqrt{N}$, compared to Theorem 3.1 from Grand-Clément & Kroer (2021); this comes from the equality (18). Let us now simplify this upper bound. Note that

$$\frac{\lambda^{T^{1/3}}}{T^{1/3}} + \frac{1}{T^{2/3}} = O\left(\frac{1}{T^{2/3}}\right),$$

because of the exponential decay of the term $\lambda^{T^{1/3}}$. Combining the two previous simplifications, we obtain that after T PD iterations, the duality gap (11) is upper bounded by

$$O\left(R_X R_Y \left(\frac{\Theta_X}{\tau} + \frac{\Theta_Y}{\sigma}\right)\frac{\sqrt{S}}{\sqrt{N}}\frac{1}{T^{2/3}}\right).$$

C. Proof Proposition 4

In this section we focus on solving (12), dropping the index $s \in \mathbb{S}$, with the understanding that $\boldsymbol{h} = \boldsymbol{h}_s \in \mathbb{R}^{A \times S}$, $\hat{\boldsymbol{y}}_i = \hat{\boldsymbol{y}}_{i,s} \in \mathcal{U}$.

Proof for $d = d_2, p = 2$. The proximal update becomes

$$\begin{aligned} \min & \sum_{i=1}^N \langle \boldsymbol{y}_i, \boldsymbol{h}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}_i'\|_2^2 \\ & \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in (\Delta(S))^A, \\ & \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_2^2 \leq \theta^2. \end{aligned}$$

If we dualize the second constraint with a Lagrange multiplier γ, we end up with computing NA Euclidean projections onto the simplex $\Delta(S)$, because the argmin of

$$\boldsymbol{y} \in \mathcal{U} \mapsto \langle \boldsymbol{y}, \boldsymbol{h}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{y} - \hat{\boldsymbol{y}}_i\|_2^2$$

is the same as the argmin of

$$\boldsymbol{y} \in \mathcal{U} \mapsto \frac{1}{2}\|\boldsymbol{y} - \frac{\sigma}{1+\sigma\gamma}\left(\frac{1}{\sigma}\boldsymbol{y'} + \gamma\hat{\boldsymbol{y}}_i - \boldsymbol{h}\right)\|_2^2.$$

We therefore compute NA Euclidean projections onto the simplex of size S, which can be performed in $O\left(NAS\log(S)\right)$ arithmetic operations. We then need to binary search over the Lagrange multiplier γ, resulting in a complexity $O\left(NAS\log(S)\log(\epsilon^{-1}\right)$.

Proof for $d = d_1, p = 1$. The proximal update becomes

$$\begin{aligned} \min & \sum_{i=1}^N \langle \boldsymbol{y}_i, \boldsymbol{h}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}_i'\|_2^2 \\ & \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in (\Delta(S))^A, \\ & \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_1 \leq \theta. \end{aligned}$$

We introduce a Lagrange multiplier $\gamma \geq 0$ for the second constraint: we now solve

$$\begin{aligned} & \max_{\gamma \geq 0} -\gamma\theta \\ & \quad + \min \sum_{i=1}^N \langle \boldsymbol{y}_i, \boldsymbol{h}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}_i'\|_2^2 + \gamma\|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_1 \\ & \quad\quad \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in (\Delta(S))^A. \end{aligned}$$

We then introduce Lagrange multipliers $(\alpha_{i,a})_{i,a}$ for each constraint $\sum_{s'=1}^S y_{i,a,s'} = 1$ for each $i = 1, ..., N$ and $a \in \mathbb{A}$:

$$\begin{aligned} & \max_{\gamma \geq 0} \max_{(\alpha_{i,a})_{i,a} \in \mathbb{R}^{N \times A}} -\sum_{i,a}\alpha_{i,a} - \gamma\theta \\ & + \sum_{i=1}^N \sum_{a=1}^A \sum_{s'=1}^S \min_{y_{i,a,s'} \geq 0} (h_{i,a,s'} + \alpha_{i,a}) y_{i,a,s'} \\ & + \frac{1}{2\sigma}(y_{i,a,s'} - y'_{i,a,s'})^2 + \gamma|y_{i,a,s'} - \hat{y}_{i,a,s'}|. \end{aligned}$$

Solving the inner minimization. Let us drop the index (i, a, s') and explain how to compute a closed-form solution to the inner univariate minimization:

$$\min_{y \geq 0} (h + \alpha)y + \frac{1}{2\sigma}(y - y')^2 + \gamma|y - \hat{y}|.$$

We can distinguish three regions.

1. $y > \hat{y}$. The first-order conditions yield
$$(h + \alpha) + (1/\sigma)(y - y') + \gamma = 0,$$
which implies $y = y' - \sigma(\gamma + h + \alpha)$. This is valid as long as $y' - \sigma(\gamma + h + \alpha) > \hat{y}$. Note that $y' - \sigma(\gamma + h + \alpha) > \hat{y}$ implies $y' - \sigma(\gamma + h + \alpha) \geq 0$, since $\hat{y} \geq 0$.

2. $y < \hat{y}$. The first-order conditions yield $y = y' - \sigma(-\gamma + h + \alpha)$, which is valid as long as $y' - \sigma(-\gamma + h + \alpha) < \hat{y}$ and $y' - \sigma(-\gamma + h + \alpha) \geq 0$.

Overall, we have

$$y = \begin{cases} y' - \sigma(\gamma + h + \alpha) & \text{if } \frac{1}{\sigma}(y' - \hat{y}) - h - \alpha > \gamma, \\ \hat{y} & \text{if } |\frac{1}{\sigma}(y' - \hat{y}) - h - \alpha| \leq \gamma, \\ (y' - \sigma(-\gamma + h + \alpha))^+ & \text{if } \frac{1}{\sigma}(y' - \hat{y}) - h - \alpha < -\gamma. \end{cases} \tag{19}$$

Note that this is essentially the shrinkage-thresholding operator, up to the last case and the $x \mapsto x^+$ function (which stems from the non-negativity constraint).

Solving the maximization over α. For a fixed Lagrange multiplier γ, our goal is now to solve

$$\max_{\alpha \in \mathbb{R}} -\alpha + \sum_{s'=1}^{S} (h_{s'} + \alpha) y_{s'} + \frac{1}{2\sigma}(y_{s'} - y'_{s'}) + \gamma |y_{s'} - \hat{y}_{s'}|, \tag{20}$$

where y follows (19). Let us rewrite (19) with the index s' and split the thresholding at zero into two cases:

$$y = \begin{cases} y' - \sigma(\gamma + h + \alpha) & \text{if } (1/\sigma)(y' - \hat{y}) - h - \alpha > \gamma, \\ \hat{y} & \text{if } |(1/\sigma)(y' - \hat{y}) - h - \alpha| \leq \gamma, \\ (y' + \gamma - h - \alpha)^+ & \text{if } (1/\sigma)(y' - \hat{y}) - h - \alpha < -\gamma, \\ 0 & \text{if } (1/\sigma)y'_{s'} - h_{s'} - \alpha < -\gamma. \end{cases}$$

For each $s' \in \mathbb{S}$ there are three breakpoints where the behavior of $y_{s'}$ changes with respect to the choice of α:

1. $(1/\sigma)y'_{s'} - h_{s'} - \alpha = -\gamma$: $y_{s'}$ becomes nonzero at a rate of $-\sigma\alpha$,

2. $(1/\sigma)(y' - \hat{y}) - h - \alpha = -\gamma$: $y_{s'}$ becomes constant at $\hat{y}_{s'}$,

3. $y(1/\sigma)(y' - \hat{y}) - h - \alpha = \gamma$: $y_{s'}$ grows above $\hat{y}_{s'}$ at a rate $-\sigma\alpha$.

This yields the following algorithm.

1. We sort the breakpoints in decreasing order of α, which takes time $O(S\log(S))$.

2. At the first breakpoint, $y_{s'} = 0$ for all s'.

3. We keep a counter `num_active` denoting how many variables change with α at the current breakpoint, initialized at zero.

4. We keep a counter `sum` denoting the value of $\sum_{s'} y_{s'}$ if we had set α equal to the current breakpoint, initialized at zero.

5. We then iterate through the breakpoints (in decreasing order). Let α_1, α_2 be the previous and current breakpoints. At every breakpoint:

   (a) set `sum`$+ = \sigma$`num_active`$\cdot (\alpha_2 - \alpha_1)$.

   (b) if `sum` > 1 then stop and go to 6.

   (c) else, we update `num_active` based on whether the current variable starts or stops changing at α_2, and go to the next breakpoint.

6. From the mean value theorem, an optimal α^* belongs to the interval $[\alpha_1, \alpha_2]$. We find it by setting $\alpha = \alpha_2 - ($`sum`$- 1)/(\sigma$`num_active`$)$.

There are NA Lagrange multipliers $(\alpha_{ia})_{i,a}$, and we can compute each of them in $O(S\log(S))$, given a Lagrange multiplier γ. We still need to use bisection to compute γ^*. Overall we end up with a complexity of $O(NA^2S^3\log(\epsilon^{-1})\epsilon^{-1})$.

Remark 7. In the context of robust MDP (i.e. N=1), note that Ho et al. (2018) gives an algorithm with complexity $O(S^2A\log(S^2A))$ to compute (3) with $d = d_1, p = 1$. It remains unclear to us if this algorithm extends to the case $N \geq 2$ and its complexity in this case.

Proof for $d = d_\infty, p = 1$. The FOM update becomes

$$\begin{aligned} \min \ & \frac{1}{N}\sum_{i=1}^{N} \langle \boldsymbol{y}_i, \boldsymbol{h} \rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}'_i\|_2^2 \\ & \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in \mathcal{U}, \\ & \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_\infty \leq \theta. \end{aligned}$$

We introduce a Lagrange multiplier $\gamma \in \mathbb{R}$ for the binding constraint, and our goal is now to solve

$$\begin{aligned} \min \ & \sum_{i=1}^{N} \langle \boldsymbol{y}_i, \boldsymbol{h} \rangle + \frac{1}{2\sigma}\|\boldsymbol{y}_i - \boldsymbol{y}'_i\|_2^2 + \gamma \cdot \sum_{i=1}^{N} \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_\infty \\ & \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in \mathcal{U}. \end{aligned}$$

Note that this problem decomposes across $i = 1, ..., N$, so that we can solve independently, for each i,

$$\begin{aligned} \min \ & \langle \boldsymbol{y}, \boldsymbol{h} \rangle + \frac{1}{2\sigma}\|\boldsymbol{y} - \boldsymbol{y}'_i\|_2^2 + \gamma\|\boldsymbol{y} - \hat{\boldsymbol{y}}_i\|_\infty \\ & \boldsymbol{y} \in \mathcal{U}. \end{aligned} \tag{21}$$

To solve (21), we can use bisection to find a feasible α such that $\gamma\|\boldsymbol{y} - \hat{\boldsymbol{y}}_i\|_\infty \leq \alpha$. This leads to solve

$$\begin{aligned} \min \ & \langle \boldsymbol{y}, \boldsymbol{h} \rangle + \frac{1}{2\sigma}\|\boldsymbol{y} - \boldsymbol{y}'_i\|_2^2 \\ & \boldsymbol{y} \in (\Delta(S))^A, \\ & \gamma\|\boldsymbol{y}_a - \hat{\boldsymbol{y}}_{i,a}\|_\infty \leq \alpha, \forall\, a \in \mathbb{A}. \end{aligned}$$

Note that this problem decomposes across each action a, so that we only have to solve A problems of the form

$$\begin{aligned} \min\ & \langle \boldsymbol{y}_{i,a}, \boldsymbol{h}_{ia} \rangle + \frac{1}{2\sigma} \| \boldsymbol{y}_{i,a} - \boldsymbol{y}'_{i,a} \|_2^2 \\ & \boldsymbol{y}_{i,a} \in \Delta(S), \\ & \gamma \| \boldsymbol{y}_{i,a} - \hat{\boldsymbol{y}}_{i,a} \|_\infty \leq \alpha. \end{aligned}$$

This brings down to solving the problem of Euclidean projection onto the simplex $\Delta(S)$ with box constraints, which can be done in $O(S \log(S) \log(\epsilon^{-1})$ (by relaxing the constraint $\boldsymbol{y}_{i,a}^\top \boldsymbol{e} = 1$). Then the overall complexity to compute an ϵ-approximation of the proximal update is in $O\left(NAS \log(S) \log^3\left(\epsilon^{-1}\right)\right)$.

D. Complexity Results for Type-∞ Wasserstein Ball

Background on type-∞ Wasserstein distance Xie (2020), Bertsimas et al. (2019), Bertsimas et al. (2018) consider ambiguity sets based on type-∞ Wasserstein distance with application to two-state distributionally robust optimization. Recent work suggests that distributionally robust optimization based on type-∞ distance has some computational advantages compared to DRO based on type-p Wasserstein distance (Xie et al., 2020).

Optimality of Markovian policy Note that Yang (2017) proves that for type p Wasserstein distance (with $p < +\infty$), an optimal policy can be found Markovian. We prove here that the same holds for Wasserstein distance of $p = +\infty$. Let us define the *value* vector for each state s as

$$v_s = \min_{\boldsymbol{x} \in \Delta(A)} \max_{\mu_s \in \mathbb{D}_s} \mathbb{E}_\pi \mathbb{E}_{\boldsymbol{y} \sim \mu_s} [\sum_{t=0}^{+\infty} \lambda^t c_{s_t a_t} |\ s_0 = s],$$

which represents the expected reward-to-go starting from a state s. Note that $s \mapsto v_s$ is well-defined because of the s-rectangularity assumption (Wiesemann et al., 2013). The Bellman equation (2) follows from the dynamic programming principle. Now we have that

$$\boldsymbol{x} \mapsto \max_{\mu_s \in \mathbb{D}_s} \mathbb{E}_{\boldsymbol{y}_s \sim \mu_s} \left[\sum_{a \in \mathbb{A}} x_{sa} \left(r_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}^* \right) \ |s_0 = s \right]$$

is convex (as the pointwise maximum of linear functions), proper (because the costs are bounded), and upper semi-continuous. Hence the minimization problem over $\boldsymbol{x} \in \Delta(A)$ is minimizing a closed proper convex function onto the closed convex set $\Delta(A)$. Therefore an optimal solution exists, i.e. there exists an optimal Markovian policy.

Proximal update. The proximal update on the max-player becomes

$$\begin{aligned} \min\ & \sum_{i=1}^N \langle \boldsymbol{y}_i, \boldsymbol{h} \rangle + \frac{1}{2\sigma} \| \boldsymbol{y}_i - \boldsymbol{y}'_i \|_2^2 \\ & \boldsymbol{y}_1, ..., \boldsymbol{y}_N \in \mathcal{U}, \\ & d(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) \leq \theta, \forall\ i = 1, ..., N. \end{aligned} \tag{22}$$

We note that this problem naturally decomposes along $i = 1, ..., N$, so that we only have to solve N subproblems of the form

$$\begin{aligned} \min\ & \langle \boldsymbol{y}, \boldsymbol{h} \rangle + \frac{1}{2\sigma} \| \boldsymbol{y}_i - \boldsymbol{y}'_i \|_2^2 \\ & \boldsymbol{y} \in \mathcal{U}, \\ & d(\boldsymbol{y}, \hat{\boldsymbol{y}}_i) \leq \theta. \end{aligned} \tag{23}$$

If we introduce a Lagrange multiplier γ for the last constraint, we note that we have to solve

$$\begin{aligned} \min\ & \sum_{i=1}^N \langle \boldsymbol{y}, \boldsymbol{h} \rangle + \frac{1}{2\sigma} \| \boldsymbol{y}_i - \boldsymbol{y}'_i \|_2^2 + \gamma \cdot d(\boldsymbol{y}, \hat{\boldsymbol{y}}_i) \\ & \boldsymbol{y} \in \mathcal{U}. \end{aligned} \tag{24}$$

It is straightforward to use the same methods as for the proximal updates for $p < +\infty$ and $d = d_1, d_2, d_\infty$, which yields the following corollary of Proposition 4.

Corollary 8. *1. Let $d = d_2$ and $p = 2$. The proximal update (24) can be computed in $O\left(NAS \log(S) \log(\epsilon^{-1})\right)$ arithmetics operations.*

2. Let $d = d_1$ and $p = 1$. The proximal update (24) can be computed in $O\left(NAS \log(S) \log(\epsilon^{-1})\right)$ arithmetics operations.

3. Let $d = d_\infty$ and $p = 1$. The proximal update (24) can be computed in $O\left(NAS \log(S) \log^3(\epsilon^{-1})\right)$ arithmetics operations.

The corresponding convergence rates for Algorithm 1 with $p = +\infty$ are given in Theorem 5.

E. Computing the Duality Gap

Remember that the duality gap in (1) is defined as

$$\max_{\mu \in \mathbb{D}} C(\boldsymbol{x}, \mu) - \min_{\boldsymbol{x'} \in \Pi} C(\boldsymbol{x'}, \mu).$$

Following Yang (2017), $\max_{\mu \in \mathbb{D}} C(\boldsymbol{x}, \mu)$ can be computed by finding the fixed point of the following operator, which is a contraction of factor λ: $F^{\boldsymbol{x}}(\boldsymbol{v})_s = \max_{\mu \in \mathbb{D}_s} \mathbb{E}_{\boldsymbol{y} \sim \mu} \left[\sum_{a=1}^A x_{sa} \left(c_{sa} + \lambda \boldsymbol{y}^\top \boldsymbol{v} \right) \right], \forall\ s \in \mathbb{S}$. Moreover, computing $\min_{\boldsymbol{x'} \in \Pi} C(\boldsymbol{x'}, \mu)$ is equivalent to solving the (nominal) MDP with fixed density $\mu \in \mathbb{D}$. This can be solved by iterating the following contraction: $F^{\boldsymbol{y}}(\boldsymbol{v})_s = \min_{\boldsymbol{x_s} \in \Delta(A)} \mathbb{E}_{\boldsymbol{y} \sim \mu} \left[\sum_{a=1}^{A} x_{sa} \left(c_{sa} + \lambda \boldsymbol{y}^\top \boldsymbol{v} \right) \right], \forall s \in \mathbb{S}$.

We present in the next figure the running times to compute (DG) up to $\epsilon = 0.25$, using the numerical setup of our numerical experiments for Garnet MDPs. We present our results for $\lambda = 0.8$. We notice that computing (DG) quickly becomes slow. Therefore, in our experiments we focus on computing (DG) for S, A, N smaller than 70.

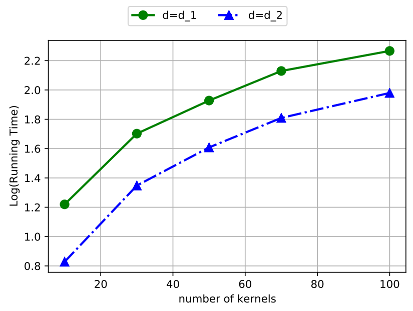


Figure 3: Running times for computing the duality gap (DG), for increasing number of kernels (while $S, A = 10$).

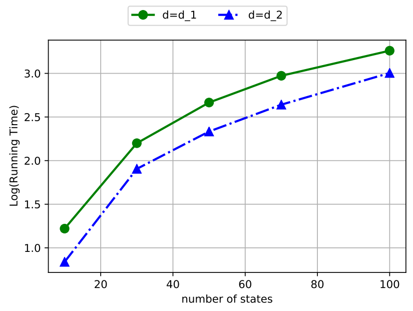


Figure 4: States.

Figure 5: Running times for computing the duality gap (DG), for increasing number of states (while $N, A = 10$).

We also note here that the duality gap is slower to compute for $d = d_1$ (where the Bellman update brings down to a large linear program) than for $d = d_2$ (where the Bellman update brings down to a convex program with less variables than for $d = d_1$ but one additional quadratic constraints). Note that in the case of $d = d_1$, NAS additional variables have to be introduced to model the absolute values $|y_{i,a,s'} - \hat{y}_{i,a,s'}|$ for all $i = 1, ..., N, a \in \mathbb{A}, s' \in \mathbb{S}$; this is probably what causes the Bellman update with $d = d_2$ to be faster, even if it introduces a (single) quadratic constraint.

F. Details on Numerical Implementations

Estimating the Bellman operator. In order to obtain $F(\boldsymbol{v})$, we use the reformulation (16) and solve it using Gurobi 8.1.1 for Python 3.7.3. Following Ben-Tal & Nemirovski (2001), we can solve (16) up to accuracy ϵ in a number of arithmetic operations in $O\left(S^{3.5} A^{3.5} \log(1/\epsilon)\right)$. For VI, AVI, Anderson and GSVI, we warm-start the computation of $F(\boldsymbol{v}^\ell)$ with the previous solution obtained from solving $F(\boldsymbol{v}^{\ell-1})$.

Computing uncertainty sets. In order to obtain the N transition kernels $(\hat{\boldsymbol{y}}_i)_{i=1}^N$, we sample some *random* (Garnet) deviations around the true nominal kernel $\boldsymbol{y}^0$. In particular, we sample N Garnet MDP instances $\boldsymbol{y}_1, ..., \boldsymbol{y}_N$ with $n_b = 0.05$ (very low level of connectivity), and we consider $\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_N$ as

$$\hat{\boldsymbol{y}}_i = 0.95 \boldsymbol{y}^0 + 0.05 \boldsymbol{y}_i, i = 1, ..., N.$$

This way, $(\hat{\boldsymbol{y}}_i)_{i=1}^N$ represent N kernels, obtained as small (random) errors from the true transition kernel $\boldsymbol{y}^0$. The nominal kernel for the machine replacement and the forest management instances are given in the next appendices.

For the machine replacement and the forest management problems, we build an uncertainty set of the form (4) with $\theta = 0.5$. We choose to present our results for $\theta = 0.5$ as they are representative of our results for other choices ($\theta \in \{0.1, 0.5, 1, 2\}$). As the Garnet MDPs have denser transitions, we choose $\theta = \sqrt{n_b A}$ as the radius for the Wasserstein balls.

Accelerated Value Iteration. The algorithm AVI (Goyal & Grand-Clément, 2018; Akian et al., 2020) is a simple variation of VI, inspired from acceleration scheme from convex optimization (Nesterov, 1983; 2013). In particular, for any sequences of scalar $(\alpha_s)_{s \geq 0}$ and $(\gamma_s)_{s \geq 0} \in \mathbb{R}^{\mathbb{N}}$, Accelerated Value Iteration (AVI) is defined as

$$\boldsymbol{v}_0, \boldsymbol{v}_1 \in \mathbb{R}^S, \begin{cases} \boldsymbol{h}_t = \boldsymbol{v}_t + \gamma_t \cdot (\boldsymbol{v}_t - \boldsymbol{v}_{t-1}), \\ \boldsymbol{v}_{t+1} \leftarrow \boldsymbol{h}_t - \alpha_t \left(\boldsymbol{h}_t - F\left(\boldsymbol{h}_t\right)\right), \end{cases} \quad \forall\, t \geq 1. \tag{AVI}$$

Following (Goyal & Grand-Clément, 2018), we choose step sizes as

$$\alpha_s = \alpha = 1/(1+\lambda), \gamma_s = \gamma = \left(1 - \sqrt{1-\lambda^2}\right)/\lambda, \forall s \ \geq 1.$$

Gauss-Seidel Value Iteration. Gauss-Seidel Value Iteration (GS-VI) is a popular asynchronous variant of VI (Puterman, 1994), where $v_s^{t+1} = \max_{a \in \mathbb{A}} \min_{\boldsymbol{y} \in \mathbb{B}_{p,s}} c_{sa} + \lambda \cdot \sum_{s'=1}^{s-1} y_{sas'} v_{s'}^{t+1} + \lambda \cdot \sum_{s'=s}^{n} y_{sas'} v_{s'}^{t}$.

Anderson Value Iteration. This algorithm (Geist & Scherrer, 2018), inspired from quasi-Newton methods from convex optimization, updates $\boldsymbol{v}^{t+1}$ as a linear combination of the last $(m+1)$-iterates $F(\boldsymbol{v}^t), ..., F(\boldsymbol{v}^{t-m})$:

$$\boldsymbol{v}^{t+1} = \sum_{i=0}^{m} \alpha_i F(\boldsymbol{v}^{t-m+i}),$$

for some weights $\alpha_0, ..., \alpha_m$. The weights $\boldsymbol{\alpha} \in \mathbb{R}^{m+1}$ are updated at every iteration, see Algorithm 1 and Equation (1) in (Geist & Scherrer, 2018) for further details. There is no heuristics for choosing m; we choose $m = 5$ in our numerical experiments.

G. Details on Machine Replacement Example

We present an example of this instance in Figure 6-7, where there are 10 states: 8 states related to the condition of the machine, and two repair states. The instances for larger number of states are constructed in the same fashion by adding some condition states for the machine. Below we give details about the states, actions, transitions and rewards.

States. The machine replacement problem involves a machine whose set of possible conditions are described by S states. The first $S-2$ states are operative states. The states 1 to $S-2$ model the condition of the machine, with 1 being perfect condition and $S-2$ being worst condition. The last two states $S-1$ and S are states representing when the machine is being repaired. The initial distribution is uniform across states.

Actions. There are two actions: *repair* and *no repair*.

Transitions. The transitions are detailed in Figures 6-7. When the action is *no repair*, the machine is likely to deteriorates toward the state $S-2$, or may stay in the same condition. When the action is *repair*, the decision-maker brings the machine to the states $S-1$ and $S-2$.

Rewards. There is a cost of 0 for states $1, ..., S-3$; letting the machine reach the worst operative state $S-2$ is penalized with a cost of 20. The state $S-1$ is a standard repair state and has a cost of 2, while the last state S is a longer and more costly repair state and has cost 10.

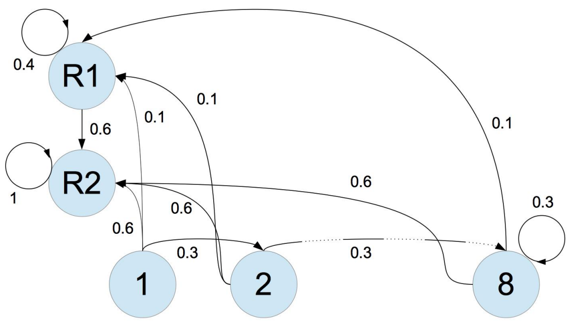


Figure 6: Nominal transition for action = *repair* in our machine replacement MDP.

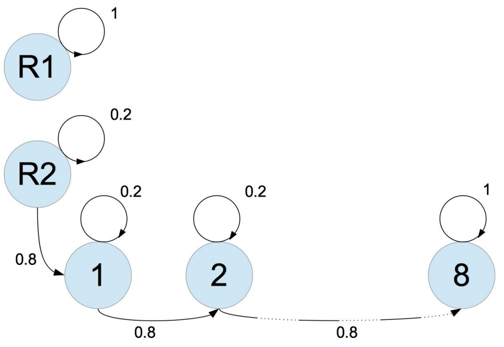


Figure 7: Nominal transition for action = *no repair* in our machine replacement MDP.

H. Details on Forest Management Example

The state in the forest management example represents the growth of the forest. The goal is to find the right balance between maintaining the forest, making money by selling cut wood. Every year, the forest may suffer from wildfires. A complete description may be found at (Cordwell et al., 2015). This is inspired from the application of dynamic programming to optimal fire management (Possingham & Tuck, 1997).

States. There are S states. The state 1 is the youngest state for the forest. The forest can not grow beyond state S. The initial distribution is uniform across states.

Actions. There are two actions, *wait* and *cut & sell*.

Transitions. If the forest is in a state s and the action is *wait*, the next state is $s+1$ with probability $1-p$ (the forest grows) and 1 with probability p (a wildfire burns the forest down). If the forest is in a state s and the action is *cut & sell*, the next state is 1 with probability 1. The probability of wildfire p is chosen at $p = 0.1$.

Rewards. There is a reward of 4 when the forest reaches the oldest state (S) and the chosen action is *wait*. There is a reward of 0 at every other state if the chosen action is *wait*. When the action is *cut & sell*, the reward at the youngest state $s = 1$ is 0, there is a reward of 1 in any other state $s \in \{1, ..., S - 1\}$, and a reward of 2 in $s = S$. We convert all rewards to cost by flipping the signs of the rewards.