# Scalable First-Order Methods for Robust MDPs

**Julien Grand-Clément, Christian Kroer**
IEOR Department, Columbia University
jg3728@columbia.edu, christian.kroer@columbia.edu

**Abstract**

Robust Markov Decision Processes (MDPs) are a powerful framework for modeling sequential decision making problems with model uncertainty. This paper proposes the first first-order framework for solving robust MDPs. Our algorithm interleaves primal-dual first-order updates with approximate Value Iteration updates. By carefully controlling the tradeoff between the accuracy and cost of Value Iteration updates, we achieve an ergodic convergence rate that is significantly better than classical Value Iteration algorithms in terms of the number of states S and the number of actions A on ellipsoidal and Kullback-Leibler s-rectangular uncertainty sets. In numerical experiments on ellipsoidal uncertainty sets we show that our algorithm is significantly more scalable than state-of-the-art approaches. Our framework is also the first one to solve robust MDPs with s-rectangular KL uncertainty sets.

## 1 Introduction

In this paper we focus on solving robust Markov Decision Processes (MDPs) with finite set of states and actions. Markov decision process models are widely used in decision-making (Bertsekas 2007; Puterman 1994). In the classical MDP setting, for each state $s \in \mathbb{S}$, the decision maker chooses a probability distribution $\boldsymbol{x}_s$ over the set of actions $\mathbb{A}$. The decision maker incurs a cost $\sum_{a=1}^{|\mathbb{A}|} x_{sa} c_{sa}$ for some non-negative scalars $c_{sa}$ and then randomly enters a new state, according to transition kernels $\boldsymbol{y} = \{\boldsymbol{y}_{sa} \in \Delta(|\mathbb{S}|)\}_{sa}$ over the next state, where $\Delta(|\mathbb{S}|)$ is the simplex of size $|\mathbb{S}|$. Given a discount factor $\lambda$, the goal of the decision-maker is to minimize the infinite horizon discounted expected cost $R(\boldsymbol{x}, \boldsymbol{y}) = E^{\boldsymbol{x},\boldsymbol{y}} \left[ \sum_{t=0}^{\infty} \lambda^t c_{s_t a_t} \,\middle|\, \boldsymbol{s}_0 \sim \boldsymbol{p}_0 \right]$.

The cost of a policy can be highly sensitive to the exact kernel parameters $\boldsymbol{y}$. We consider a robust approach where the uncertainty in $\boldsymbol{y}$ is adversarially selected from an *uncertainty set* $\mathbb{P}$ centered around a nominal estimation $\boldsymbol{y}^0$ of the true transition kernel. Our goal is to solve the *robust MDP* problem $\min_{\boldsymbol{x}\in\Pi} \max_{\boldsymbol{y}\in\mathbb{P}} R(\boldsymbol{x}, \boldsymbol{y})$ (Iyengar 2005; Nilim and Ghaoui 2005; Wiesemann, Kuhn, and Rustem 2013; Goh et al. 2018; Goyal and Grand-Clement 2018), which has found applications in healthcare (Steimle and Denton 2017; Steimle, Kaufman, and Denton 2018; Goh et al. 2018; Grand-Clement et al. 2020). We focus on $s$-rectangular uncertainty sets, where $\mathbb{P} = \underset{s\in\mathbb{S}}{\times} \mathbb{P}_s$, for $\mathbb{P}_s \subseteq \mathbb{R}_+^{|\mathbb{A}|\times|\mathbb{S}|}$, and solving the robust MDP problem is equivalent to computing the fixed point of the *Bellman operator*, thus allowing a *value iteration* (VI) algorithm (Wiesemann, Kuhn, and Rustem 2013).

We focus on two specific classes of $s$-rectangular uncertainty sets. *Kullback-Leibler* (KL) uncertainty sets are constructed from density estimation and naturally appear as approximations of the confidence intervals for the maximum likelihood estimates of $\boldsymbol{y}$ given some historical transition data (Iyengar 2005). *Ellipsoidal* uncertainty sets are widely used because of their tractability and the probabilistic guarantees of the optimal solutions of the robust problems (Ben-Tal and Nemirovski 2000; Bertsimas, den Hertog, and Pauphilet 2019). For ellipsoidal uncertainty sets, the value iteration algorithm involves solving a convex program with a quadratic constraint at every epoch. While this can be done in polynomial time with modern Interior Point Methods (IPMs, (Lobo et al. 1998)), this requires inverting matrices at every step of the IPM which can be intractable for large MDP instances. Typically, for $S = |\mathbb{S}|, A = |\mathbb{A}|$, the complexity of VI to return an $\epsilon$-solution to the robust MDP problem with ellipsoidal uncertainty sets is $O\left(A^{3.5} S^{4.5} \log^2(\epsilon^{-1})\right)$. This may prove prohibitive for large instances. For KL uncertainty sets, we are not aware of any tractable algorithm for solving $s$-rectangular robust MDP with KL uncertainty sets, even though they are well understood in Distributionally Robust Optimization (Hu and Hong 2013).

Many problems in machine learning and game theory can be written in the form $\min_{\boldsymbol{x}\in X} \max_{\boldsymbol{y}\in Y} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})$, where $\mathcal{L}$ is a convex-concave function and $X, Y$ are reflexive Banach spaces, e.g. regularized finite-sum loss minimization, imaging models, and sequential two-player zero-sum games (Chambolle and Pock 2011; Kroer et al. 2018). Even though convex duality often allows reformulating this saddle-point problem as a single convex program, first-order methods (FOMs) such as Chambolle & Pock's Primal-Dual Algorithm (PDA, (Chambolle and Pock 2011)), Mirror Descent (Nemirovski and Yudin 1983) or Mirror Prox (Nemirovski 2004) are typically preferred for large instances. This is due to the expensive matrix calculations involved in IPMs or the simplex algorithm. Naively, one may hope to apply FOMs directly to

the robust MDP problem, which looks superficially similar to the saddle-point problem. However, since the robust MDP problem is not convex-concave FOMs may fail to converge.

Our main contributions can be summarized as follows.

**A first-order method for robust MDP.** We present a new algorithmic framework for solving robust MDPs that is significantly more scalable than previous methods. Our algorithm adapts FOMs for solving static zero-sum games to a dynamic setting with varying payoff matrices. Only cheap proximal mappings need to be computed at each iteration. Our framework interleaves FOM updates with occasional approximate VI updates. By carefully controlling the pace of VI updates, and developing bounds on the change in the payoff matrices of the zero-sum games, we show that the ergodic average of policies generated by our framework converges (in value) at a rate of nearly $\frac{1}{T}$ in terms of the number of FOM steps $T$.

**Suitable proximal setups.** Our algorithmic framework is general and works for any uncertainty set for which a suitable proximal setup exists. We instantiate our algorithm on KL and ellipsoidal uncertainty sets. To the best of our knowledge, our algorithm is the first to address $s$-rectangular KL uncertainty sets for robust MDPs, and is the most scalable for ellipsoidal $s$-rectangular uncertainty sets in terms of number of states and actions.

**Empirical performance.** We focus our numerical experiments on ellipsoidal and KL uncertainty sets. We investigate several proximal setups, and find that an $\ell_2$ setup performs better than an $\ell_1$ setup, despite better theoretical guarantees for the $\ell_1$ setup. Similar observations have been made for numerical performance on stationary saddle-point optimization (Chambolle and Pock 2016; Gao, Kroer, and Goldfarb 2019). Finally, we show that our approach is significantly more scalable than state-of-the-art VI setups, both on random instances and on applications inspired from healthcare and machine replacement.

## Related Work

We now present a summary of the most related literature.

**Approximate value iteration and regularized MDP.** For the *nominal* MDP setting, approximate Value Iteration (de Farias and Roy 2003; Petrik 2010; Scherrer et al. 2015) considers inexact Bellman updates which arise from sample-based errors or function approximation; note that contrary to works involving value function approximation, we are solving the MDP up to any chosen accuracy. Geist, Scherrer, and Pietquin (2019) adds KL regularization to the VI update for nominal MDP and relate this to Mirror Descent. In contrast to Geist, Scherrer, and Pietquin (2019), we focus on robust MDPs, where VI requires computing the robust Bellman operator. Our framework is based on a connection to zero-sum games with changing payoff matrices and we use FOMs to efficiently approximate the robust Bellman update. This is very different from Geist, Scherrer, and Pietquin (2019), where regularized VI itself is treated as a FOM.

**Faster value iteration algorithms.** For nominal MDPs, several algorithms have been proposed to accelerate the convergence of VI, including Anderson mixing (Zhang, O'Donoghue, and Boyd 2018; Geist and Scherrer 2018) and acceleration and momentum schemes (Goyal and Grand-Clement 2019). However, these methods modify the VI algorithm itself, and do not accelerate the computation of each Bellman update.

**Faster Bellman updates.** For $s,a$-rectangular uncertainty sets, robust Bellman updates were studied for norm-ball uncertainty sets for the $\ell_1$ and weighted $\ell_1$ norms ((Iyengar 2005; Ho, Petrik, and W.Wiesemann 2018)), $\ell_2$ norm (Iyengar 2005), KL-divergence (Nilim and Ghaoui 2005) and $\ell_\infty$ norm (Givan, Leach, and Dean 1997). To the best of our knowledge, the only paper on fast computation of robust Bellman updates for $s$-rectangular uncertainty sets is Ho, Petrik, and W.Wiesemann (2018) which considers weighted $\ell_1$-balls for $\mathbb{P}_s$ and attains a complexity of $O\left(AS^3\log(AS^2)\log(\epsilon^{-1})\right)$ for finding an $\epsilon$-solution to the robust MDP problem. This complexity result relies on linear programming theory and cannot directly be extended to other settings for $\mathbb{P}_s$ (e.g. ellipsoidal or KL uncertainty sets).

## 2 Preliminaries on Robust MDP

**Notation.** We write $|\mathbb{S}| = S, |\mathbb{A}| = A$ and we assume $S < +\infty, A < +\infty$. Given a policy $\boldsymbol{x} \in \Pi = (\Delta(A))^S$ and a kernel $\boldsymbol{y} \in \mathbb{P}$, we define the one-step cost vector $\boldsymbol{c_x} \in \mathbb{R}^S$ and the value vector $\boldsymbol{v}^{\boldsymbol{x},\boldsymbol{y}} \in \mathbb{R}^S$ as $c_{\boldsymbol{x},s} = \sum_{a=1}^A x_{sa}c_{sa}$, $v_s^{\boldsymbol{x},\boldsymbol{y}} = E^{\boldsymbol{x},\boldsymbol{y}}\left[\sum_{t=0}^\infty \lambda^t c_{s_t a_t} \,\middle|\, s_0 = s\right], \forall\, s \in \mathbb{S}\,.$

**Value Iteration.** We first define Value Iteration (VI) for general $s$-rectangular uncertainty sets. Let $\mathbb{P} = \underset{s\in\mathbb{S}}{\times} \mathbb{P}_s$, for $\mathbb{P}_s \subseteq \mathbb{R}_+^{|\mathbb{A}|\times|\mathbb{S}|}$. and let us define the (robust) *Bellman operator* $F : \mathbb{R}^S \to \mathbb{R}^S$, where for $\boldsymbol{v} \in \mathbb{R}^S$,

$$F(\boldsymbol{v})_s = \min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\boldsymbol{y}_s\in\mathbb{P}_s} \left\{\sum_{a=1}^A x_{sa}\left(c_{sa} + \lambda \boldsymbol{y}_{sa}^\top \boldsymbol{v}\right)\right\}, \tag{2.1}$$

for each $s \in \mathbb{S}$. Note that with the notation $F^{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{v})_s = \sum_{a=1}^A x_{sa}\left(c_{sa} + \lambda \cdot \boldsymbol{y}_{sa}^\top \boldsymbol{v}\right)$, we can also write $F(\boldsymbol{v})_s = \min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\boldsymbol{y}_s\in\mathbb{P}_s} F^{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{v})_s$, which shows that the robust VI update is a stationary saddle-point problem. Solving the robust MDP problem is equivalent to computing $\boldsymbol{v}^*$, the fixed-point of $F$:

$$v_s^* = \min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\boldsymbol{y}_s\in\mathbb{P}_s} \{\sum_{a=1}^A x_{sa}\left(c_{sa} + \lambda \cdot \boldsymbol{y}_{sa}^\top \boldsymbol{v}^*\right)\}, \forall\, s \in \mathbb{S}. \tag{2.2}$$

Since $F$ is a contraction with factor $\lambda$, this can be done with the Value Iteration (VI) Algorithm:

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

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 optimal pair $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ can be computed as any pair attaining the $\min\max$ in $F(\boldsymbol{v}^*)$. An $\epsilon$-optimal pair can be computed as a solution to (2.1), when $\|\boldsymbol{v} - F(\boldsymbol{v})\|_\infty < \epsilon(1-\lambda)(2\lambda)^{-1}$ (Puterman 1994). Our algorithm relies on approximately solving 2.1 as part of VI; controlling $\epsilon_\ell$, the accuracy of epoch $\ell$ of VI, plays a crucial role in the analysis of our algorithm. In the appendix,

we present *approximate* Value Iteration, where the Bellman update $F(\boldsymbol{v}^\ell)$ at epoch $\ell$ is only computed up to accuracy $\epsilon_\ell$.

**Ellipsoidal and KL uncertainty sets.** We will show specific results for two types of $s$-rectangular uncertainty sets, though our algorithmic framework applies more generally, as long as appropriate proximal mappings can be computed. We consider KL $s$-rectangular uncertainty sets where $\mathbb{P}_s$ equals

$$\{(\boldsymbol{y}_{sa})_{a\in\mathbb{A}} \in (\Delta(S))^A \mid \sum_{a\in\mathbb{A}} KL(\boldsymbol{y}_{sa}, \boldsymbol{y}^0_{sa}) \leq \alpha\}, \quad (2.3)$$

and *ellipsoidal* $s$-rectangular uncertainty sets where $\mathbb{P}_s$ equals

$$\{(\boldsymbol{y}_{sa})_{a\in\mathbb{A}} \in (\Delta(S))^A \mid \sum_{a\in\mathbb{A}} \frac{1}{2}\|\boldsymbol{y}_{sa} - \boldsymbol{y}^0_{sa}\|_2^2 \leq \alpha\}. \quad (2.4)$$

Note that (2.4) is different from the ellipsoidal uncertainty sets considered in Ben-Tal and Nemirovski (2000), which also adds box constraints. However, Bertsimas, den Hertog, and Pauphilet (2019) shows that the same probabilistic guarantees exist for (2.4) as in the case of the uncertainty sets considered in Ben-Tal and Nemirovski (2000). For solving $s$-rectangular KL uncertainty sets, no algorithm is known (contrary to the significantly more conservative $s,a$-rectangular case). Wiesemann, Kuhn, and Rustem (2013) solves $s$-rectangular ellipsoidal uncertainty sets (2.4) using conic programs; we choose to instantiate VI differently in this case as follows. Using min-max convex duality twice, we can reformulate each of the $S$ min-max programs (2.1) into a larger convex program with linear objective and constraints, and one quadratic constraint.Using IPMs each program can be solved up to $\epsilon$ accuracy in $O\left(A^{3.5}S^{3.5}\log(1/\epsilon)\right)$ arithmetic operations ((Ben-Tal and Nemirovski 2001), Section 4.6.2). Therefore, the complexity of (VI) is

$$O\left(A^{3.5}S^{4.5}\log^2(\epsilon^{-1})\right). \quad (2.5)$$

As mentioned earlier, this becomes intractable as soon as the number of states becomes on the order of hundreds, as highlighted in our numerical experiments of Section 4.

## 3 First-Order Methods for Robust MDPs

We start by briefly introducing first-order methods (FOMs) in the context of our problem, and giving a high-level overview of our first-order framework for solving robust MDPs. A FOM is a method that iteratively produces pairs of solution candidates $\boldsymbol{x}^t, \boldsymbol{y}^t$, where the $t$'th solution pair is derived from $\boldsymbol{x}^{t-1}, \boldsymbol{y}^{t-1}$ combined with a first-order approximation to the direction of improvement at $\boldsymbol{x}^{t-1}, \boldsymbol{y}^{t-1}$. Using only first-order information is desirable for large-scale problems because second-order information eventually becomes too slow to compute, meaning that even a single iteration of a second-order method ends up being intractable. See e.g. Beck (2017) or Ben-Tal and Nemirovski (2001) for more on FOMs.

Our algorithmic framework is based on the observation that there exists a collection of matrices $\boldsymbol{K}^*_s : \Delta(A) \times \mathbb{P}_s \to \mathbb{R}$, for $s \in \mathbb{S}$, such that computing an optimal solution $\boldsymbol{x}^*, \boldsymbol{y}^*$ to the robust MDP problem boils down to solving $S$ bilinear saddle-point problems (BSPPs), each of the form

$$\min_{\boldsymbol{x}_s\in\Delta(A)} \max_{\boldsymbol{y}_s\in\mathbb{P}_s} \langle \boldsymbol{c}_s, \boldsymbol{x}_s\rangle + \langle \boldsymbol{K}^*_s \boldsymbol{x}_s, \boldsymbol{y}_s\rangle. \quad (3.1)$$

This is a straightforward consequence of the Bellman equation 2.1 and its reformulation using $F^{\boldsymbol{x},\boldsymbol{y}}$. The matrix $\boldsymbol{K}^*_s$ is the payoff matrix associated with the optimal value vector $v^*$. If we knew $\boldsymbol{K}^*_s$, then we could solve (3.1) by applying existing FOMs for solving BSPPs.

Now, obviously we do not know $\boldsymbol{v}^*$ before running our algorithm. However, we know that Value Iteration constructs a sequence $\{\boldsymbol{v}_\ell\}_{\ell\geq 0}$ which converges to $\boldsymbol{v}^*$. Letting $\{\boldsymbol{K}^\ell_s\}_{\ell\geq 0}$ be the associated payoff matrices for each value-vector estimate $\boldsymbol{v}_\ell$ and state $s$, we thus have a sequence of payoff matrices converging to $\boldsymbol{K}^*_s$ for each $s$. We will apply a FOM to such a sequence of BSPPs $\{\boldsymbol{K}^\ell_s\}_{\ell\geq 0}$ based on approximate Value Iteration updates.

Our algorithmic framework, which we call FOM-VI, works as follows. We utilize an existing primal-dual FOM for solving problems of the form (3.1), where the FOM should be of the type that generates a sequence of iterates $\boldsymbol{x}_t, \boldsymbol{y}_t$, with an ergodic convergence rate on the time-averaged iterates. Even though such FOMs are designed for a *fixed* BSPP with a single payoff matrix $\boldsymbol{K}$, we apply the FOM updates to a changing sequence of payoff matrices $\{\boldsymbol{K}^\ell_s\}^k_{\ell=1}$. For each payoff matrix $\boldsymbol{K}^\ell_s$ we apply $T_\ell$ iterations of the FOM, after which we apply an approximate VI update to generate $\boldsymbol{K}^{\ell+1}_s$. We refer to each step $\ell$ with a payoff matrix $\boldsymbol{K}^\ell_s$ as an *epoch*, while *iteration* refers to steps of our FOM. We will apply many iterations per epoch.

The convergence rate of our algorithm is, intuitively, based on the following facts: (i) the average of the iterates generated during epoch $\ell$ provides a good estimate of the VI update associated with $\boldsymbol{v}^\ell$, and (ii) the sequence of payoff matrices generated by the approximate VI updates is changing in a controlled manner, such that $\boldsymbol{K}^\ell_s$ and $\boldsymbol{K}^{\ell+1}_s$ are not too different.

These facts allow us to show that the averaged strategy across *all* epochs $\ell$ converges to a solution to (3.1) without too much degradation in the convergence rate compared to having run the same number of iterations directly on (3.1).

### First-Order Method Setup

In this paper, we use the PDA algorithm of Chambolle and Pock (2016) as our FOM, but the derivations could also be performed with other FOMs whose convergence rate is based on applying a telescoping argument to a sum of descent inequalities, e.g. mirror prox of Nemirovski (2004) or saddle-point mirror descent of Ben-Tal and Nemirovski (2001); the latter would yield a slower rate of convergence.

We now describe PDA as it applies to BSPPs such as (3.1), for an arbitrary payoff matrix $\boldsymbol{K}$ and some state $s$. PDA relies on what we will call a *proximal setup*. A proximal setup consists of a set of norms $\|\cdot\|_X, \|\cdot\|_Y$ for the spaces of $\boldsymbol{x}_s, \boldsymbol{y}_s$, as well as *distance-generating functions* $\psi_X$ and $\psi_Y$, which are 1-strongly convex with respect to $\|\cdot\|_X$ on $\Delta(A)$ and $\|\cdot\|_Y$ on $\mathbb{P}_s$, respectively. Using the distance-generating functions, PDA uses the *Bregman divergence*

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

to measure the distance between two points $\boldsymbol{x}, \boldsymbol{x'} \in \Delta(A)$. The Bregman divergence $D_Y$ is defined analogously.

The convergence rate of PDA then depends on the maximum Bregman divergence distance $\Theta_X = \max_{\boldsymbol{x},\boldsymbol{x'}\in\Delta(A)} D_X(\boldsymbol{x},\boldsymbol{x'})$ between any two points, and the maximum norm $R_X = \max_{\boldsymbol{x}\in\Delta(A)} \|\boldsymbol{x}\|_X$ on $\Delta(A)$. The quantities $\Theta_Y$ and $R_Y$ are defined analogously on $\mathbb{P}_s$.

Given $D_X$ and $D_Y$, the associated *prox mappings* are

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

Intuitively, the prox mappings generalize taking a step in the direction of the negative gradient, as in gradient descent. Given some gradient $\boldsymbol{g}_x$, $\text{prox}_x(\boldsymbol{g}_x, \boldsymbol{x'_s})$ moves in the direction of improvement, but is penalized by the Bregman divergence $D_X(\boldsymbol{x}_s, \boldsymbol{x'_s})$, which attempts to ensure that we stay in a region where the first-order approximation is still good.

Given step sizes $\tau, \sigma > 0$ and current iterates $\boldsymbol{x}_s^t, \boldsymbol{y}_s^t$, PDA generates the iterates for $t+1$ by taking prox steps in the negative gradient direction given the current strategies:

$$\begin{aligned} \boldsymbol{x}_s^{t+1} &= \text{prox}_x(\tau \boldsymbol{K}^\top \boldsymbol{y}_s^t, \boldsymbol{x}_s^t), \\ \boldsymbol{y}_s^{t+1} &= \text{prox}_y(\sigma \boldsymbol{K}(2\boldsymbol{x}_s^{t+1} - \boldsymbol{x}_s^t), \boldsymbol{x}_s^t), \end{aligned} \tag{3.2}$$

Note that for the $\boldsymbol{y}_s^{t+1}$ update, the "direction of improvement" is measured according to the extrapolated point $2\boldsymbol{x}_s^{t+1} - \boldsymbol{x}_s^t$, as opposed to at either $\boldsymbol{x}_s^t$ or $\boldsymbol{x}_s^{t+1}$. If a simpler single current iterate is used to take the gradient for $\boldsymbol{y}_s^{t+1}$, then the overall PDA setup yields an algorithm that converges at a $O(1/\sqrt{T})$ rate. The extrapolation is used to get a stronger $O(1/T)$ rate.

Let $\Omega = 2\left(\Theta_X/\tau + \Theta_Y/\sigma\right)$ and let $\tau, \sigma > 0$ be such that, for $L_{\boldsymbol{K}} \geq \sup_{\|\boldsymbol{x}\|_X\leq 1, \|\boldsymbol{y}\|_Y\leq 1} \langle \boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}\rangle$, we have

$$\left(\frac{1}{\tau} - L_f\right)\frac{1}{\sigma} \geq L_{\boldsymbol{K}}^2. \tag{3.3}$$

After $T$ iterations of PDA, we can construct weighted averages $(\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T) = (1/S_T)\sum_{t=1}^T \omega_t(\boldsymbol{x}_t, \boldsymbol{y}_t)$ of all iterates, using weights $\omega_1, ..., \omega_T$ and normalization factor $S_T = \sum_{t=1}^T \omega_t$. In the case of a static BSPP, if the stepsizes are chosen such that they satisfy (3.3), then the average of the iterates from PDA satisfies the convergence rate:

$$\max_{\boldsymbol{y}\in\mathbb{P}_s} \langle \boldsymbol{K}\bar{\boldsymbol{x}}^T, \boldsymbol{y}\rangle - \min_{\boldsymbol{x}\in\Delta(A)} \langle \boldsymbol{K}\boldsymbol{x}, \bar{\boldsymbol{y}}^T\rangle \leq \Omega\omega_T/S_T.$$

Here we are using the weighted average of iterates, as in Gao, Kroer, and Goldfarb (2019).Since PDA applies the two prox mappings (3.2) at every iteration, it is crucial that these prox mappings can be computed efficiently. Ideally, in time roughly linear in the dimension of the iterates. A significant part of our contribution is to show that this is indeed the case for several important types of uncertainty sets.

In our setting, where the payoff matrix $\boldsymbol{K}_s^\ell$ in the BSPP is changing over time, the existing convergence rate for PDA does not apply. Instead, we have to consider how to deal with the error that is introduced in the process due to the changing payoffs.

**Algorithm 1** First-order Method for Robust MDP with $s$-rectangular uncertainty set.

1: **Input** Number of epochs $k$, number of iterations per epoch $T_1, ..., T_k$, weights $\omega_1, ..., \omega_T$, and stepsizes $\tau, \sigma$.
2: **Initialize** $\ell = 1$, $\boldsymbol{v}^\ell = \boldsymbol{0}$, and $\boldsymbol{x}^0, \boldsymbol{y}^0$ at random.
3: **for** epoch $\ell = 1, ..., k$ **do**
4:  **for** $s \in \mathbb{S}$ **do**
5:  Set $\tau_\ell = T_1 + ... + T_{\ell-1}$
6:  **for** $t = \tau_\ell, \ldots, \tau_\ell + T_\ell$ **do**
7:  $\boldsymbol{x}_s^{t+1} = \text{prox}_x(\tau \boldsymbol{K}_s[\boldsymbol{v}^\ell]^\top \boldsymbol{y}_s^t, \boldsymbol{x}_s^t)$
8:  $\boldsymbol{y}_s^{t+1} = \text{prox}_y(\sigma \boldsymbol{K}_s[\boldsymbol{v}^\ell](2\boldsymbol{x}_s^{t+1} - \boldsymbol{x}_s^t), \boldsymbol{y}_s^t)$
9:  $S_\ell = \sum_{t=\tau_\ell}^{\tau_\ell+T_\ell} \omega_t$
10:  $\bar{\boldsymbol{x}}_s^\ell = \sum_{t=\tau_\ell}^{\tau_\ell+T_\ell} \frac{\omega_t}{S_\ell}\boldsymbol{x}_s^t$, $\bar{\boldsymbol{y}}_s^\ell = \sum_{t=\tau_\ell}^{\tau_\ell+T_\ell} \frac{\omega_t}{S_\ell}\boldsymbol{y}_s^t$
11:  Update $v_s^{\ell+1} = F^{\bar{\boldsymbol{x}}_s^\ell, \bar{\boldsymbol{y}}_s^\ell}(\boldsymbol{v}^\ell)_s$.
12: **Output** $\bar{\boldsymbol{x}}_s^T = \sum_{t=1}^T \frac{\omega_t}{S_T}\boldsymbol{x}_s^t, \bar{\boldsymbol{y}}_s^T = \sum_{t=1}^T \frac{\omega_t}{S_T}\boldsymbol{y}_s^t, \forall s \in \mathbb{S}$

## First-Order Method Value Iteration (FOM-VI)

We now describe our algorithm in detail, as well as the choices of $\|\cdot\|_X, \|\cdot\|_Y$ that lead to tractable proximal updates (3.2). As we have described in (3.1), given a vector $\boldsymbol{v} \in \mathbb{R}^S$ and $s \in \mathbb{S}$, the matrix $\boldsymbol{K}_s \in \mathbb{R}^{A\times A\times S}$ is defined such that

$$F_s^{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{v}) = \langle \boldsymbol{c}_s, \boldsymbol{x}_s\rangle + \langle \boldsymbol{K}_s\boldsymbol{x}_s, \boldsymbol{y}_s\rangle.$$

We will write this as $\boldsymbol{K} = \boldsymbol{K}[\boldsymbol{v}]$. The pseudocode for the FOM-VI algorithm is given in Algorithm 1. At each epoch $\ell$, we have some current estimate $v^\ell$ of the value vector, which is used to construct the payoff matrix $\boldsymbol{K}^\ell$ for the $\ell$'th BSPP. For each state $s$, we then run $T_\ell$ iterations of PDA, where, crucially, the first such iteration starts from the last iterates $(\boldsymbol{x}_{\tau_\ell}, \boldsymbol{y}_{\tau_\ell})$ generated at the previous epoch. The average iterate constructed from just these $T_\ell$ iterations is then used to construct the next value vector $v_s^{\ell+1}$ via an approximate VI update (lines 10 and 11). Finally, after the last epoch $k$, we output the average of all the iterates generates across all the epochs, using the weights $\omega_1, ..., \omega_T$.

We prove that FOM-VI satisfies the following convergence rate. We state our results for the two special cases where the norms $\|\cdot\|_X$ and $\|\cdot\|_Y$ are both equal to the $\ell_1$ norm (we call this the $\ell_1$ *setup*) or $\ell_2$ norm ($\ell_2$ *setup*) on the spaces $\Delta(A), \mathbb{P}_s$. FOM-VI could also be instantiated with other norms. The proof is in the appendix.

**Theorem 3.1.** *Assume that the stepsizes $\tau, \sigma$ are such that* (3.3) *holds, and for each epoch $\ell$, we set $T_\ell = \ell^q$ for some $q \in \mathbb{N}$. Let $\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T$ be the averages of the FOM-VI iterates using the weights $w_1, ..., w_T$. Then for all states $s \in \mathbb{S}$,*

$$\begin{aligned} &\max_{\boldsymbol{y}\in\mathbb{P}_s} F^{\bar{\boldsymbol{x}}^T, \boldsymbol{y}}(\boldsymbol{v}^*)_s - \min_{\boldsymbol{x}\in\Delta(A)} F^{\boldsymbol{x}, \bar{\boldsymbol{y}}^T}(\boldsymbol{v}^*)_s \\ \leq& O\left(CR_XR_Y\left(\frac{\Theta_X}{\tau} + \frac{\Theta_Y}{\sigma}\right)\left(\frac{\lambda^{T^{1/(q+1)}}}{T^{1/(q+1)}} + \frac{1}{T^{q/(q+1)}}\right)\right), \end{aligned}$$

*with $C = 1$ in the $\ell_1$ setup, and $C = \sqrt{S}$ in the $\ell_2$ setup.*

## Tractable Proximal Setups for Algorithm 1

In the previous section we saw that FOM-VI instantiated with appropriate proximal setups yields an attractive con-

vergence rate. For a given proximal setup, the convergence rate in Theorem 3.1 depends on the maximum-norm quantities $R_X, R_Y$ and the polytope diameter measures $\Theta_X, \Theta_Y$. However, another important issue was previously not discussed: in order to run FOM-VI we must compute the iterates $x_s^{t+1}, y_s^{t+1}$, which means that the updates in (3.2) must be fast to compute (ideally in closed form). We next present several tractable proximal setups for Algorithm 1.

**Tractable updates for $\Delta(A)$.** Since decision space for $x$ is a simplex, we can apply well-known results to get a proximal setup. For the $\ell_2$ setup (i.e. where $\|\cdot\|_X = \|\cdot\|_2$), we can set $\psi_X(\boldsymbol{x}) = (1/2)\|\boldsymbol{x}\|_2^2$, in which case $D_X$ is the squared Euclidean distance. For this setup, $\Theta_X = 1$, and $\boldsymbol{x}_s^{t+1}$ can be computed in $A \log A$ time, using a well-known algorithm based on sorting (Ben-Tal and Nemirovski 2001).

For the $\ell_1$ setup, (i.e. where $\|\cdot\|_X = \|\cdot\|_1$), we set $\psi_X(\boldsymbol{x}) = \textsc{Ent}(\boldsymbol{x}) := \sum_i x_i \log x_i$ (i.e. the negative entropy), in which case $D_X$ is the *KL divergence*. The advantage of this setup is that the strong convexity is with respect to the $\ell_1$ norm, which makes the Lipschitz associated to the payoff matrix a constant (as opposed to $\sqrt{S}$ for the $\ell_2$ norm), while the polytope diameter is only $\Theta_X = \log A$. Finally, $\boldsymbol{x}_s^{t+1}$ can be computed in closed form. Thus, from a theoretical perspective, the $\ell_1$ setup is more attractive than the $\ell_2$ setup for $\Delta(A)$. This is well-known in the literature.

In all cases, $R_X = 1$, since $\boldsymbol{x}$ comes from a simplex.

**Tractable updates for ellipsoidal uncertainty.** The proximal updates for $y$ turn out to be more complicated. In the first place, they depend heavily on the form of $\mathbb{P}_s$. First, we present our results for the case where $\mathbb{P}_s$ is an ellipsoidal s-rectangular uncertainty set as in (2.4). We present both $\ell_1$ and $\ell_2$ setups.

In the $\ell_2$ setup for ellipsoidal uncertainty, we let $\|\cdot\|_Y$ be the $\ell_2$ norm, and $\psi_Y(\boldsymbol{y}) = (1/2)\|\boldsymbol{y}\|_2^2$. The Bregman divergence $D_Y(\boldsymbol{y}, \boldsymbol{y'})$ is then simply the squared Euclidean distance. In this case, we get that $R_Y = \sqrt{A}$, since the squared norm of each individual simplex is at most one, and then we take the square root. The polytope diameter is $\Theta_Y = 2A$ for the same reason. We show in Proposition 3.2 below that the iterate $\boldsymbol{y}_s^{t+1}$ can be computed efficiently.

In the $\ell_1$ setup for ellipsoidal uncertainty, we let $\|\cdot\|_Y$ be the $\ell_1$ norm, and $\psi_Y(\boldsymbol{y}) = (A/2)\sum_{a=1}^{A} \textsc{Ent}(\boldsymbol{y}_a)$, where $\textsc{Ent}(\boldsymbol{y}_a)$ is the negative entropy function. The Bregman divergence $D_Y(\boldsymbol{y}, \boldsymbol{y'}) = (A/2)\sum_{a=1}^{A} \mathrm{KL}(\boldsymbol{y}_a, \boldsymbol{y}'_a)$ is then a sum over KL divergences on each action. In this case, we get that $R_Y = A$, since we are taking the $\ell_1$ norm over $A$ simplexes, while the polytope diameter is $\Theta_Y = A^2 \log S$.

Proposition 3.2 shows that for both our $\ell_2$-based and $\ell_1$-based setup for $y$, the next iterate can be computed efficiently. We present a detailed proof in the appendix.

**Proposition 3.2.** *For the $\ell_2$ setup, the proximal update* (3.2) *with uncertainty set* (2.4) *can be approximated up to $\epsilon$ in a number of arithmetic operations of $O\left(AS\log(S)\log(\epsilon^{-1})\right)$.*

*For the $\ell_1$ setup, the proximal update* (3.2) *with uncertainty set* (2.4) *can be approximated up to $\epsilon$ in a number of arithmetic operations of $O\left(AS\log^2(\epsilon^{-1})\right)$.*

**Tractable updates for KL uncertainty.** As in the case of ellipsoidal uncertainty, we present both $\ell_1$ and $\ell_2$ setups for KL uncertainty. The setups are exactly the same as for ellipsoidal uncertainty (i.e. same norms, distance functions, and Bregman divergences), and all constants remain the same. The reason that all constants remain the same is because our bounds on the maximum norms and $\Theta_Y$, for both uncertainty set types, are based on bounding these values over the bigger set consisting of the Cartesian product of $A$ simplexes. The question thus becomes whether (3.2) can be computed efficiently (for $y$) when $D_Y$ is the sum over KL divergences on each action. We present our results in the following proposition.

**Proposition 3.3.** *For the $\ell_2$ setup, the proximal update* (3.2) *with uncertainty set* (2.3) *can be approximated up to $\epsilon$ in a number of arithmetic operations in $O\left(AS\log^2(\epsilon^{-1})\right)$.*

*For the $\ell_1$ setup, the proximal update* (3.2) *with uncertainty set* (2.3) *can be approximated up to $\epsilon$ in a number of arithmetic operations in $O\left(AS\log(\epsilon^{-1})\right)$.*

**Remark 3.4.** At a cursory reading, our results in Propositions 3.2 and 3.3 may seem similar to those of (Nilim and Ghaoui 2005) and (Iyengar 2005). Both authors introduce bisection algorithms for computing Bellman updates, but these are for the simpler case of $(s,a)$-rectangular uncertainty sets. In that case, the Bellman updates can be computed by enumerating the set of actions $a \in \mathbb{A}$, since an optimal solution exists among the set of pure actions. In contrast, in our setting the optimal $x \in \Delta(A)$ may require randomization, which is why we must solve a min-max problem as in (2.1).

## Complexity of Algorithm 1

Armed with our various proximal setups, we can finally state the performance guarantees provided by FOM-VI explicitly for the various setups. Since the constants for the $\ell_1$ and $\ell_2$ setups are the same for both KL and ellipsoidal uncertainty sets, we start by stating a single theorem which gives a bound on the error after $T$ iterations for either type of uncertainty set. The following theorem works for any polynomial scheme for choosing the iterate weights when averaging, as well as how many FOM iterations to perform in-between each VI update.

**Theorem 3.5.** *Let $p, q \in \mathbb{N}$ and at time step $t \geq 0$, let the iterate weight be $\omega_t = t^p$, and the number of FOM iterations at epoch $\ell$ be $T_\ell = \ell^q$. After $T$ iterations of Algorithm 1, $\max_{\boldsymbol{y}\in\mathbb{P}_s} F^{\bar{\boldsymbol{x}}^T,\boldsymbol{y}}(\boldsymbol{v}^*)_s - \min_{\boldsymbol{x}\in\Delta(A)} F^{\boldsymbol{x},\bar{\boldsymbol{y}}^T}(\boldsymbol{v}^*)_s$ is upper bounded by*

- $O\left(A^2\sqrt{\frac{\log(S)}{\log(A)}}\left(\frac{1}{T^{q/(q+1)}} + \frac{\lambda^{T^{1/(q+1)}}}{T^{1/(q+1)}}\right)\right)$ *in the $\ell_1$ setup,*
- $O\left(AS\left(\frac{1}{T^{q/(q+1)}} + \frac{\lambda^{T^{1/(q+1)}}}{T^{1/(q+1)}}\right)\right)$ *in the $\ell_2$ setup.*

The careful reader may notice that the choice of $p \in \mathbb{N}$ in our polynomial averaging scheme does not figure in the bound of Theorem 3.5: any valid choice of $p$ leads to the same bound. However, in practice the choice of $p$ turns out to be very important as we shall see later. Secondly, the reader may notice an interesting dependence on $q$: the

term $O(1/T^{q/(q+1)})$ gets better as $q$ increases; while larger $q$ worsens the exponential rate with base $\lambda$ in the term $O(\lambda^{T^{1/(q+1)}}/T^{1/(q+1)})$. For any fixed $q$, the dominant term is $O(1/T^{q/(q+1)})$.

**Complexity for ellipsoidal uncertainty sets.** We will now combine Proposition 3.2, which gives the cost per iteration of FOM-VI, with Theorem 3.5, to get a total complexity of FOM-VI when considering both the number of iterations and cost per iteration.

First, let us consider $q = 2$, which is the setup we will focus on in our experiments. The complexity of the $\ell_1$ setup is $O\left(A^4S^2\left(\frac{\log(S)}{\log(A)}\right)^{0.75}\log^2(\epsilon^{-1})\epsilon^{-1.5}\right)$ and for the $\ell_2$ setup it is $O\left(A^{2.5}S^{3.5}\log(S)\log(\epsilon^{-1})\epsilon^{-1.5}\right)$. These results are better than the complexity of VI in terms of the number of states and actions. This comes at the cost of the dependence on the desired accuracy $\epsilon$, which is worse than for VI. This is of course expected when applying a first-order method rather than IPMs. However, in practice we expect that our algorithms will be preferable when solving problems with large $A$ and $S$, as is often the case with first-order methods. Indeed, we find numerically that this occurs for $S, A \geq 50$ on ellipsoidal uncertainty sets (see Section 4).

Next, let us consider what happens as $q$ gets large. In that case, the complexity of the $\ell_1$ setup approaches $O\left(A^3S^2\left(\log(S)/\log(A)\right)^{0.5}\log^2(\epsilon^{-1})\epsilon^{-1}\right)$, while the complexity of the $\ell_2$ setup approaches $O\left(A^2S^3\log(S)\log(\epsilon^{-1})\epsilon^{-1}\right)$. This last complexity result is $O(A^{1.5}S^{1.5})$ better than the VI complexity (2.5) in terms of instance size.

Next let us compare the $\ell_1$ and $\ell_2$ setups. When $S = A$, the $\ell_2$ and $\ell_1$ setup have better dependence on number of states and actions than VI (by 2 order of magnitudes). If the number of actions $A$ is considered a constant, then the $\ell_1$ has better convergence guarantees than the $\ell_2$ setup. However, each proximal update in the $\ell_1$ setup requires two interwoven binary searches over Lagrange multipliers, which can prove time-consuming in practice, as we show in our numerical experiments.

**Complexity for KL uncertainty sets.** Similarly to ellipsoidal uncertainty sets, we can analyze our performance on KL uncertainty sets. Again we combine Proposition 3.3 with Theorem 3.5. For $q = 2$, the $\ell_1$ setup has complexity $O\left(A^4S^2\left(\frac{\log(S)}{\log(A)}\right)^{0.75}\log(\epsilon^{-1})\epsilon^{-1.5}\right)$ for returning an $\epsilon$-optimal solution, while the $\ell_2$ setup has complexity $O\left(A^{2.5}S^{3.5}\log(\epsilon^{-1})\epsilon^{-1.5}\right)$. For large $q$, the complexity approaches $O\left(A^3S^2\left(\frac{\log(S)}{\log(A)}\right)^{0.5}\log(\epsilon^{-1})\epsilon^{-1}\right)$ for the $\ell_1$ setup and $O\left(A^2S^3\log(\epsilon^{-1})\epsilon^{-1}\right)$ for the $\ell_2$ setup. To the best of our knowledge, this is the first algorithmic result for $s$-rectangular KL uncertainty sets.

Finally, note that in terms of storage complexity, all our setups only need to store the current value vector $\boldsymbol{v}^\ell \in \mathbb{R}^S$ and the running weighted average $(\bar{\boldsymbol{x}}^\ell, \bar{\boldsymbol{y}}^\ell)$ of the iterates. In total, we need to store $O\left(S^2A\right)$ coefficients, which is the same as the number of decision variables of a solution.

# 4 Numerical Experiments

In this section we study the performance of our approach numerically. We focus here on ellipsoidal uncertainty sets, where we can compare our methods to Value Iteration. We present results for KL uncertainty sets in the appendix.

**Duality gap in the robust MDP problem.** For a given policy-kernel pair $(\boldsymbol{x}, \boldsymbol{y})$, we measure the performance as the duality gap (DG) $= \max_{\boldsymbol{y'}\in\mathbb{P}} R(\boldsymbol{x}, \boldsymbol{y'}) - \min_{\boldsymbol{x'}\in\Pi} R(\boldsymbol{x'}, \boldsymbol{y})$. Note that (DG)$\leq \epsilon$ implies that $\boldsymbol{x}$ is $2\epsilon$-optimal in the robust MDP problem.

**Best empirical setup of Algorithm 1.** For the sake of conciseness, our extensive comparisons of the various proximal setups and parameter choices ($p, q \in \mathbb{N}$) are presented in the appendix.Here we focus on the conclusions. The proximal setup with the best empirical performance is the $\ell_2$ setup where $(\|\cdot\|_X, \|\cdot\|_Y) = (\ell_2, \ell_2)$, even though its theoretical guarantees may be worse than the $\ell_1$ setup (for large state space); this is similar to the matrix-game setting (Gao, Kroer, and Goldfarb 2019). For averaging the PD iterates, an increasing weight scheme, i.e. $p \geq 1$ in $\omega_t = t^p$, is clearly stronger (this is again similar to the matrix-game setting). We also recommend setting $q = 2$ (or even larger), as this leads to better empirical performance for the true duality gap (DG) in the settings where we could compute that duality gap.

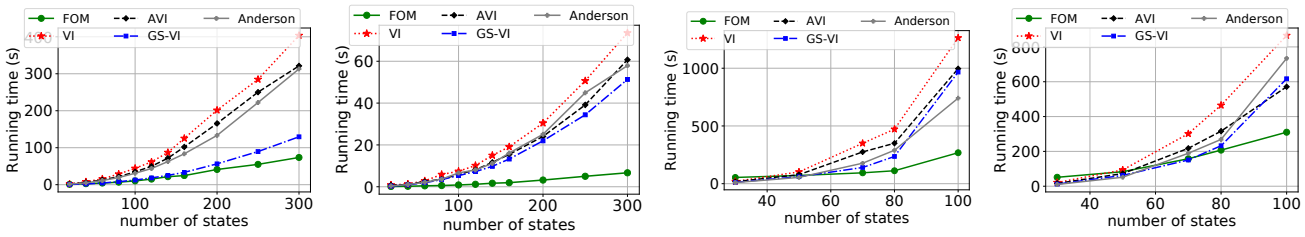
## Comparison with Value Iteration

We present our comparisons with the VI algorithm in Figures 1a-1d. We also compare FOM-VI with Gauss-Seidel VI (*GS-VI*, (Puterman 1994)), Anderson VI (*Anderson*, (Geist and Scherrer 2018)), and *Accelerated VI* (*AVI*, Goyal and Grand-Clement (2019)).The y-axis shows the number of seconds it takes each algorithm to compute an $\epsilon$-optimal policy, for $\epsilon = 0.1$. Following our analysis of the various setups for our algorithm, these plots focus on the $\ell_2$ setup with $(p, q) = (2, 2)$.

**Empirical setup.** All simulations are implemented in Python 3.7.3, and performed on a laptop with 2.2 GHz Intel Core i7 and 8 GB of RAM. We use Gurobi 8.1.1 to solve any linear or quadratic optimization problems involved. In order to obtain an $\epsilon$-solution of the robust MDP problem with the VI algorithms, we use the stopping condition $\|\boldsymbol{v}_{s+1} - \boldsymbol{v}_s\|_\infty \leq \epsilon \cdot (1-\lambda) \cdot (2\lambda)^{-1}$ (Chapter 6.3 in Puterman (1994)). We stop Algorithm 1 as soon as (DG) $\leq \epsilon/2$. We initialize the algorithms with $\boldsymbol{v}_0 = \boldsymbol{0}$. At epoch $\ell$ of Value, AVI and Anderson, we warm-start each computation of $F(\boldsymbol{v}^\ell)$ with the optimal solution obtained from the previous epoch $\ell - 1$.

We consider two type of instances for our simulation. The first type of instances is inspired from real-life application and consists of a healthcare management instance and a machine replacement instance. The second type is based on random *Garnet* MDPs, a class of random MDP instances widely used for benchmarking algorithms.

**Results for healthcare instances.** We consider an MDP instance inspired from a healthcare application. We model the evolution of a patient's health using a Markov chain, using a




(a) Healthcare instance. (b) Machine replacement instance. (c) Garnet, high connectivity. (d) Garnet, low connectivity.

Figure 1: Comparison of FOM-VI with variants of Value Iteration on four MDP domains.

simplification of the models used in Goh et al. (2018); Grand-Clement et al. (2020). Note that such a model is prone to errors as (i) the Markovian assumption is only an approximation of the true dynamics of the patient's health, (ii) the presence of unobservable confounders may introduce biases in our observed transitions. Therefore, it is important to account for model mispecification in this setting. More specifically, we consider an MDP where there are $S-1$ health states, one '*mortality*' state and three actions (drug level), corresponding to high, medium and low drug levels. The state 1 corresponds to a healthy condition while the state $S-1$ is more likely to lead to mortality. The goal of the decision maker is to prescribe a given drug dosage (low/high/medium) at every state, in order to keep the patient alive (avoiding the mortality state), while minimizing the invasiveness of the treatment. We observe $N = 60$ samples around the nominal kernel transitions, presented in the appendices, and we construct ellipsoidal uncertainty sets with radius $\alpha = \sqrt{SA}$. Figure 1a shows the results, where our algorithm outperforms VI by about one order of magnitude on this structured and simple MDP instance, even though GS-VI performs well better than VI too. Additionally, our algorithm scales much better with instance size.

**Results for machine replacement problems** We also consider a machine replacement problem studied by Delage and Mannor (2010) and Wiesemann, Kuhn, and Rustem (2013). The problem 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. Even though the transition parameters can be estimated from historical data, one often does not have access to enough historical data to exactly assess the probability of a machine breaking down when in a given condition. Additionally, the historical data may contain errors; this warrants the use of a robust model for finding a good replacement policy. In particular, the machine replacement problem involves a machine whose set of possible conditions are described by $S$ states. There are two actions: *repair* and *no repair*. 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. 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 last two states $S-1$ and $S$ are states representing when the machine is being repaired. 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. The initial distribution is uniform across states. Figures describing the MDP can be found in the appendix.On this instance, FOM-VI clearly outperforms every variants of VI, as seen on Figure 1b.

**Random Garnet MDP instances.** We generate Garnet MDPs (Generalized Average Reward Non-stationary Environment Test-bench, Archibald, McKinnon, and Thomas (1995); Bhatnagar et al. (2007)), which are an abstract class of MDPs parametrized by a branching factor $n_{branch}$, equal to the proportion of reachable next states from each state-action pair $(s, a)$. Garnet MDPs are a popular class of finite MDPs used for benchmarking algorithms for MDPs (Tarbouriech and Lazaric 2019; Piot, Geist, and Pietquin 2016; Jian et al. 2019). The parameter $n_{branch}$ controls the level of connectivity of the underlying Markov chains. We test our algorithm for high connectivity ($n_{branch} = 50\%$, Figure 1c) and low connectivity ($n_{branch} = 20\%$, Figure 1d) in our simulations. We draw the cost parameters at random uniformly in $[0, 10]$ and we fix a discount factor $\lambda = 0.8$. The radius $\alpha$ of the $\ell_2$ ball from the uncertainty set (2.4) is set to $\alpha = \sqrt{n_{branch} \times A}$.

In Figures 1c-1d, we note that for smaller instances, the performance of FOM-VI is similar to both VI, AVI, GS-VI and Anderson. This is expected: our algorithm has worse convergence guarantees in terms of the dependence in $\epsilon$, but better guarantees in terms of the number of state-actions $S, A$. When the number of states and actions grows larger, FOM-VI performs significantly better than the three other methods.

## 5 Future Works

Our work introduces a novel first-order framework for solving robust MDP, with better theoretical convergence guarantees and significantly better empirical performances than state-of-the-art value iteration algorithms. There are several interesting future directions. For instance, it is important to design new algorithms based on other first-order methods, e.g. mirror descent, mirror prox or online gradient descent. It is also possible to find novel tractable proximal setups beyond KL divergence and $\ell_2$ balls. Finally, extensions to *distributionally* robust settings can lead to tractable algorithms for finding less pessimistic solutions.

## References

Archibald, T.; McKinnon, K.; and Thomas, L. 1995. On the generation of Markov decision processes. *Journal of the*

*Operational Research Society* 46(3): 354–361.

Beck, A. 2017. *First-order methods in optimization*. SIAM.

Ben-Tal, A.; and Nemirovski, A. 2000. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming* 88(3): 411–424.

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

Bertsekas, D. 2007. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific.

Bertsimas, D.; den Hertog, D.; and Pauphilet, J. 2019. Probabilistic guarantees in Robust Optimization .

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

Chambolle, A.; and Pock, T. 2011. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision* 40(1): 120–145.

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

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

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

Gao, Y.; Kroer, C.; and Goldfarb, D. 2019. Increasing Iterate Averaging for Solving Saddle-Point Problems. *arXiv preprint arXiv:1903.10646* .

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

Geist, M.; Scherrer, B.; and Pietquin, O. 2019. A theory of regularized Markov decision processes. *arXiv preprint arXiv:1901.11275* .

Givan, R.; Leach, S.; and Dean, T. 1997. Bounded parameter Markov decision processes. In *European Conference on Planning*, 234–246. Springer.

Goh, J.; Bayati, M.; Zenios, S. A.; Singh, S.; and Moore, D. 2018. Data uncertainty in Markov chains: Application to cost-effectiveness analyses of medical innovations. *Operations Research* 66(3): 697–715.

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

Goyal, V.; and Grand-Clement, J. 2019. A First-Order Approach To Accelerated Value Iteration. *arXiv preprint arXiv:1905.09963* .

Grand-Clement, J.; Chan, C. W.; Goyal, V.; and Escobar, G. 2020. Robust Policies For Proactive ICU Transfers. *arXiv preprint arXiv:2002.06247* .

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

Hu, Z.; and Hong, L. J. 2013. Kullback-Leibler divergence constrained distributionally robust optimization. *Available at Optimization Online* .

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

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

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

Lobo, M. S.; Vandenberghe, L.; Boyd, S.; and Lebret, H. 1998. Applications of second-order cone programming. *Linear algebra and its applications* 284(1-3): 193–228.

Nemirovski, A. 2004. 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.

Nemirovski, A.; and Yudin, D. 1983. *Problem complexity and method efficiency in optimization.*

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

Petrik, M. 2010. *Optimization-based approximate dynamic programming*. Ph.D. thesis, University of Massachusetts Amherst.

Piot, B.; Geist, M.; and Pietquin, O. 2016. Difference of Convex Functions Programming Applied to Control with Expert Data. *arXiv preprint arXiv:1606.01128* .

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

Scherrer, B.; Ghavamzadeh, M.; Gabillon, V.; Lesner, B.; and Geist, M. 2015. Approximate modified policy iteration and its application to the game of Tetris. *Journal of Machine Learning Research* 16(49): 1629–1676.

Steimle, L. N.; and Denton, B. T. 2017. Markov decision processes for screening and treatment of chronic diseases. In *Markov Decision Processes in Practice*, 189–222. Springer.

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

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

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

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