Journal of Machine Learning Research 22 (2021) 1-46 Submitted 5/20; Revised 3/21; Published 10/21

Partial Policy Iteration for L_1-Robust Markov Decision Processes

Chin Pang Ho CLINT.HO@CITYU.EDU.HK
School of Data Science
City University of Hong Kong
83 Tat Chee Avenue
Kowloon Tong, Hong Kong

Marek Petrik MPETRIK@CS.UNH.EDU
Department of Computer Science
University of New Hampshire
Durham, NH, USA, 03861

Wolfram Wiesemann WW@IMPERIAL.AC.UK
Imperial College Business School
Imperial College London
London SW7 2AZ, United Kingdom

Editor: Ambuj Tewari

Abstract

Robust Markov decision processes (MDPs) compute reliable solutions for dynamic decision problems with partially-known transition probabilities. Unfortunately, accounting for uncertainty in the transition probabilities significantly increases the computational complexity of solving robust MDPs, which limits their scalability. This paper describes new, efficient algorithms for solving the common class of robust MDPs with s- and sa-rectangular ambiguity sets defined by weighted L_1 norms. We propose partial policy iteration, a new, efficient, flexible, and general policy iteration scheme for robust MDPs. We also propose fast methods for computing the robust Bellman operator in quasi-linear time, nearly matching the ordinary Bellman operator's linear complexity. Our experimental results indicate that the proposed methods are many orders of magnitude faster than the state-of-the-art approach, which uses linear programming solvers combined with a robust value iteration.

Keywords: Robust Markov decision processes, optimization, reinforcement learning

1. Introduction

Markov decision processes (MDPs) provide a versatile methodology for modeling and solving dynamic decision problems under uncertainty (Puterman, 2005). Unfortunately, however, MDP solutions can be very sensitive to estimation errors in the transition probabilities and rewards. This is of particular worry in *reinforcement learning* applications, where the model is fit to data and therefore inherently uncertain. Robust MDPs (RMDPs) do not assume that the transition probabilities are known precisely but instead allow them to take on any value from a given *ambiguity set* or uncertainty set (Xu and Mannor, 2006; Mannor et al., 2012; Hanasusanto and Kuhn, 2013; Tamar et al., 2014; Delgado et al., 2016). With appropriately chosen ambiguity sets, RMDP solutions are often much less sensitive to model errors (Xu and Mannor, 2009; Petrik, 2012; Petrik et al., 2016).

Most of the RMDP literature assumes *rectangular* ambiguity sets that constrain the errors in the transition probabilities independently for each state (Iyengar, 2005; Nilim and El Ghaoui, 2005; Le Tallec, 2007; Kaufman and Schaefer, 2013; Wiesemann et al., 2013). This assumption is crucial to retain many of the desired structural features of MDPs. In particular, the robust return of an RMDP with a rectangular ambiguity set is maximized by a *stationary policy*, and the optimal value function satisfies a robust variant of the Bellman optimality equation. Rectangularity also ensures that an optimal policy can be computed in *polynomial time* by robust versions of the classical value or policy iteration (Iyengar, 2005; Hansen et al., 2013).

A particularly popular class of rectangular ambiguity sets is defined by bounding the L_1-distance of any plausible transition probabilities from a *nominal* distribution (Iyengar, 2005; Strehl et al., 2009; Jaksch et al., 2010; Petrik and Subramanian, 2014; Taleghan et al., 2015; Petrik et al., 2016). Such ambiguity sets can be readily constructed from samples (Weissman et al., 2003; Behzadian et al., 2021), and their polyhedral structure implies that the worst transition probabilities can be computed by the solution of linear programs (LPs). Unfortunately, even for the specific class of L_1-ambiguity sets, an LP has to be solved for each state and each step of the value or policy iteration. Generic LP algorithms have a worst-case complexity that is approximately quartic in the number of states (Vanderbei, 1998), and they thus become prohibitively expensive for large RMDPs.

In this paper, we propose a new framework for solving RMDPs. Our framework applies to both sa-rectangular ambiguity sets, where adversarial nature observes the agent's actions before choosing the worst plausible transition probabilities (Iyengar, 2005; Nilim and El Ghaoui, 2005), and s-rectangular ambiguity sets, where nature must commit to a realization of the transition probabilities before observing the agent's actions (Le Tallec, 2007; Wiesemann et al., 2013). We achieve a significant theoretical and practical acceleration over the robust value and policy iteration by reducing the number of iterations needed and by reducing the computational complexity of each iteration. The overall speedup of our framework—both theoretical and practical—allows us to solve RMDPs with L_1-ambiguity sets in a time complexity that is similar to that of classical MDPs. Our framework comprises of three components, each of which represents a novel contribution.

Our first contribution is *partial policy iteration* (PPI), which generalizes the classical modified policy iteration to RMDPs. PPI resembles the robust modified policy iteration (Kaufman and Schaefer, 2013), which has been proposed for sa-rectangular ambiguity sets. In contrast to the robust modified policy iteration, however, PPI applies to both sa-rectangular and s-rectangular ambiguity sets, and it is guaranteed to converge at the same linear rate as robust value and robust policy iteration. In our experimental results, PPI outperforms robust value iteration by several orders of magnitude.

Our second contribution is a fast algorithm for computing the robust Bellman operator for sa-rectangular weighted L_1-ambiguity sets. Our algorithm employs the *homotopy continuation* strategy (Vanderbei, 1998): it starts with a singleton ambiguity set for which the worst transition probabilities can be trivially identified, and it subsequently traces the most adverse transition probabilities as the size of the ambiguity set increases. The time complexity of our homotopy method is quasi-linear in the number of states and actions, which is significantly faster than the quartic worst-case complexity of generic LP solvers.

Our third contribution is a fast algorithm for computing the robust Bellman operator for s-rectangular weighted L_1-ambiguity sets. While often less conservative and hence more appropriate in practice, s-rectangular ambiguity sets are computationally challenging since the agent's optimal policy can be randomized (Wiesemann et al., 2013). We propose a *bisection* approach to decompose the s-rectangular Bellman computation into a series of sa-rectangular Bellman computations. When our bisection method is combined with our homotopy method, its time complexity is quasi-linear in the number of states and actions, compared again to the quartic complexity of generic LP solvers.

Put together, our contributions comprise a complete framework that can be used to solve RMDPs efficiently. Besides being faster than solving LPs directly, our framework does not require an expensive black-box commercial optimization package such as CPLEX, Gurobi, or Mosek. A well-tested and documented implementation of the methods described in this paper is available at `https://github.com/marekpetrik/craam2`. Compared to an earlier conference version of this work (Ho et al., 2018), the present paper introduces PPI, it improves the bisection method to work with PPI, it provides extensive and simpler proofs, and it reports more complete experimental results.

The remainder of the paper is organized as follows. We summarize relevant prior work in Section 2 and subsequently review basic properties of RMDPs in Section 3. Section 4 describes our partial policy iteration (PPI), Section 5 develops the homotopy method for sa-rectangular ambiguity sets, and Section 6 is devoted to the bisection method for s-rectangular ambiguity sets. Section 7 compares our algorithms with the solution of RMDPs via Gurobi, a leading commercial LP solver, and we offer concluding remarks in Section 8.

We use the following notation throughout the paper. Regular lowercase letters (such as p) denote scalars, boldface lowercase letters (such as $\boldsymbol{p}$) denote vectors, and boldface uppercase letters (such as $\boldsymbol{X}$) denote matrices. Indexed values are printed in bold if they are vectors and in regular font if they are scalars. That is, p_i refers to the i-th component of a vector $\boldsymbol{p}$, whereas $\boldsymbol{p}_i$ is the i-th vector of a sequence of vectors. An expression in parentheses indexed by a set of natural numbers, such as $(p_i)_{i\in\mathcal{Z}}$ for $\mathcal{Z} = \{1, \dots, k\}$, denotes the vector $(p_1, p_2, \dots, p_k)$. Similarly, if each $\boldsymbol{p}_i$ is a vector, then $\boldsymbol{P} = (\boldsymbol{p}_i)_{i\in\mathcal{Z}}$ is a matrix with each vector $\boldsymbol{p}_i^\top$ as a row. The expression $(\boldsymbol{p}_i)_j \in \mathbb{R}$ represents the component in i-th row and j-th column. All vector inequalities are understood to hold component-wise. Calligraphic letters and uppercase Greek letters (such as $\mathcal{X}$ and Ξ) are reserved for sets. The symbols $\mathbf{1}$ and $\mathbf{0}$ denote vectors of all ones and all zeros, respectively, of the size appropriate to their context. The symbol $\boldsymbol{I}$ denotes the identity matrix of the appropriate size. The probability simplex in $\mathbb{R}^S_+$ is denoted as $\Delta^S = \left\{\boldsymbol{p} \in \mathbb{R}^S_+ \mid \mathbf{1}^\top \boldsymbol{p} = 1\right\}$. The set $\mathbb{R}$ represents real numbers and the set $\mathbb{R}_+$ represents non-negative real numbers.

2. Related Work

We review relevant prior work that aims at (i) reducing the number of iterations needed to compute an optimal RMDP policy, as well as (ii) reducing the computational complexity of each iteration. We also survey algorithms for related machine learning problems.

The standard approach for computing an optimal RMDP policy is *robust value iteration*, which is a variant of the classical value iteration for ordinary MDPs that iteratively applies the robust Bellman operator to an increasingly accurate approximation of the optimal robust value function (Givan et al., 2000; Iyengar, 2005; Le Tallec, 2007; Wiesemann et al., 2013). Robust value iteration is easy to implement and versatile, and it converges linearly with a rate of γ, the discount factor of the RMDP.

Unfortunately, robust value iteration requires many iterations and thus performs poorly when the discount factor of the RMDP approaches 1. To alleviate this issue, *robust policy iteration* alternates between robust policy evaluation steps that determine the robust value function for a fixed policy and policy improvement steps that select the optimal greedy policy for the current estimate of the robust value function (Iyengar, 2005; Hansen et al., 2013). While the theoretical convergence rate guarantee for robust policy iteration matches that for robust value iteration, its practical performance tends to be superior for discount factors close to 1. However, unlike the classical policy iteration for ordinary MDPs, which solves a system of linear equations in each policy evaluation step, robust policy iteration solves a large LP in each robust policy evaluation step. The need to solve large LPs restricts robust policy iteration to small RMDPs.

Modified policy iteration, also known as optimistic policy iteration, tends to significantly outperform both value and policy iteration on ordinary MDPs (Puterman, 2005). Modified policy iteration adopts the same strategy as policy iteration, but it merely approximates the value function in each policy evaluation step by executing a small number of value iterations. Generalizing the modified policy iteration to RMDPs is not straightforward. There were several early attempts to develop a robust modified policy iteration (Satia and Lave, 1973; White and Eldeib, 1994), but their convergence guarantees are in doubt (Kaufman and Schaefer, 2013). The challenge is that the alternating maximization (in the policy improvement step) and minimization (in the policy evaluation step) may lead to infinite cycles in the presence of approximation errors. Several natural robust policy iteration variants have been shown to loop infinitely on some inputs (Condon, 1993).

To the best of our knowledge, *robust modified policy iteration* (RMPI) is the first generalization of the classical modified policy iteration to RMDPs with provable convergence guarantees (Kaufman and Schaefer, 2013). RMPI alternates between robust policy evaluation steps and policy improvement steps. The robust policy evaluation steps approximate the robust value function of a fixed policy by executing a small number of value iterations, and the policy improvement steps select the optimal greedy policy for the current estimate of the robust value function. Our partial policy iteration (PPI) improves on RMPI in several respects. RMPI only applies to sa-rectangular problems in which there exist optimal deterministic policies, while PPI also applies to s-rectangular problems in which all optimal policies may be randomized. Also, RMPI relies on value iteration to partially evaluate a fixed policy, whereas PPI can evaluate the fixed policy more efficiently using other schemes such as policy or modified policy iteration. Finally, PPI enjoys a guaranteed linear convergence rate of γ, the discount factor of the RMDP.

Besides value and (modified) policy iteration, ordinary MDPs have been successfully solved with accelerated value iteration, which reduces the number of required Bellman operator evaluations. Recent accelerated value iteration methods have employed Anderson and Nesterov-type acceleration approaches (Geist and Scherrer, 2018; Goyal and Grand-

Clement, 2019; Zhang et al., 2020). These acceleration schemes eschew policy evaluation in lieu of computing a linear combination of recent value function iterates. However, it is unclear how these accelerated value iteration approaches can be generalized to RMDPs.

In addition to accelerating value iteration, prior work has also focused on speeding up the computation of the robust Bellman operator for structured ambiguity sets. While this evaluation amounts to the solution of a convex optimization problem for generic convex ambiguity sets and reduces to the solution of an LP for polyhedral ambiguity sets, the resulting polynomial runtime guarantees are insufficient due to the large number of evaluations required. Quasi-linear time algorithms for computing Bellman updates for RMDPs with *unweighted* sa-rectangular L_1-ambiguity sets have been proposed by Iyengar (2005) and Petrik and Subramanian (2014). Similar algorithms have been used to guide the exploration of MDPs (Strehl et al., 2009; Taleghan et al., 2015). In contrast, our algorithm for sa-rectangular ambiguity sets applies to both unweighted and weighted L_1-ambiguity sets, where the latter ones have been shown to provide superior robustness guarantees (Behzadian et al., 2021). The extension to weighted norms requires a surprisingly large change to the algorithm. Quasi-linear time algorithms have also been proposed for sa-rectangular L_∞-ambiguity sets (Givan et al., 2000), L_2-ambiguity sets (Iyengar, 2005), and KL-ambiguity sets (Iyengar, 2005; Nilim and El Ghaoui, 2005). We are not aware of any previous specialized algorithms for s-rectangular ambiguity sets, which are significantly more challenging as all optimal policies may be randomized, and it is therefore not possible to compute the worst transition probabilities independently for each action.

Our algorithm for computing the robust Bellman operator over an sa-rectangular ambiguity set resembles LARS, a homotopy method for solving the LASSO problem (Drori and Donoho, 2006; Hastie et al., 2009; Murphy, 2012). It also resembles methods for computing fast projections onto the L_1-ball (Duchi et al., 2008; Thai et al., 2015) and the weighted L_1-ball (van den Berg and Friedlander, 2011). In contrast to those works, our algorithm optimizes a linear function (instead of a more general quadratic one) over the intersection of the (weighted) L_1-ball and the probability simplex (as opposed to the entire L_1-ball).

Our algorithm for computing the robust Bellman operator for s-rectangular ambiguity sets employs a bisection method. This is a common optimization technique for solving low-dimensional problems. We are not aware of works that use bisection to solve s-rectangular RMDPs or similar machine learning problems. However, a bisection method has been previously used to solve sa-rectangular RMDPs with KL-ambiguity sets (Nilim and El Ghaoui, 2005). That bisection method, however, has a different motivation, solves a different problem, and bisects on different problem parameters.

Throughout this paper, we focus on RMDPs with sa-rectangular or s-rectangular ambiguity sets but note that several more-general classes have been proposed recently (Mannor et al., 2012, 2016; Goyal and Grand-Clement, 2018). These k-rectangular and r-rectangular sets have tangible advantages, but also introduce additional computational complications.

3. Robust Markov Decision Processes

This section surveys RMDPs and their basic properties. We cover both sa-rectangular and s-rectangular ambiguity sets but limit the discussion to norm-constrained ambiguity sets.

An MDP $(\mathcal{S}, \mathcal{A}, \boldsymbol{p}_0, \boldsymbol{p}, \boldsymbol{r}, \gamma)$ is described by a state set $\mathcal{S} = \{1, \ldots, S\}$ and an action set $\mathcal{A} = \{1, \ldots, A\}$. The initial state is selected randomly according to the distribution $\boldsymbol{p}_0 \in \Delta^S$. When the MDP is in state $s \in \mathcal{S}$, taking the action $a \in \mathcal{A}$ results in a stochastic transition to a new state $s' \in \mathcal{S}$ according to the distribution $\boldsymbol{p}_{s,a} \in \Delta^S$ with a reward of $r_{s,a,s'} \in \mathbb{R}$. We condense the transition probabilities $\boldsymbol{p}_{s,a}$ to the transition function $\boldsymbol{p} = (\boldsymbol{p}_{s,a})_{s\in\mathcal{S},a\in\mathcal{A}} \in (\Delta^S)^{S\times A}$ which can also be also interpreted as a function $\boldsymbol{p} : \mathcal{S} \times \mathcal{A} \to \Delta^S$. Similarly, we condense the rewards to vectors $\boldsymbol{r}_{s,a} = (r_{s,a,s'})_{s'\in\mathcal{S}} \in \mathbb{R}^S$ and $\boldsymbol{r} = (\boldsymbol{r}_{s,a})_{s\in\mathcal{S},a\in\mathcal{A}}$. The discount factor is $\gamma \in (0, 1)$.

A (stationary) randomized policy $\boldsymbol{\pi} = (\boldsymbol{\pi}_s)_{s\in\mathcal{S}}$, $\boldsymbol{\pi}_s \in \Delta^A$ for all $s \in \mathcal{S}$, is a function that prescribes to take an action $a \in \mathcal{A}$ with the probability $\pi_{s,a}$ whenever the MDP is in a state $s \in \mathcal{S}$. We use $\Pi = (\Delta^A)^S$ to denote the set of all randomized stationary policies.

For a given policy $\boldsymbol{\pi} \in \Pi$, an MDP becomes a *Markov reward process*, which is a Markov chain with the $S \times S$ transition matrix $\boldsymbol{P}(\boldsymbol{\pi}) = (\boldsymbol{p}_s(\boldsymbol{\pi}))_{s\in\mathcal{S}}$ and the rewards $\boldsymbol{r}(\boldsymbol{\pi}) = (r_s(\boldsymbol{\pi}))_{s\in\mathcal{S}} \in \mathbb{R}^S$ where

$$\boldsymbol{p}_s(\boldsymbol{\pi}) = \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_{s,a} \quad \text{and} \quad r_s(\boldsymbol{\pi}) = \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_{s,a}^\top \boldsymbol{r}_{s,a} \ ,$$

and $\boldsymbol{p}_s(\boldsymbol{\pi}) \in \Delta^S$ and $r_s(\boldsymbol{\pi}) \in \mathbb{R}$. The total expected discounted reward of this Markov reward process is

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_{S_t,A_t,S_{t+1}}\right] \ = \ \boldsymbol{p}_0^\top (\boldsymbol{I} - \gamma \cdot \boldsymbol{P}(\boldsymbol{\pi}))^{-1} \boldsymbol{r}(\boldsymbol{\pi}) \ .$$

Here, the initial random state S_0 is distributed according to $\boldsymbol{p}_0$, the subsequent random states $S_1, S_2, \ldots$ are distributed according to $\boldsymbol{p}(\boldsymbol{\pi})$, and the random actions $A_0, A_1, \ldots$ are distributed according to $\boldsymbol{\pi}$. The value function of this Markov reward process is $\boldsymbol{v}(\boldsymbol{\pi}, \boldsymbol{p}) = (\boldsymbol{I} - \gamma \cdot \boldsymbol{P}(\boldsymbol{\pi}))^{-1} \boldsymbol{r}(\boldsymbol{\pi})$. For each state $s \in \mathcal{S}$, $v_s(\boldsymbol{\pi}, \boldsymbol{p})$ describes the total expected discounted reward once the Markov reward process enters s. It is well-known that the total expected discounted reward of an MDP is optimized by a deterministic policy $\boldsymbol{\pi}$ satisfying $\pi_{s,a} \in \{0, 1\}$ for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$ (Puterman, 2005).

RMDPs generalize MDPs in that they account for the uncertainty in the transition function $\boldsymbol{p}$. More specifically, the RMDP $(\mathcal{S}, \mathcal{A}, \boldsymbol{p}_0, \mathcal{P}, \boldsymbol{r}, \gamma)$ assumes that the transition function $\boldsymbol{p}$ is chosen adversarially from an *ambiguity set* (or *uncertainty set*) of plausible values $\mathcal{P} \subseteq (\Delta^S)^{S\times A}$ (Hanasusanto and Kuhn, 2013; Wiesemann et al., 2013; Petrik and Subramanian, 2014; Petrik et al., 2016; Russell and Petrik, 2019). The objective is to compute a policy $\boldsymbol{\pi} \in \Pi$ that maximizes the *return*, or the expected sum of discounted rewards, under the worst-case transition function from $\mathcal{P}$:

$$\max_{\boldsymbol{\pi}\in\Pi} \min_{\boldsymbol{p}\in\mathcal{P}} \ \boldsymbol{p}_0^\top \boldsymbol{v}(\boldsymbol{\pi}, \boldsymbol{p}) \ . \tag{3.1}$$

The maximization in (3.1) represents the objective of the agent, while the minimization can be interpreted as the objective of adversarial nature. To ensure that the minimum exists, we assume throughout the paper that the set $\mathcal{P}$ is *compact*.

The optimal policies in RMDPs are history-dependent, stochastic and NP-hard to compute even when restricted to be stationary (Iyengar, 2005; Wiesemann et al., 2013). However, the problem (3.1) is tractable for some broad classes of ambiguity sets $\mathcal{P}$. The most common such class are the *sa-rectangular ambiguity sets*, which are defined as Cartesian products of sets $\mathcal{P}_{s,a} \subseteq \Delta^S$ for each state s and action a (Iyengar, 2005; Nilim and El Ghaoui, 2005; Le Tallec, 2007):

$$\mathcal{P} = \left\{ \boldsymbol{p} \in (\Delta^S)^{S\times A} \;\middle|\; \boldsymbol{p}_{s,a} \in \mathcal{P}_{s,a} \;\; \forall s \in \mathcal{S}, a \in \mathcal{A} \right\} . \tag{3.2}$$

Since each probability vector $\boldsymbol{p}_{s,a}$ belongs to a separate set $\mathcal{P}_{s,a}$, adversarial nature can select the worst transition probabilities independently for each state and action. This amounts to nature being able to observe the agent's action prior to choosing the transition probabilities. Similar to ordinary MDPs, there always exists an optimal deterministic stationary policy in sa-rectangular RMDPs (Iyengar, 2005; Nilim and El Ghaoui, 2005).

In this paper, we study sa-rectangular ambiguity sets that constitute weighted L_1-balls around some *nominal transition probabilities* $\bar{\boldsymbol{p}}_{s,a} \in \Delta^S$:

$$\mathcal{P}_{s,a} = \left\{ \boldsymbol{p} \in \Delta^S \;\middle|\; \|\boldsymbol{p} - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_{s,a} \right\}$$

Here, the weights $\boldsymbol{w}_{s,a} \in \mathbb{R}^S_+$ are assumed to be strictly positive: $w_{s,a} > \boldsymbol{0}, s \in \mathcal{S}, a \in \mathcal{A}$. The radius $\kappa_{s,a} \in \mathbb{R}_+$ of the ball is called the *budget*, and the weighted L_1-norm is defined as

$$\|\boldsymbol{x}\|_{1,\boldsymbol{w}} = \sum_{i=1}^{n} w_i \, |x_i| \; .$$

The weights $\boldsymbol{w}_{s,a}$ can be used to control the shape of the ambiguity sets to compute better policies. For example, RMDPs with optimized weights $\boldsymbol{w}_{s,a}$ provide significantly improved percentile criterion guarantees compared to uniform weights (Behzadian et al., 2021).

L_1-ball ambiguity sets have gained popularity in RMDPs (Iyengar, 2005; Petrik and Subramanian, 2014; Petrik et al., 2016; Derman et al., 2019; Behzadian et al., 2021) and optimistic MDPs (Strehl et al., 2009; Jaksch et al., 2010; Taleghan et al., 2015) for the following reasons. First, the L_1-distance between probability distributions corresponds to the *total variation distance*, a simple and intuitive statistical distance metric. Second, the robust Bellman operator over the L_1-ambiguity set can be solved as a linear program using mature, widely-available solvers. And third, RMDPs combined with existing L_1-norm concentration inequalities can be used to provide finite sample guarantees (Weissman et al., 2003; Petrik et al., 2016; Behzadian et al., 2021).

Similarly to MDPs, the robust value function $\boldsymbol{v_\pi} = \min_{\boldsymbol{p}\in\mathcal{P}} \boldsymbol{v}(\boldsymbol{\pi}, \boldsymbol{p})$ of an sa-rectangular RMDP for a policy $\boldsymbol{\pi} \in \Pi$ can be computed using the *robust Bellman policy update* $\mathfrak{L}_{\boldsymbol{\pi}} : \mathbb{R}^S \to \mathbb{R}^S$. For sa-rectangular RMDPs constrained by the L_1-norm, the operator $\mathfrak{L}_{\boldsymbol{\pi}}$ is defined for each state $s \in \mathcal{S}$ as

$$\begin{aligned} (\mathfrak{L}_{\boldsymbol{\pi}} \boldsymbol{v})_s &= \sum_{a\in\mathcal{A}} \left(\pi_{s,a} \cdot \min_{\boldsymbol{p}\in\mathcal{P}_{s,a}} \boldsymbol{p}^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \right) \\ &= \sum_{a\in\mathcal{A}} \left(\pi_{s,a} \cdot \min_{\boldsymbol{p}\in\Delta^S} \left\{ \boldsymbol{p}^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \;\middle|\; \|\boldsymbol{p} - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_{s,a} \right\} \right) . \end{aligned} \tag{3.3}$$

The robust value function is the unique solution to $\boldsymbol{v_\pi} = \mathfrak{L}_{\boldsymbol{\pi}} \boldsymbol{v_\pi}$ (Iyengar, 2005). To compute the optimal value function, we use the sa-rectangular *robust Bellman optimality operator*

$\mathfrak{L} : \mathbb{R}^S \to \mathbb{R}^S$ defined as

$$
\begin{aligned}
(\mathfrak{L}\boldsymbol{v})_s &= \max_{a \in \mathcal{A}} \min_{\boldsymbol{p} \in \mathcal{P}_{s,a}} \boldsymbol{p}^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \\
&= \max_{a \in \mathcal{A}} \min_{\boldsymbol{p} \in \Delta^S} \left\{ \boldsymbol{p}^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \;\middle|\; \|\boldsymbol{p} - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_{s,a} \right\} .
\end{aligned}
\tag{3.4}
$$

Let $\boldsymbol{\pi}^\star \in \Pi$ be an optimal robust policy which solves (3.1). Then the optimal robust value function $\boldsymbol{v}^\star = \boldsymbol{v}_{\boldsymbol{\pi}^\star}$ is the unique vector that satisfies $\boldsymbol{v}^\star = \mathfrak{L}\boldsymbol{v}^\star$ (Iyengar, 2005; Wiesemann et al., 2013). In addition, a policy $\boldsymbol{\pi}$ is called *greedy* for a value function $\boldsymbol{v}$ whenever $\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{v} = \mathfrak{L}\boldsymbol{v}$.

The value $\boldsymbol{p} \in \Delta^S$ in the equations above represents a probability vector rather than the transition function $\boldsymbol{p} \in (\Delta^S)^{S \times A}$. To prevent confusion between the two in the remainder of the paper, we specify the dimensions of $\boldsymbol{p}$ whenever it is not obvious from its context.

As mentioned above, sa-rectangular sets assume that nature can observe the agent's action when choosing the robust transition probabilities. This assumption grants nature too much power and often results in overly conservative policies (Le Tallec, 2007; Wiesemann et al., 2013). *S-rectangular ambiguity sets* partially alleviate this issue while preserving the computational tractability of sa-rectangular sets. They are defined as Cartesian products of sets $\mathcal{P}_s \subseteq (\Delta^S)^A$ for each state s (as opposed to state-action pairs earlier):

$$
\mathcal{P} = \left\{ \boldsymbol{p} \in (\Delta^S)^{S \times A} \;\middle|\; (\boldsymbol{p}_{s,a})_{a \in \mathcal{A}} \in \mathcal{P}_s \ \forall s \in \mathcal{S} \right\}
\tag{3.5}
$$

Since the probability vectors $\boldsymbol{p}_{s,a}$, $a \in \mathcal{A}$, for the same state s are subjected to the joint constraints captured by $\mathcal{P}_s$, adversarial nature can no longer select the worst transition probabilities independently for each state and action. The presence of these joint constraints amounts to nature choosing the transition probabilities while only observing the state and not the agent's action (but observing the agent's policy). In contrast to ordinary MDPs and sa-rectangular RMDPs, s-rectangular RMDPs are optimized by randomized policies in general (Le Tallec, 2007; Wiesemann et al., 2013). As before, we restrict our attention to s-rectangular ambiguity sets defined in terms of L_1-balls around nominal transition probabilities:

$$
\mathcal{P}_s = \left\{ \boldsymbol{p} \in (\Delta^S)^A \;\middle|\; \sum_{a \in \mathcal{A}} \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_s \right\}
$$

In contrast to the earlier sa-rectangular ambiguity set, nature is now restricted by a single budget $\kappa_s \in \mathbb{R}_+$ for all transition probabilities $(\boldsymbol{p}_{s,a})_{a \in \mathcal{A}}$ relating to a state $s \in \mathcal{S}$. We note that although sa-rectangular ambiguity sets are a special case of s-rectangular ambiguity sets in general (Wiesemann et al., 2013), this is not true for our particular classes of L_1-ball ambiguity sets.

The s-rectangular *robust Bellman policy update* $\mathfrak{L}_{\boldsymbol{\pi}} : \mathbb{R}^S \to \mathbb{R}^S$ is defined as

$$
\begin{aligned}
(\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{v})_s &= \min_{\boldsymbol{p} \in \mathcal{P}_s} \sum_{a \in \mathcal{A}} \left(\pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \right) \\
&= \min_{\boldsymbol{p} \in (\Delta^S)^A} \left\{ \sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \;\middle|\; \sum_{a \in \mathcal{A}} \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_s \right\} .
\end{aligned}
\tag{3.6}
$$

```
**Input:** Tolerances $\epsilon_1, \epsilon_2, \ldots$ such that $\epsilon_{k+1} < \gamma \epsilon_k$ and desired precision $\delta$
**Output:** Policy $\boldsymbol{\pi}_k$ such that $\|\boldsymbol{v}_{\boldsymbol{\pi}_k} - \boldsymbol{v}^\star\|_\infty \leqslant \delta$
$k \leftarrow 0,\ \boldsymbol{v}_0 \leftarrow$ an arbitrary initial value function ;
**repeat**
    $k \leftarrow k + 1$;
    // Policy improvement
    Compute $\tilde{\boldsymbol{v}}_k \leftarrow \mathfrak{L} \boldsymbol{v}_{k-1}$ and choose *greedy* $\boldsymbol{\pi}_k$ such that $\mathfrak{L}_{\boldsymbol{\pi}_k} \boldsymbol{v}_{k-1} = \tilde{\boldsymbol{v}}_k$;
    // Policy evaluation
    With policy $\boldsymbol{\pi}_k$, define a MDP using Definition 4.1 and compute $\boldsymbol{v}_k$ such that
        $\|\mathfrak{L}_{\boldsymbol{\pi}_k} \boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty \leqslant (1-\gamma)\, \epsilon_k$;
**until** $\|\mathfrak{L} \boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty + \|\mathfrak{L}_{\boldsymbol{\pi}_k} \boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty < (1-\gamma)\, \delta$;
**return** $\boldsymbol{\pi}_k$
```

Algorithm 4.1: Partial Policy Iteration (PPI)

As in the sa-rectangular case, the robust value function is the unique solution to $\boldsymbol{v}_{\boldsymbol{\pi}} = \mathfrak{L}_{\boldsymbol{\pi}} \boldsymbol{v}_{\boldsymbol{\pi}}$ (Wiesemann et al., 2013). The s-rectangular *robust Bellman optimality operator* $\mathfrak{L} : \mathbb{R}^S \to \mathbb{R}^S$ is defined as

$$\begin{aligned}(\mathfrak{L}\boldsymbol{v})_s &= \max_{\boldsymbol{d} \in \Delta^A} \min_{\boldsymbol{p} \in \mathcal{P}_s} \sum_{a \in \mathcal{A}} d_a \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \\ &= \max_{\boldsymbol{d} \in \Delta^A} \min_{\boldsymbol{p} \in (\Delta^S)^A} \left\{ \sum_{a \in \mathcal{A}} d_a \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}) \;\middle|\; \sum_{a \in \mathcal{A}} \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_{s,a}\|_{1, \boldsymbol{w}_{s,a}} \leqslant \kappa_s \right\} .\end{aligned} \tag{3.7}$$

The optimal robust value function $\boldsymbol{v}^\star = \boldsymbol{v}_{\boldsymbol{\pi}^\star}$ in an s-rectangular RMDP is also the unique vector that satisfies $\boldsymbol{v}^\star = \mathfrak{L}\boldsymbol{v}^\star$ (Iyengar, 2005; Wiesemann et al., 2013). We use the same symbols $\mathfrak{L}_{\boldsymbol{\pi}}$ and $\mathfrak{L}$ for sa-rectangular and s-rectangular ambiguity sets when their meaning is clear from the context.

4. Partial Policy Iteration

In this section, we describe and analyze a new iterative method for solving RMDPs with sa-rectangular or s-rectangular ambiguity sets which we call *Partial Policy Iteration* (PPI). It resembles standard policy iteration; it evaluates policies only partially before improving them. PPI is the first policy iteration method that provably converges to the optimal solution for s-rectangular RMDPs. We first describe and analyze PPI and then compare it with existing robust policy iteration algorithms.

Algorithm 4.1 provides an outline of PPI. The algorithm follows the familiar pattern of interleaving approximate policy evaluation with policy improvement and thus resembles the modified policy iteration (also known as optimistic policy iteration) for classical, ordinary MDPs (Puterman, 2005). In contrast to classical policy iteration, which always evaluates policies precisely, PPI approximates policy evaluation. This is fast and sufficient, particularly when evaluating highly suboptimal policies.

Notice that by employing the robust Bellman optimality operator $\mathfrak{L}$, the policy improvement step in Algorithm 4.1 selects the updated greedy policy $\boldsymbol{\pi}_k$ in view of the worst transition function from the ambiguity set. Although the robust Bellman optimality operator $\mathfrak{L}$ requires more computational effort than its ordinary MDP counterpart, it is necessary as several variants of PPI that employ an ordinary Bellman optimality operator have been shown to fail to converge to the optimal solution (Condon, 1993).

The policy evaluation step in Algorithm 4.1 is performed by approximately solving the following *robust policy evaluation MDP*, which is similar to the *adversarial MDP* used in the context of r-rectangular RMDPs (Goyal and Grand-Clement, 2018).

Definition 4.1. For an s-rectangular RMDP $(\mathcal{S}, \mathcal{A}, \boldsymbol{p}_0, \mathcal{P}, \boldsymbol{r}, \gamma)$ and a fixed policy $\boldsymbol{\pi} \in \Pi$, we define the *robust policy evaluation MDP* $(\mathcal{S}, \bar{\mathcal{A}}, \boldsymbol{p}_0, \bar{\boldsymbol{p}}, \bar{\boldsymbol{r}}, \gamma)$ as follows. The continuous state-dependent action sets $\bar{\mathcal{A}}(s)$, $s \in \mathcal{S}$, represent nature's choice of the transition probabilities and are defined as $\bar{\mathcal{A}}(s) = \mathcal{P}_s$. Thus, nature's decisions are of the form $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_a)_{a \in \mathcal{A}} \in (\Delta^S)^A$ with $\boldsymbol{\alpha}_a \in \Delta^S$, $a \in \mathcal{A}$. The transition function $\bar{\boldsymbol{p}}$ and the rewards $\bar{\boldsymbol{r}}$ are defined as

$$\bar{\boldsymbol{p}}_{s,\boldsymbol{\alpha}} = \sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \boldsymbol{\alpha}_a \quad \text{and} \quad \bar{r}_{s,\boldsymbol{\alpha}} = -\sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \boldsymbol{\alpha}_a^\top \boldsymbol{r}_{s,a} \ ,$$

where $\bar{\boldsymbol{p}}_{s,\boldsymbol{\alpha}} \in \Delta^S$ and $\bar{r}_{s,\boldsymbol{\alpha}} \in \mathbb{R}$. All other parameters of the robust policy evaluation MDP coincide with those of the RMDP. Moreover, for sa-rectangular RMDPs we replace $\bar{\mathcal{A}}(s) = \mathcal{P}_s$ with $\bar{\mathcal{A}}(s) = \times_{a \in \mathcal{A}} \mathcal{P}_{s,a}$.

We emphasize that although the robust policy evaluation MDP in Definition 4.1 computes the robust value function of the policy $\boldsymbol{\pi}$, it is, nevertheless a regular ordinary MDP. Indeed, although the robust policy evaluation MDP has an infinite action space, its optimal value function exists since the Assumptions 6.0.1–6.0.4 of Puterman (2005) are satisfied. Moreover, since the rewards $\bar{\boldsymbol{r}}$ are continuous (in fact, linear) in $\boldsymbol{\alpha}$ and the sets $\bar{\mathcal{A}}(s)$ are compact by construction of $\mathcal{P}$, there also exists an optimal deterministic stationary policy by Theorem 6.2.7 of Puterman (2005) and the extreme value theorem. When the action sets $\bar{\mathcal{A}}(s)$ are polyhedral, as is the case in our setting, the greedy action for each state can be computed readily from an LP, and the MDP can be solved using any standard MDP algorithm. Section 6.3 describes a new algorithm that computes greedy actions in quasi-linear time, which is much faster than the time required by generic LP solvers.

The next proposition shows that the optimal solution to the robust policy evaluation MDP from Definition 4.1 corresponds to the robust value function $\boldsymbol{v}_{\boldsymbol{\pi}}$ of the policy $\boldsymbol{\pi}$.

Proposition 4.2. *For an RMDP $(\mathcal{S}, \mathcal{A}, \boldsymbol{p}_0, \mathcal{P}, \boldsymbol{r}, \gamma)$ and a policy $\boldsymbol{\pi} \in \Pi$, the optimal value function $\bar{\boldsymbol{v}}^\star$ of the associated robust policy evaluation MDP satisfies $\bar{\boldsymbol{v}}^\star = -\boldsymbol{v}_{\boldsymbol{\pi}}$.*

Proof. Let $\bar{\mathfrak{L}}$ be the Bellman operator for the robust policy evaluation MDP. To prove the result, we first argue that $\bar{\mathfrak{L}}\boldsymbol{v} = -(\mathfrak{L}_{\boldsymbol{\pi}}(-\boldsymbol{v}))$ for every $\boldsymbol{v} \in \mathbb{R}^S$. Indeed, Definition 4.1 and basic algebraic manipulations reveal that

$$\begin{aligned}
(\bar{\mathfrak{L}}\boldsymbol{v})_s &= \max_{\boldsymbol{\alpha}\in\bar{\mathcal{A}}(s)} \bar{r}_{s,\boldsymbol{\alpha}} + \gamma\cdot\bar{\boldsymbol{p}}_{s,\boldsymbol{\alpha}}^\top\boldsymbol{v} \\
&= \max_{\boldsymbol{\alpha}\in\mathcal{P}_s} \left(-\sum_{a\in\mathcal{A}}\pi_{s,a}\cdot\boldsymbol{\alpha}_a^\top\boldsymbol{r}_{s,a}\right) + \gamma\cdot\left(\sum_{a\in\mathcal{A}}\pi_{s,a}\cdot\boldsymbol{\alpha}_a\right)^\top\boldsymbol{v} \qquad \text{from Definition 4.1} \\
&= \max_{\boldsymbol{\alpha}\in\mathcal{P}_s} \sum_{a\in\mathcal{A}}\pi_{s,a}\cdot\boldsymbol{\alpha}_a^\top\left(-\boldsymbol{r}_{s,a}+\gamma\cdot\boldsymbol{v}\right) \\
&= -\min_{\boldsymbol{\alpha}\in\mathcal{P}_s} \sum_{a\in\mathcal{A}}\pi_{s,a}\cdot\boldsymbol{\alpha}_a^\top\left(\boldsymbol{r}_{s,a}+\gamma\cdot(-\boldsymbol{v})\right) \;=\; (-\mathfrak{L}_{\boldsymbol{\pi}}(-\boldsymbol{v}))_s\;.
\end{aligned} \tag{4.1}$$

Let $\bar{\boldsymbol{v}}^\star = \bar{\mathfrak{L}}\bar{\boldsymbol{v}}^\star$ be the fixed point of $\bar{\mathfrak{L}}$, whose existence and uniqueness is guaranteed by the Banach fixed-point theorem since $\bar{\mathfrak{L}}$ is a contraction under the L_∞-norm. Substituting $\bar{\boldsymbol{v}}^\star$ into (4.1) then gives

$$\bar{\boldsymbol{v}}^\star = \bar{\mathfrak{L}}\bar{\boldsymbol{v}}^\star = -\mathfrak{L}_{\boldsymbol{\pi}}(-\bar{\boldsymbol{v}}^\star) \quad\Longrightarrow\quad -\bar{\boldsymbol{v}}^\star = \mathfrak{L}_{\boldsymbol{\pi}}(-\bar{\boldsymbol{v}}^\star)\;,$$

which shows that $-\bar{\boldsymbol{v}}^\star$ is the unique fixed point of $\mathfrak{L}_{\boldsymbol{\pi}}$ since this operator is also an L_∞-contraction (see Lemma A.1 in Appendix A). ■

The robust policy evaluation MDP can be solved by value iteration, (modified) policy iteration, linear programming, or another suitable method. We describe in Section 6.3 an efficient algorithm for calculating $\mathfrak{L}_{\boldsymbol{\pi}_k}$. The accuracy requirement $\|\mathfrak{L}_{\boldsymbol{\pi}_k}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty \leqslant (1-\gamma)\,\epsilon_k$ in Algorithm 4.1 can be used as the stopping criterion in the employed method. As we show next, this condition guarantees that $\|\boldsymbol{v}_k - \boldsymbol{v}_{\boldsymbol{\pi}_k}\|_\infty \leqslant \epsilon_k$, that is, $\boldsymbol{v}_k$ is an ϵ_k-approximation to the robust value function of $\boldsymbol{\pi}_k$.

Proposition 4.3. *Consider a value function $\boldsymbol{v}_k$ and a policy $\boldsymbol{\pi}_k$ in any iteration k of Algorithm 4.1. Then the robust value function $\boldsymbol{v}_{\boldsymbol{\pi}_k}$ of $\boldsymbol{\pi}_k$ satisfies*

$$\|\boldsymbol{v}_{\boldsymbol{\pi}_k} - \boldsymbol{v}_k\|_\infty \leqslant \frac{1}{1-\gamma}\,\|\mathfrak{L}_{\boldsymbol{\pi}_k}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty\;.$$

Proof. The inequality follows immediately from the fact that $\mathfrak{L}_{\boldsymbol{\pi}_k}\boldsymbol{v}_k = \mathfrak{L}\boldsymbol{v}_k$ by construction and from Corollary A.4 if we set $\boldsymbol{\pi} = \boldsymbol{\pi}_k$ and $\boldsymbol{v} = \boldsymbol{v}_k$. ■

Algorithm 4.1 terminates once the condition $\|\mathfrak{L}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty < \frac{1-\gamma}{2}\,\delta$ is met. Note that this condition can be verified using the computations from the current iteration and thus does not require a new application of the Bellman optimality operator. As the next proposition shows, this termination criterion guarantees that the computed policy $\boldsymbol{\pi}_k$ is within δ of the optimal policy.

Proposition 4.4. *Consider any value function $\boldsymbol{v}_k$ and any policy $\boldsymbol{\pi}_k$ greedy for $\boldsymbol{v}_k$. If $\boldsymbol{v}^\star$ is the optimal robust value function that solves $\boldsymbol{v}^\star = \mathfrak{L}\boldsymbol{v}^\star$, then*

$$\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_k}\|_\infty \;\leqslant\; \frac{1}{1-\gamma}\left(\|\mathfrak{L}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty + \|\mathfrak{L}_{\boldsymbol{\pi}_k}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty\right)\;,$$

where $\boldsymbol{v}_{\boldsymbol{\pi}_k}$ the robust value function of $\boldsymbol{\pi}_k$.

The statement of Proposition 4.4 parallels the well-known properties of approximate value functions for classical, ordinary MDPs (Williams and Baird, 1993).

Proof of Proposition 4.4. Using the triangle inequality of vector norms, we see that

$$\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_k}\|_\infty \leqslant \|\boldsymbol{v}^\star - \boldsymbol{v}_k\|_\infty + \|\boldsymbol{v}_k - \boldsymbol{v}_{\boldsymbol{\pi}_k}\|_\infty \ .$$

Using Corollary A.4 in Appendix A with $\boldsymbol{v} = \boldsymbol{v}_k$, the first term $\|\boldsymbol{v}^\star - \boldsymbol{v}_k\|_\infty$ can be bounded from above as follows.

$$\|\boldsymbol{v}^\star - \boldsymbol{v}_k\|_\infty \leqslant \frac{1}{1-\gamma} \|\mathfrak{L}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty$$

The second term $\|\boldsymbol{v}_k - \boldsymbol{v}_{\boldsymbol{\pi}_k}\|_\infty$ above can be bounded using Proposition 4.3. The result then follows by combining the two bounds. ■

We are now ready to show that PPI converges linearly with a rate of at most γ to the optimal robust value function $\boldsymbol{v}^\star$ satisfying $\boldsymbol{v}^\star = \mathfrak{L}\boldsymbol{v}^\star$. This is no worse than the convergence rate of the robust value iteration. The result mirrors results for classical, ordinary MDPs. Regular policy iteration is not known to converge at a faster rate than value iteration even though it is strongly polynomial (Puterman, 2005; Post and Ye, 2015; Hansen et al., 2013).

Theorem 4.5. *Consider $c > 1$ such that $\epsilon_{k+1} \leqslant \gamma^c \epsilon_k$ for all k in Algorithm 4.1. Then the optimality gap of the policy $\boldsymbol{\pi}_{k+1}$ computed in each iteration $k \geqslant 1$ is bounded as*

$$\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}\right\|_\infty \leqslant \gamma^k \left(\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_1}\|_\infty + \frac{2\,\epsilon_1}{(1-\gamma^{c-1})(1-\gamma)} \right) \ .$$

Theorem 4.5 requires the sequence of acceptable evaluation errors ϵ_k to decrease faster than the discount factor γ. As one would expect, the theorem shows that smaller values of ϵ_k lead to a faster convergence in terms of the number of iterations. On the other hand, smaller ϵ_k values also imply that each individual iteration is computationally more expensive.

The proof of Theorem 4.5 follows an approach similar to the convergence proofs of policy iteration (Puterman and Brumelle, 1979; Puterman, 2005), modified policy iteration (Puterman and Shin, 1978; Puterman, 2005) and robust modified policy iteration (Kaufman and Schaefer, 2013). The proofs for (modified) policy iteration start by assuming that the initial value function $\boldsymbol{v}_0$ satisfies $\boldsymbol{v}_0 \leqslant \boldsymbol{v}^\star$; the policy updates and evaluations then increase $\boldsymbol{v}_k$ as fast as value iteration while preserving $\boldsymbol{v}_k \leqslant \boldsymbol{w}_k$ for some $\boldsymbol{w}_k$ satisfying $\lim_{k\to\infty} \boldsymbol{w}_k = \boldsymbol{v}^\star$. The incomplete policy evaluation in RMDPs may result in $\boldsymbol{v}_k \geqslant \boldsymbol{v}^\star$, which precludes the use of the modified policy iteration proof strategy. The convergence proof for RMPI (Kaufman and Schaefer, 2013) inverts the argument by starting with $\boldsymbol{v}_0 \geqslant \boldsymbol{v}^\star$ and decreasing $\boldsymbol{v}_k$ while preserving $\boldsymbol{v}_k \geqslant \boldsymbol{w}_k$. This property, however, is only guaranteed to hold when the policy evaluation step is performed using value iteration. PPI, on the other hand, makes no assumptions on how the policy evaluation step is performed. Its approximate value functions $\boldsymbol{v}_k$ may not satisfy $\boldsymbol{v}_k \leqslant \boldsymbol{v}^\star$ or $\boldsymbol{v}_k \geqslant \boldsymbol{v}_{k+1}$, but the decreasing approximation errors ϵ_k guarantee improvements in $\boldsymbol{v}_{\boldsymbol{\pi}_k}$ that are sufficiently close to those of robust policy iteration. As a result, PPI is guaranteed to compute $\boldsymbol{\pi}^\star$ even though the policies $\boldsymbol{\pi}_k$ can actually become *worse* in the short run.

Proof of Theorem 4.5. We first show that the robust value function of policy $\boldsymbol{\pi}_{k+1}$ is at least as good as that of $\boldsymbol{\pi}_k$ with a tolerance that depends on ϵ_k. Using this result, we then prove that in each iteration k, the optimality gap of the determined policy $\boldsymbol{\pi}_k$ shrinks by the factor γ, again with a tolerance that depends on ϵ_k. In the third and final step, we recursively apply our bound on the optimality gap of the policies $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots$ to obtain the stated convergence rate.

Recall that the vector $\boldsymbol{v}_{\boldsymbol{\pi}_k}$ in Algorithm 4.1 satisfies $\boldsymbol{v}_{\boldsymbol{\pi}_k} = \mathfrak{L}_{\boldsymbol{\pi}_k} \boldsymbol{v}_{\boldsymbol{\pi}_k}$ and represents the *precise* robust value function of $\boldsymbol{\pi}_k$. In contrast, the vector $\boldsymbol{v}_k$ merely approximates $\boldsymbol{v}_{\boldsymbol{\pi}_k}$. Moreover, we denote by $\boldsymbol{\pi}^\star$ the optimal robust policy for the robust value function $\boldsymbol{v}^\star$ and abbreviate the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}_k}$ as $\mathfrak{L}_k$. The proof uses basic properties of robust Bellman operators, which are summarized in Appendix A.

As for the first step, recall that the policy evaluation step of PPI computes a value function $\boldsymbol{v}_k$ that approximates the robust value function $\boldsymbol{v}_{\boldsymbol{\pi}_k}$ within a certain tolerance:

$$\|\mathfrak{L}_k \boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty \leqslant (1-\gamma)\,\epsilon_k \ .$$

Combining this bound with Proposition 4.3 yields $\|\boldsymbol{v}_{\boldsymbol{\pi}_k} - \boldsymbol{v}_k\|_\infty \leqslant \epsilon_k$, which is equivalent to

$$\begin{aligned} \boldsymbol{v}_{\boldsymbol{\pi}_k} &\geqslant \boldsymbol{v}_k \quad - \epsilon_k \cdot \mathbf{1} && (4.2)\\ \boldsymbol{v}_k &\geqslant \boldsymbol{v}_{\boldsymbol{\pi}_k} - \epsilon_k \cdot \mathbf{1} \ . && (4.3) \end{aligned}$$

We use this bound to bound $\mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_k}$ from below as follows:

$$\begin{aligned} \mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_k} &\geqslant \mathfrak{L}_{k+1}(\boldsymbol{v}_k - \epsilon_k \mathbf{1}) && \text{from (4.3) and Lemma A.2}\\ &\geqslant \mathfrak{L}_{k+1} \boldsymbol{v}_k - \gamma\epsilon_k \mathbf{1} && \text{from Lemma A.5}\\ &\geqslant \mathfrak{L}_k \boldsymbol{v}_k - \gamma\epsilon_k \mathbf{1} && \mathfrak{L}_{k+1} \text{ is greedy to } \boldsymbol{v}_k\\ &\geqslant \mathfrak{L}_k(\boldsymbol{v}_{\boldsymbol{\pi}_k} - \epsilon_k \mathbf{1}) - \gamma\epsilon_k \mathbf{1} && \text{from (4.2) and Lemma A.2}\\ &\geqslant \mathfrak{L}_k \boldsymbol{v}_{\boldsymbol{\pi}_k} - 2\gamma\epsilon_k \mathbf{1} && \text{from Lemma A.5}\\ &\geqslant \boldsymbol{v}_{\boldsymbol{\pi}_k} - 2\gamma\epsilon_k \mathbf{1} && \text{because } \boldsymbol{v}_{\boldsymbol{\pi}_k} = \mathfrak{L}_k \boldsymbol{v}_{\boldsymbol{\pi}_k} \end{aligned} \qquad (4.4)$$

This lower bound on $\mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_k}$ readily translates into the following lower bound on $\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}$:

$$\begin{aligned} \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \boldsymbol{v}_{\boldsymbol{\pi}_k} &= \mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \boldsymbol{v}_{\boldsymbol{\pi}_k} && \text{from } \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} = \mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}\\ &= (\mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_k}) + (\mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_k} - \boldsymbol{v}_{\boldsymbol{\pi}_k}) && \text{add } 0\\ &\geqslant \gamma \boldsymbol{P}(\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \boldsymbol{v}_{\boldsymbol{\pi}_k}) + (\mathfrak{L}_{k+1} \boldsymbol{v}_{\boldsymbol{\pi}_k} - \boldsymbol{v}_{\boldsymbol{\pi}_k}) && \text{from Lemma A.6}\\ &\geqslant \gamma \boldsymbol{P}(\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \boldsymbol{v}_{\boldsymbol{\pi}_k}) - 2\gamma\epsilon_k \mathbf{1} && \text{from (4.4)} \end{aligned}$$

Here, $\boldsymbol{P}$ is the stochastic matrix defined in Lemma A.6. Basic algebraic manipulations show that the inequality above further simplifies to

$$(\boldsymbol{I} - \gamma \boldsymbol{P})(\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \boldsymbol{v}_{\boldsymbol{\pi}_k}) \geqslant -2\gamma\epsilon_k \mathbf{1} \ .$$

Recall that for any stochastic matrix $\boldsymbol{P}$, the inverse $(\boldsymbol{I} - \gamma\boldsymbol{P})^{-1}$ exists, satisfies $(\boldsymbol{I} - \gamma\boldsymbol{P})^{-1}\mathbf{1} = (1-\gamma)^{-1}\mathbf{1}$ and is *monotone* in the sense that $(\boldsymbol{I} - \gamma\boldsymbol{P})^{-1}\boldsymbol{x} \geqslant \mathbf{0}$ for any $\boldsymbol{x} \geqslant \mathbf{0}$.

These well-known results all follow the von Neumann series expansion of $(\boldsymbol{I}-\gamma\boldsymbol{P})^{-1}$. Using these properties, the lower bound on $\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}$ simplifies to

$$\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} \;\geqslant\; \boldsymbol{v}_{\boldsymbol{\pi}_k} - \frac{2\,\gamma\,\epsilon_k}{1-\gamma}\mathbf{1}\;, \tag{4.5}$$

which concludes the first step.

To prove the second step, note that the policy improvement step of PPI reduces the optimality gap of $\boldsymbol{\pi}_k$ as follows:

$$\begin{aligned}
\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} &= \boldsymbol{v}^\star - \mathfrak{L}_{k+1}\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} && \text{from the definition of } v_{\boldsymbol{\pi}_{k+1}}\\
&= (\boldsymbol{v}^\star - \mathfrak{L}_{k+1}\boldsymbol{v}_{\boldsymbol{\pi}_k}) - (\mathfrak{L}_{k+1}\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \mathfrak{L}_{k+1}\boldsymbol{v}_{\boldsymbol{\pi}_k}) && \text{subtract 0}\\
&\leqslant (\boldsymbol{v}^\star - \mathfrak{L}_{k+1}\boldsymbol{v}_{\boldsymbol{\pi}_k}) - \gamma\cdot\boldsymbol{P}(\boldsymbol{v}_{\boldsymbol{\pi}_{k+1}} - \boldsymbol{v}_{\boldsymbol{\pi}_k}) && \text{for some } \boldsymbol{P} \text{ from Lemma A.6}\\
&\leqslant (\boldsymbol{v}^\star - \mathfrak{L}_{k+1}\boldsymbol{v}_{\boldsymbol{\pi}_k}) + \frac{2\gamma^2\epsilon_k}{1-\gamma}\mathbf{1} && \text{from (4.5) and } \boldsymbol{P}\mathbf{1}=\mathbf{1}\\
&\leqslant (\boldsymbol{v}^\star - \mathfrak{L}_{k+1}\boldsymbol{v}_k) + \left(\gamma\epsilon_k + \frac{2\gamma^2\epsilon_k}{1-\gamma}\right)\mathbf{1} && \text{from (4.4)}\\
&\leqslant (\boldsymbol{v}^\star - \mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}_k) + \left(\gamma\epsilon_k + \frac{2\gamma^2\epsilon_k}{1-\gamma}\right)\mathbf{1} && \mathfrak{L}_{k+1} \text{ is greedy to } \boldsymbol{v}_k\\
&\leqslant (\boldsymbol{v}^\star - \mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}_{\boldsymbol{\pi}_k}) + \left(2\gamma\epsilon_k + \frac{2\gamma^2\epsilon_k}{1-\gamma}\right)\mathbf{1} && \text{from (4.3) and Lemmas A.2, A.5}\\
&= (\mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}^\star - \mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}_{\boldsymbol{\pi}_k}) + \frac{2\gamma\epsilon_k}{1-\gamma}\mathbf{1} && \text{from } \boldsymbol{v}^\star = \mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}^\star
\end{aligned}$$

Corollary A.3 shows that $\boldsymbol{v}^\star \geqslant \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}$, which allows us to apply the L_∞-norm operator on both sides of the inequality above. Using the contraction property of the robust Bellman policy update (see Lemma A.1), the bound above implies that

$$\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}\right\|_\infty \;\leqslant\; \left\|\mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}^\star - \mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{v}_{\boldsymbol{\pi}_k}\right\|_\infty + \frac{2\gamma\epsilon_k}{1-\gamma} \;\leqslant\; \gamma\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_k}\right\|_\infty + \frac{2\gamma\epsilon_k}{1-\gamma}\;, \tag{4.6}$$

which concludes the second step.

To prove the third and final step, we recursively apply the inequality (4.6) to bound the overall optimality gap of policy $\boldsymbol{\pi}_{k+1}$ as follows:

$$\begin{aligned}
\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_{k+1}}\right\|_\infty &\leqslant \gamma\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_k}\right\|_\infty + \frac{2\gamma\epsilon_k}{1-\gamma}\\
&\leqslant \gamma^2\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_{k-1}}\right\|_\infty + \frac{2\gamma\epsilon_k}{1-\gamma} + \frac{2\gamma^2\epsilon_{k-1}}{1-\gamma}\\
&\leqslant \dots\\
&\leqslant \gamma^k\left\|\boldsymbol{v}^\star - \boldsymbol{v}_{\boldsymbol{\pi}_1}\right\|_\infty + \frac{2}{1-\gamma}\sum_{j=0}^{k-1}\epsilon_{j+1}\gamma^{k-j}\;.
\end{aligned}$$

The postulated choice $\epsilon_j \leqslant \gamma^c\epsilon_{j-1} \leqslant \gamma^{2c}\epsilon_{j-2} \leqslant \dots \leqslant \gamma^{(j-1)c}\epsilon_1$ with $c>1$ implies that

$$\sum_{j=0}^{k-1}\epsilon_{j+1}\gamma^{k-j} \;\leqslant\; \epsilon_1\sum_{j=0}^{k-1}\gamma^{jc}\gamma^{k-j} \;=\; \gamma^k\epsilon_1\sum_{j=0}^{k-1}\gamma^{j(c-1)} \;\leqslant\; \gamma^k\frac{\epsilon_1}{1-\gamma^{c-1}}\;.$$

The result follows by substituting the value of the geometric series in the bound above. ■

PPI improves on several existing algorithms for RMDPs. To the best of our knowledge, the only method that has been shown to solve s-rectangular RMDPs is the robust value iteration (Wiesemann et al., 2013). Robust value iteration is simple and versatile, but it may be slow because computing $\mathfrak{L}$ for s-rectangular RMDPs requires $O(S^3 A^2 \log SA)$ time (see Theorem 6.4). In comparison, PPI uses $\mathfrak{L}$ only to improve policies and can resort to policy iteration to compute the fixed point of $\mathfrak{L}_{\boldsymbol{\pi}_k}$. The evaluation step in policy iteration runs in $O(S^3)$ time required for solving a system of linear equations.

Robust Modified Policy Iteration (RMPI) (Kaufman and Schaefer, 2013), a similar algorithm for sa-rectangular RMDPs, can be cast as a special case of PPI in which the policy evaluation step is solved by value iteration rather than by an arbitrary MDP algorithm. Value iteration can be much slower than (modified) policy iteration due to the complexity of computing $\mathfrak{L}_{\boldsymbol{\pi}_k}$. RMPI also does not reduce the approximation error ϵ_k in the policy evaluations but must be run for a fixed number of value iterations to guarantee convergence. In contrast, PPI only requires that the tolerances ϵ_k decrease at a sufficient rate.

Robust policy iteration (Iyengar, 2005; Hansen et al., 2013) is also similar to PPI, but it has only been proposed in the context of sa-rectangular RMDPs. The main difference to PPI is that the policy evaluation step in robust policy iteration is performed exactly with the tolerance $\epsilon_k = 0$ for all iterations k, which can be done by solving a large LP (Iyengar, 2005). Although this approach is elegant and simple to implement, our experimental results show that it does not scale to even moderately-sized problems.

PPI is general and works for sa-rectangular and s-rectangular RMDPs whose robust Bellman operators $\mathfrak{L}$ and $\mathfrak{L}_{\boldsymbol{\pi}}$ can be computed efficiently. In the next two sections we show that, in fact, the robust Bellman optimality and update operators can be computed efficiently for sa-rectangular and s-rectangular ambiguity sets defined by bounds on the L_1-norm.

5. Computing the Bellman Operator: SA-Rectangular Sets

In this section, we develop an efficient homotopy algorithm to compute the sa-rectangular robust Bellman optimality operator $\mathfrak{L}$ defined in (3.4). Our algorithm computes the inner minimization over $\boldsymbol{p} \in \mathcal{P}_{s,a}$ in (3.4); to compute $\mathfrak{L}\boldsymbol{v}$ for some $\boldsymbol{v} \in \mathbb{R}^S$, we simply execute our algorithm for each action $a \in \mathcal{A}$ and select the maximum of the obtained objective values. To simplify the notation, we fix a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$ throughout this section and drop the associated subscripts whenever the context is unambiguous (for example, we use $\bar{\boldsymbol{p}}$ instead of $\bar{\boldsymbol{p}}_{s,a}$). We also fix a value function $\boldsymbol{v}$ throughout this section.

Our algorithm uses the idea of homotopy continuation (Vanderbei, 1998) to solve the optimization problem $q : \mathbb{R}_+ \to \mathbb{R}$ is parameterized by ξ for a given positive $\boldsymbol{w}$:

$$q(\xi) \;=\; \min_{\boldsymbol{p} \in \Delta^S} \left\{ \boldsymbol{p}^\top \boldsymbol{z} \;\mid\; \|\boldsymbol{p} - \bar{\boldsymbol{p}}\|_{1,\boldsymbol{w}} \leqslant \xi \right\} \tag{5.1}$$

Here, we use the abbreviation $\boldsymbol{z} = \boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}$. Note that ξ plays the role of the budget $\kappa_{s,a}$ in our sa-rectangular uncertainty set $\mathcal{P}_{s,a}$, and that $q(\kappa_{s,a})$ computes the inner minimization over $\boldsymbol{p} \in \mathcal{P}_{s,a}$ in (3.4). Our homotopy method achieves its efficiency by computing $q(\xi)$ for $\xi = 0$ and subsequently for all $\xi \in (0, \kappa_{s,a}]$ instead of computing $q(\kappa_{s,a})$ directly (Asif and

$i \in \ldots \rightarrow$	$\mathcal{N}_B$	$\mathcal{U}_B$	$\mathcal{L}_B$	$\mathcal{E}_B$	$\bar{\mathcal{N}}_B$	$\bar{\mathcal{U}}_B$	$\bar{\mathcal{L}}_B$
$p_i - \bar{p}_i \leqslant l_i$	·	✓	·	✓	·	✓	·
$\bar{p}_i - p_i \leqslant l_i$	·	·	✓	✓	·	·	✓
$p_i \geqslant 0$	·	·	·	·	✓	✓	✓

Table 1: Possible subsets of active constraints in (5.3). Check marks indicate active constraints that are included in the basis B for each index $i = 1, \ldots, S$.

Romberg, 2009; Garrigues and El Ghaoui, 2009). The problem $q(0)$ is easy since the only feasible solution is $\boldsymbol{p} = \bar{\boldsymbol{p}}$, and thus $q(0) = \bar{\boldsymbol{p}}^\top \boldsymbol{z}$. We then trace an optimal solution $\boldsymbol{p}^\star(\xi)$ as ξ increases, until we reach $\xi = \kappa_{s,a}$. Our homotopy algorithm is fast because the optimal solution can be traced efficiently when ξ is increased. As we show below, $q(\xi)$ is piecewise linear with at most S^2 pieces (or S pieces, if all components of $\boldsymbol{w}$ are equal), and exactly two components of $\boldsymbol{p}^\star(\xi)$ change when ξ increases.

By construction, $q(\xi)$ varies with ξ only when ξ is small enough so that the constraint $\|\boldsymbol{p} - \bar{\boldsymbol{p}}\|_{1,\boldsymbol{w}} \leqslant \xi$ in (5.1) is binding at optimality. To avoid case distinctions for the trivial case when $\|\boldsymbol{p} - \bar{\boldsymbol{p}}\|_{1,\boldsymbol{w}} < \xi$ at optimality and $q(\xi)$ is constant, we assume in the remainder of this section that ξ is small enough. Our homotopy algorithm treats large ξ identically to the largest ξ for which the constraint is binding at optimality.

In the remainder of this section, we first investigate the structure of basic feasible solutions to the problem (5.1) in Section 5.1. We then exploit this structure to develop our homotopy method in Section 5.2, and we conclude with a complexity analysis in Section 5.3.

5.1 Properties of the Parametric Optimization Problem

We now discuss important technical properties of the parametric optimization problem in (5.1), which can be reformulated as the following linear program:

$$
\begin{array}{rlll}
q(\xi) = & \underset{\boldsymbol{p},\boldsymbol{l} \in \mathbb{R}^S}{\text{minimize}} & \boldsymbol{z}^\top \boldsymbol{p} & \\
& \text{subject to} & p_i - \bar{p}_i \leqslant l_i & i = 1, \ldots, S \\
& & \bar{p}_i - p_i \leqslant l_i & i = 1, \ldots, S \\
& & p_i \geqslant 0 & i = 1, \ldots, S \\
& & \mathbf{1}^\top \boldsymbol{p} = 1, \;\; \boldsymbol{w}^\top \boldsymbol{l} = \xi &
\end{array}
\tag{5.2}
$$

Note that the constraint $\boldsymbol{l} \geqslant \mathbf{0}$ is enforced implicitly. Solving (5.2) using a generic LP algorithm can be too slow to be practical as our empirical results in Section 7 show.

Throughout the paper, we make the following assumption regarding problem (5.2).

Assumption 5.1. The parameters $\boldsymbol{z}$ and $\boldsymbol{w}$ of (5.2) satisfy the following conditions.

1. Every $i, j, k, \ell \in \{1, \ldots, S\}$ with $i \neq j$ and $k \neq \ell$ satisfy
$$(w_i + w_j)(z_k - z_\ell) \neq (w_k + w_\ell)(z_i - z_j).$$
2. Every $i, j, k, \ell \in \{1, \ldots, S\}$ with $i \neq j$, $k \neq \ell$, $w_i \neq w_j$ and $w_k \neq w_\ell$ satisfy
$$(w_i - w_j)(z_k - z_\ell) \neq (w_k - w_\ell)(z_i - z_j).$$

As we will see later in Lemma 5.5, the above assumption implies the uniqueness of the solution $\boldsymbol{p}^\star$ in problem (5.2) for any $\xi \in \mathbb{R}_+$. Since Assumption 5.1 imposes a finite number of equality constraints on $\boldsymbol{z}$ and $\boldsymbol{w}$, the assumption can be satisfied by arbitrarily small perturbations of $\boldsymbol{z}$, which result in arbitrarily small perturbations of $q(\xi)$ in (5.2). When the L_1-norm is unweighted, that is, when $w_1, \ldots, w_S = 1$, the assumption requires that the pairwise differences $z_i - z_j$ of $\boldsymbol{z}$ are all different.

To develop the homotopy algorithm, we need the concept of a basis to a linear program (Bertsimas and Tsitsiklis, 1997; Vanderbei, 1998). Each *basis* B in (5.2) is fully characterized by $2S$ *linearly independent* (inequality and/or equality) constraints that are *active*; see for example Definition 2.9 of Bertsimas and Tsitsiklis (1997). Remember that an active constraint is satisfied with equality, but not every constraint that is satisfied as equality has to be active in any basis B. Recall that each basis uniquely defines the values $\boldsymbol{p}$ and $\boldsymbol{l}$ in the linear program for any ξ.

The key to the efficiency of our method is the special structure of the bases in (5.2), which we describe next. For any given component $i = 1, \ldots, S$, a subset of the following constraints in (5.1) can be active:

$$p_i - \bar{p}_i \leqslant l_i, \qquad \bar{p}_i - p_i \leqslant l_i, \qquad p_i \geqslant 0 \ . \tag{5.3}$$

Table 1 shows all possible subsets of active constraints (5.3). The letters $\mathcal{N}$, $\mathcal{U}$, $\mathcal{L}$ and $\mathcal{E}$ mnemonize the cases where none of the constraints is active, only the upper bound or the lower bound on $\bar{p}_i$ is active and where both bounds are simultaneously active and hence p_i equals $\bar{p}_i$. The three cases in which the nonnegativity constraint $p_i \geqslant 0$ is active are adorned by a bar.

Note that the constraints (5.3) are linearly dependent for each $i = 1, \ldots, S$ because they involve only two variables; thus, they cannot be all active. As a result, the sets in Table 1 are mutually exclusive, jointly exhaustive, and partition the index set $1, \ldots, S$.

In addition to the inequality constraints (5.3), a basis B may include one or both of the equality constraints from (5.2). The set $\mathcal{Q}_B \subseteq \{1, 2\}$ indicates which of these equality constraints are included in the basis B. Together with the sets from Table 1, $\mathcal{Q}_B$ uniquely identifies any basis B. The $2S$ linearly independent active constraints involving the $2S$ decision variables uniquely specify a solution $(\boldsymbol{p}, \boldsymbol{l})$ for a given basis B as

$$\begin{aligned} p_i - \bar{p}_i &= l_i && \forall i \in \mathcal{U}_B \cup \mathcal{E}_B \cup \bar{\mathcal{U}}_B \\ \bar{p}_i - p_i &= l_i && \forall i \in \mathcal{L}_B \cup \mathcal{E}_B \cup \bar{\mathcal{L}}_B \\ p_i &= 0 && \forall i \in \bar{\mathcal{N}}_B \cup \bar{\mathcal{U}}_B \cup \bar{\mathcal{L}}_B \\ \boldsymbol{1}^\top \boldsymbol{p} &= 1 && \text{if } 1 \in \mathcal{Q}_B \\ \boldsymbol{w}^\top \boldsymbol{l} &= \xi && \text{if } 2 \in \mathcal{Q}_B \ . \end{aligned} \tag{5.4}$$

We use $\boldsymbol{p}_B(\xi)$ to denote the solution $\boldsymbol{p}$ to (5.4) and define $q_B(\xi) = \boldsymbol{z}^\top \boldsymbol{p}_B(\xi)$ for any ξ. The vector $\boldsymbol{p}_B(\xi)$ may be feasible in (5.2) only for some values of ξ.

Before proving formally the structure of the optimal bases in (5.2), we illustrate their importance when developing the homotopy method. It is well known that $q(\xi)$ and $\boldsymbol{p}^\star(\xi)$ are *piecewise linear* in ξ (Vanderbei, 1998) and are linear in ξ for each optimal basis in (5.2). A point of non-linearity (referred to as a "breakpoint" or a "knot") occurs whenever there

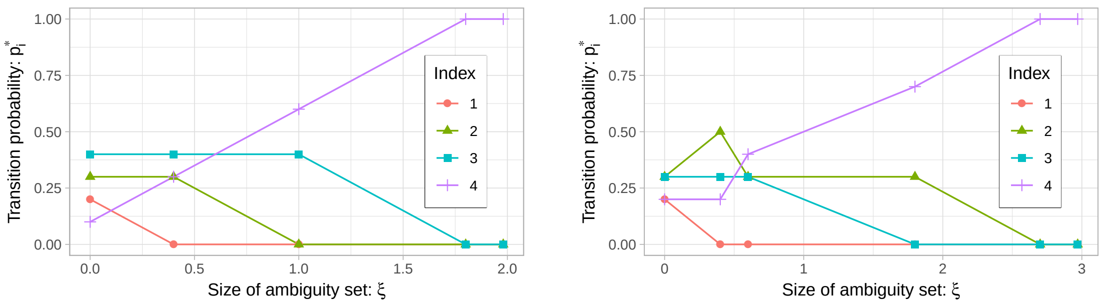

Figure 1: Example evolution of $\boldsymbol{p}^\star(\xi)$ for a uniform (left) and a non-uniform weight vector $\boldsymbol{w}$ (right). Point markers indicate breakpoints where the optimal bases change.

is a change in the optimal bases for a particular ξ. Our algorithm starts with $\xi = 0$ and traces an optimal basis in (5.2) while increasing ξ. The following two examples show the evolution of $\boldsymbol{p}^\star(\xi)$, which is unique in these cases, as a function of ξ.

Example 5.2 (Uniform Weights)**.** *Consider the function $q(\xi)$ in* (5.1) *for an RMDP with 4 states, $\boldsymbol{z} = (4, 3, 2, 1)^\top$, $\bar{\boldsymbol{p}} = (0.2, 0.3, 0.4, 0.1)^\top$ and $\boldsymbol{w} = \mathbf{1}$. Figure 1 (left) depicts the evolution of $\boldsymbol{p}^\star(\xi)$ as a function of ξ. Component p_4 is the receiver for all values of ξ, and the donors are the components p_1, p_2 and p_3. We show in Section 5.3 that for uniform weights $\boldsymbol{w}$, the component with the smallest value of $\boldsymbol{z}$ is always the sole receiver.*

Example 5.3 (Non-Uniform Weights)**.** *Consider the function $q(\xi)$ in* (5.1) *for an RMDP with 4 states, $\boldsymbol{z} = (2.9, 0.9, 1.5, 0.0)^\top$, $\bar{\boldsymbol{p}} = (0.2, 0.3, 0.3, 0.2)^\top$ and $w = (1, 1, 2, 2)^\top$. Figure 1 (right) depicts the evolution of $\boldsymbol{p}^\star(\xi)$ as a function of ξ. The donor-receiver pairs are* $(1, 2)$, $(2, 4)$ $(3, 4)$ *and again* $(2, 4)$. *In particular, several components can serve as receivers for different values of ξ when $\boldsymbol{w}$ is non-uniform. Also, the same component can serve as a donor more than once.*

Examples 5.2 and 5.3 illustrate that the values of *exactly two* components of $\boldsymbol{p}^\star(\xi)$ change with increasing ξ. Since the components of $\boldsymbol{p}^\star(\xi)$ must sum to 1, one component p_j increases and another component p_i decreases. We say that p_i is a *donor* as it donates some of its probability mass to the *receiver* p_j.

The following key lemma will be used to show that the behavior observed above is not a coincidence and that at most two components of $\boldsymbol{p}(\xi)$ change with an increasing ξ. As will become apparent in Lemma 5.5, only components in $\mathcal{U}_B$ and $\mathcal{L}_B$ and $\mathcal{N}_B$ can change with an increasing ξ. The lemma bounds the cardinality of these sets from above by 2.

Lemma 5.4. *Any basis B to* (5.2) *satisfies* $|\mathcal{U}_B| + |\mathcal{L}_B| + |\bar{\mathcal{N}}_B| + 2|\mathcal{N}_B| = |\mathcal{Q}_B| \leqslant 2$.

Proof. The statement follows from a counting argument. Since the sets listed in Table 1 partition the index set $1, \ldots, S$, their cardinalities must sum to S:

$$|\mathcal{N}_B| + |\mathcal{U}_B| + |\mathcal{L}_B| + |\mathcal{E}_B| + |\bar{\mathcal{N}}_B| + |\bar{\mathcal{U}}_B| + |\bar{\mathcal{L}}_B| = S. \tag{5.5}$$

Each index $i = 1, \ldots, S$ contributes between zero and two active constraints to the basis. For example, $i \in \mathcal{N}_B$ contributes no constraint, whereas $i \in \bar{\mathcal{U}}_B$ contributes 2 constraints. The requirement that B contains exactly $2S$ linearly independent constraints translates to

$$0 \cdot |\mathcal{N}_B| + 1 \cdot |\mathcal{U}_B| + 1 \cdot |\mathcal{L}_B| + 2 \cdot |\mathcal{E}_B| + 1 \cdot |\bar{\mathcal{N}}_B| + 2 \cdot |\bar{\mathcal{U}}_B| + 2 \cdot |\bar{\mathcal{L}}_B| + |\mathcal{Q}_B| = 2S \ . \quad (5.6)$$

Subtracting two times (5.5) from (5.6), we get

$$-2 \cdot |\mathcal{N}_B| - |\mathcal{U}_B| - |\mathcal{L}_B| - |\bar{\mathcal{N}}_B| + |\mathcal{Q}_B| = 0 \ .$$

The result then follows by performing elementary algebra. ■

We next show that for any basis B feasible in (5.2) for a given ξ, the components in $\mathcal{U}_B$ and $\mathcal{L}_B$ act as donor-receiver pairs.

Lemma 5.5. *Consider some $\xi > 0$ and a basis B to problem* (5.2) *that is feasible in a neighborhood of ξ. Then the derivatives $\dot{p}_i = \frac{d}{d\xi}(\boldsymbol{p}_B(\xi))_i, i = 1, \ldots, S$, and $\dot{q} = \frac{d}{d\xi} q_B(\xi)$ satisfy:*

(C1) *If $\mathcal{U}_B = \{i\}$ and $\mathcal{L}_B = \{j\}$, $i \neq j$, then:*

$$\dot{q} = \frac{z_i - z_j}{w_i + w_j}, \qquad \dot{p}_i = \frac{1}{w_i + w_j}, \qquad \dot{p}_j = -\frac{1}{w_i + w_j}.$$

(C2) *If $\mathcal{U}_B = \{i, j\}$, $i \neq j$ and $w_i \neq w_j$, and $\mathcal{L}_B = \emptyset$, then:*

$$\dot{q} = \frac{z_i - z_j}{w_i - w_j}, \qquad \dot{p}_i = \frac{1}{w_i - w_j}, \qquad \dot{p}_j = -\frac{1}{w_i - w_j}.$$

The derivatives $\dot{\boldsymbol{p}}$ and $\dot{q}$ of all other types of feasible bases to problem (5.2) *are zero.*

The derivatives in the proposition exist since the functions $\boldsymbol{p}_B(\xi)$ and $q_B(\xi)$ are linear for any fixed basis B. The derivative $\dot{\boldsymbol{p}}$ shows that in a basis of class (C1), i is the receiver and j is the donor. In a basis of class (C2), on the other hand, an inspection of $\dot{\boldsymbol{p}}$ reveals that i is the receiver and j is the donor whenever $w_i > w_j$, and the reverse situation occurs when $w_i < w_j$.

Proof of Lemma 5.5. In this proof, we consider a fixed basis B and thus drop the subscript B to reduce clutter. We also denote by $\boldsymbol{x}_{\mathcal{D}}$ the subvector of $\boldsymbol{x} \in \mathbb{R}^S$ formed by the elements x_i, $i \in \mathcal{D}$, whose indices are contained in the set $\mathcal{D} \subseteq \mathcal{S}$.

First, observe that $\dot{p}_i \neq 0$ is only possible if $i \in \mathcal{U} \cup \mathcal{L} \cup \mathcal{N}$. According to Table 1, if $i \notin \mathcal{U} \cup \mathcal{L} \cup \mathcal{N}$, then $i \in \mathcal{E} \cup \bar{\mathcal{U}} \cup \bar{\mathcal{L}} \cup \bar{\mathcal{N}}$. In the case where $i \in \mathcal{E}$, we have $p_i = \overline{p}_i$ from the definition of $\mathcal{E}$, and thus $\dot{p}_i = 0$. If $i \in \bar{\mathcal{U}} \cup \bar{\mathcal{L}} \cup \bar{\mathcal{N}}$, then $p_i = 0$ from the definitions of the sets, and thus $\dot{p}_i = 0$.

To derive the desired results, we consider the changes of q and $\boldsymbol{p}$ when we vary ξ in its neighborhood with the same basis B, which by definition identifies the active constraints in (5.2) even when ξ changes. Because at least two components of $\boldsymbol{p}_B(\xi)$ need to change as we vary ξ, we can restrict ourselves to bases B that satisfy $|\mathcal{U}| + |\mathcal{L}| + |\mathcal{N}| \geqslant 2$. Since Lemma 5.4 furthermore shows that $|\mathcal{U}| + |\mathcal{L}| + 2|\mathcal{N}| \leqslant 2$, we only need to consider three following cases:

(C1) $|\mathcal{U}| = |\mathcal{L}| = 1$ and $|\mathcal{N}| = 0$;
(C2) $|\mathcal{U}| = 2$ and $|\mathcal{L}| = |\mathcal{N}| = 0$;
(C3) $|\mathcal{L}| = 2$ and $|\mathcal{U}| = |\mathcal{N}| = 0$;

In the remainder of the proof, let $\boldsymbol{p}$ and $\boldsymbol{l}$ be the unique vectors that satisfy the active constraints in the basis B. Then, Table 1 implies the following useful equality that any $\boldsymbol{p}$ must satisfy.

$$\begin{aligned} 1 = \mathbf{1}^\top \boldsymbol{p} &= \mathbf{1}^\top \boldsymbol{p}_{\mathcal{N}} + \mathbf{1}^\top \boldsymbol{p}_{\mathcal{U}} + \mathbf{1}^\top \boldsymbol{p}_{\mathcal{L}} + \mathbf{1}^\top \boldsymbol{p}_{\mathcal{E}} + \mathbf{1}^\top \boldsymbol{p}_{\bar{\mathcal{N}}} + \mathbf{1}^\top \boldsymbol{p}_{\bar{\mathcal{U}}} + \mathbf{1}^\top \boldsymbol{p}_{\bar{\mathcal{L}}} \\ &= \mathbf{1}^\top \boldsymbol{p}_{\mathcal{N}} + \mathbf{1}^\top \boldsymbol{p}_{\mathcal{U}} + \mathbf{1}^\top \boldsymbol{p}_{\mathcal{L}} + \mathbf{1}^\top \bar{\boldsymbol{p}}_{\mathcal{E}} \end{aligned} \tag{5.7}$$

Case (C1): $\mathcal{U} = \{i\}$, $\mathcal{L} = \{j\}$, $i \neq j$, and $\mathcal{N} = \emptyset$:
Equation (5.7) implies that $p_i + p_j = 1 - \mathbf{1}^\top \bar{\boldsymbol{p}}_{\mathcal{E}}$ and thus $\dot{p}_i + \dot{p}_j = 0$. We also have

$$\begin{aligned} \boldsymbol{w}^\top \boldsymbol{l} &= \boldsymbol{w}_{\mathcal{N}}^\top \boldsymbol{l}_{\mathcal{N}} + \boldsymbol{w}_{\mathcal{U}}^\top \boldsymbol{l}_{\mathcal{U}} + \boldsymbol{w}_{\mathcal{L}}^\top \boldsymbol{l}_{\mathcal{L}} + \boldsymbol{w}_{\mathcal{E}}^\top \boldsymbol{l}_{\mathcal{E}} + \boldsymbol{w}_{\bar{\mathcal{N}}}^\top \boldsymbol{l}_{\bar{\mathcal{N}}} + \boldsymbol{w}_{\bar{\mathcal{U}}}^\top \boldsymbol{l}_{\bar{\mathcal{U}}} + \boldsymbol{w}_{\bar{\mathcal{L}}}^\top \boldsymbol{l}_{\bar{\mathcal{L}}} \\ &= w_i l_i + w_j l_j + \boldsymbol{w}_{\mathcal{E}}^\top \boldsymbol{l}_{\mathcal{E}} + \boldsymbol{w}_{\bar{\mathcal{U}}}^\top \boldsymbol{l}_{\bar{\mathcal{U}}} + \boldsymbol{w}_{\bar{\mathcal{L}}}^\top \boldsymbol{l}_{\bar{\mathcal{L}}} \\ &= w_i l_i + w_j l_j - \boldsymbol{w}_{\bar{\mathcal{U}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{U}}} + \boldsymbol{w}_{\bar{\mathcal{L}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{L}}} \\ &= w_i (p_i - \bar{p}_i) + w_j (\bar{p}_j - p_j) - \boldsymbol{w}_{\bar{\mathcal{U}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{U}}} + \boldsymbol{w}_{\bar{\mathcal{L}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{L}}} \ , \end{aligned}$$

where the second identity follows from the fact that $\mathcal{N} = \emptyset$, $\mathcal{U} = \{i\}$ and $\mathcal{L} = \{j\}$ by assumption, as well as $\bar{\mathcal{N}} = \emptyset$ due to Lemma 5.4. The third identity holds since the active constraints in $\mathcal{E}$, $\bar{\mathcal{U}}$ and $\bar{\mathcal{L}}$ imply that $\boldsymbol{l}_{\mathcal{E}} = \mathbf{0}$, $\boldsymbol{l}_{\bar{\mathcal{U}}} = -\bar{\boldsymbol{p}}_{\bar{\mathcal{U}}}$ and $\boldsymbol{l}_{\bar{\mathcal{L}}} = \bar{\boldsymbol{p}}_{\bar{\mathcal{L}}}$, respectively. The last identity, finally, is due to the fact that $p_i - \bar{p}_i = l_i$ since $i \in \mathcal{U}$ and $\bar{p}_j - p_j = l_j$ since $j \in \mathcal{L}$. Since any feasible basis B satisfies that $\boldsymbol{w}^\top \boldsymbol{l} = \xi$, we thus obtain that

$$\begin{aligned} & w_i (p_i - \bar{p}_i) + w_j (\bar{p}_j - p_j) = \xi + \boldsymbol{w}_{\bar{\mathcal{U}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{U}}} - \boldsymbol{w}_{\bar{\mathcal{L}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{L}}} \\ \Longrightarrow \quad & w_i \dot{p}_i - w_j \dot{p}_j = 1 && \text{taking } {}^{d}\!/_{d\xi} \text{ on both sides} \\ \Longleftrightarrow \quad & w_i \dot{p}_i + w_j \dot{p}_i = 1 && \text{from } \dot{p}_i + \dot{p}_j = 0 \\ \Longleftrightarrow \quad & \dot{p}_i = \tfrac{1}{w_i + w_j}. \end{aligned}$$

The expressions for $\dot{p}_j$ and $\dot{q}$ follow from $\dot{p}_i + \dot{p}_j = 0$ and elementary algebra, respectively.

Case (C2): $\mathcal{U} = \{i, j\}$, $i \neq j$, and $\mathcal{L} = \mathcal{N} = \emptyset$:
Similar steps to case (C1) show that

$$w_i (p_i - \bar{p}_i) + w_j (p_j - \bar{p}_j) = \xi + \boldsymbol{w}_{\bar{\mathcal{U}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{U}}} - \boldsymbol{w}_{\bar{\mathcal{L}}}^\top \bar{\boldsymbol{p}}_{\bar{\mathcal{L}}} \ ,$$

which in turn yields the desired expressions for $\dot{p}_i$, $\dot{p}_j$ and $\dot{q}$. Note that if $w_i = w_j$ in the equation above, then the left hand side's derivative with respect to ξ is zero, and we obtain a contradiction. This allows us to assume that $w_i \neq w_j$ in case (C2).

Case (C3): $\mathcal{L} = \{i, j\}$, $i \neq j$, and $\mathcal{U} = \mathcal{N} = \emptyset$:
Note that $\boldsymbol{p}_{\mathcal{L}} \leqslant \bar{\boldsymbol{p}}_{\mathcal{L}}$ since $\boldsymbol{l}_{\mathcal{L}}$ satisfies both $\boldsymbol{l}_{\mathcal{L}} \geqslant \mathbf{0}$ and $\boldsymbol{l}_{\mathcal{L}} = \bar{\boldsymbol{p}}_{\mathcal{L}} - \boldsymbol{p}_{\mathcal{L}}$. Since (5.7) implies that $\mathbf{1}^\top \boldsymbol{p} = \mathbf{1}^\top \boldsymbol{p}_{\mathcal{L}} + \mathbf{1}^\top \bar{\boldsymbol{p}}_{\mathcal{E}} = 1$, however, we conclude that $\boldsymbol{p}_{\mathcal{L}} = \bar{\boldsymbol{p}}_{\mathcal{L}}$, that is, we must have $\dot{\boldsymbol{p}} = \mathbf{0}$ and $\dot{q} = 0$. ■

5.2 Homotopy Algorithm

We are now ready to describe our homotopy method, which is presented in Algorithm 5.1. The algorithm starts at $\xi_0 = 0$ with the optimal solution $\boldsymbol{p}_0 = \bar{\boldsymbol{p}}$ achieving the objective

```
Input: LP parameters $\boldsymbol{z}$, $\boldsymbol{w}$ and $\bar{\boldsymbol{p}}$
Output: Breakpoints $(\xi_t)_{t=0,\dots T+1}$ and values $(q_t)_{t=0,\dots T+1}$, defining the function $q$
Initialize $\xi_0 \leftarrow 0$, $\boldsymbol{p}_0 \leftarrow \bar{\boldsymbol{p}}$ and $q_0 \leftarrow q(\xi_0) = \boldsymbol{p}_0^\top \boldsymbol{z}$ ;

// Derivatives $\dot{q}$ for bases of (5.2) (see Lemma 5.5)
for $i = 1 \dots S$ do
    for $j = 1 \dots S$ satisfying $i \neq j$ do
        Case C1 ($\mathcal{U}_B = \{i\}$ and $\mathcal{L}_B = \{j\}$): $\alpha_{i,j} \leftarrow (z_i - z_j)/(w_i + w_j)$ ;
        Case C2 ($\mathcal{U}_B = \{i,j\}$): $\beta_{i,j} \leftarrow (z_i - z_j)/(w_i - w_j)$ if $w_i \neq w_j$ ;
    end
end

// Sort derivatives and map to bases (see Lemma 5.5)
Store $(\alpha_{i,j}, \mathrm{C1})$, $i \neq j$ and $\alpha_{i,j} < 0$, and $(\beta_{i,j}, \mathrm{C2})$, $i \neq j$ and $\beta_{i,j} < 0$, in a list $\mathcal{D}$ ;
Sort the list $\mathcal{D}$ in ascending order of the first component;
Construct bases $B_1, \dots, B_T$ from $\mathcal{D} = (d_1, \dots, d_T)$ as:
    $B_m = \begin{cases} (\mathcal{U}_B = \{i\},\ \mathcal{L}_B = \{j\}) & \text{if } d_m = (\alpha_{i,j}, \mathrm{C1}), \\ (\mathcal{U}_B = \{i,j\},\ \mathcal{L}_B = \emptyset) & \text{if } d_m = (\beta_{i,j}, \mathrm{C2}); \end{cases}$

// Trace optimal $\boldsymbol{p}_B(\xi)$ with increasing $\xi$
for $l = 1 \dots T$ do
    if $B_l$ infeasible for $\xi_{l-1}$ then
        Set $\xi_l \leftarrow \xi_{l-1}$, $\boldsymbol{p}_l \leftarrow \boldsymbol{p}_{l-1}$ and $q_l \leftarrow q_{l-1}$ ;
        continue;
    end
    Compute $\dot{\boldsymbol{p}}$, $\dot{q}$ for $B_l$ as well as both cases (C1) and (C2) from Lemma 5.5 ;
    Compute maximum $\Delta\xi$ for which $B_l$ remains feasible:
    $\Delta\xi \leftarrow \begin{cases} \max\{x \geqslant 0 \mid (\boldsymbol{p}_{l-1})_j + x \cdot \dot{p}_j \geqslant 0\} & \text{if } d_l = (\alpha_{i,j}, \mathrm{C1}), \\ \max\{x \geqslant 0 \mid (\boldsymbol{p}_{l-1})_j + x \cdot \dot{p}_j \geqslant \bar{p}_j\} & \text{if } d_l = (\beta_{i,j}, \mathrm{C2}) \text{ and } w_i > w_j, \\ \max\{x \geqslant 0 \mid (\boldsymbol{p}_{l-1})_i + x \cdot \dot{p}_i \geqslant \bar{p}_i\} & \text{if } d_l = (\beta_{i,j}, \mathrm{C2}) \text{ and } w_i < w_j; \end{cases}$
    Set $\xi_l \leftarrow \xi_{l-1} + \Delta\xi$, $\boldsymbol{p}_l \leftarrow \boldsymbol{p}_{l-1} + \Delta\xi \cdot \dot{\boldsymbol{p}}$, and $q_l \leftarrow q_{l-1} + \Delta\xi \cdot \dot{q}$ ;
end
Set $\xi_{T+1} \leftarrow \infty$ and $q_{T+1} \leftarrow q_T$;
return Breakpoints $(\xi_t)_{t=0,\dots T+1}$ and values $(q_t)_{t=0,\dots T+1}$.
```

Algorithm 5.1: Homotopy method to compute $q(\xi)$.

value $q_0 = \boldsymbol{p}_0^\top \boldsymbol{z}$. The algorithm subsequently traces each optimal basis as ξ increases, until the basis becomes infeasible and is replaced with the next basis. Since the function $q(\xi)$ is convex, it is sufficient to consider bases that have a derivative $\dot{q}$ that is no smaller than the ones traced previously. Note that a basis of class (C1) satisfies $\mathcal{U}_B = \{i\}$ and $\mathcal{L}_B = \{j\}$ for some receiver $i \in S$ and some donor $j \in S$, $j \neq i$, and this basis is feasible at $\boldsymbol{p} = \boldsymbol{p}^\star(\xi)$, $\xi \geqslant 0$, only if $p_i \in [\bar{p}_i, 1]$ and $p_j \in [0, \bar{p}_j]$ (see Lemma 5.5). Likewise, a basis of class (C2) satisfies $\mathcal{U}_B = \{i, j\}$, $i \neq j$, and $\mathcal{L}_B = \varnothing$, and it is feasible at $\boldsymbol{p} = \boldsymbol{p}^\star(\xi)$, $\xi \geqslant 0$, only if $p_i \in [\bar{p}_i, 1]$ and $p_j \in [\bar{p}_j, 1]$. In a basis of class (C2), i is the receiver and j is the donor whenever $w_i > w_j$, and the reverse situation occurs when $w_i < w_j$. In the case where $\dot{q}_{B_1} = \dot{q}_{B_2}$ for two different bases B_1 and B_2, the homotopy method would have to inspect the solution trajectories of both bases as they can differ for larger values of ξ. This would increase the computational burden of the homotopy method. Assumption 5.1 excludes these pathological cases by stipulating that all bases in Algorithm 5.1 have pairwise different slopes $\dot{q}$. As a by-product, the assumption guarantees that $\boldsymbol{p}^\star(\xi)$ is unique for all ξ since there is only one optimal sequence of bases, which implies that $\boldsymbol{p}(\xi)$ remains unique as ξ increases. Our implementation accounts for floating-point errors by using a queue to store and examine the feasibility of all bases that are within some small ϵ of the last $\dot{q}$.

Algorithm 5.1 generates the entire solution path of $q(\xi)$. Since $q(\xi)$ is a piecewise linear function, the outputs of Algorithm 5.1 are the breakpoints of $q(\xi)$ and their function values. If the goal is to compute the function q for a particular value of ξ, then we can terminate the algorithm once the for loop over l has reached this value. In contrast, our bisection method for s-rectangular ambiguity sets (described in the next section) requires the entire solution path to compute robust Bellman policy updates. We also note that Algorithm 5.1 records all vectors $\boldsymbol{p}_1, \ldots \boldsymbol{p}_T$. This is done for ease of exposition; for practical implementations, it is sufficient to only store the current iterate $\boldsymbol{p}_l$ and update the two components that change in the "for loop" over l.

The following theorem proves the correctness of our homotopy algorithm. It shows that the function q is a piecewise linear function defined by the output of Algorithm 5.1.

Theorem 5.6. *Let $(\xi_t)_{t=0,\ldots,T+1}$ and $(q_t)_{t=0,\ldots,T+1}$ be the output of Algorithm 5.1. Then, $q(\xi)$ is a piecewise linear function with breakpoints ξ_l that satisfies $q(\xi_t) = q_t$ for $t = 0, \ldots, T+1$.*

We prove the statement by contradiction. Since each point q_l returned by Algorithm 5.1 corresponds to the objective value of a feasible solution to problem (5.2) at $\xi = \xi_l$, the output generated by Algorithm 5.1 provides an upper bound on $q(\xi)$. Assume to the contrary that the output does not coincide point-wise with the function $q(\xi)$. In that case, there must be a value of ξ at which the homotopy method disregards a feasible basis that has a strictly smaller derivative than the one selected. This, however, contradicts the way in which bases are selected by the algorithm.

Proof of Theorem 5.6. For $\xi \leqslant \xi_T$, Algorithm 5.1 computes the piecewise linear function

$$g(\xi) = \min_{\boldsymbol{\alpha} \in \Delta^{T+1}} \left\{ \sum_{t=0}^{T} \alpha_t \, q_t \;\middle|\; \sum_{t=0}^{T} \alpha_t \, \xi_t = \xi \right\}.$$

To prove the statement, we show that $g(\xi) = q(\xi)$ for all $\xi \in [0, \xi_T]$. Note that $g(\xi) \geqslant q(\xi)$ for all $\xi \in [0, \xi_T]$ by construction since our algorithm only considers feasible bases. Also, from the construction of g, we have that $q(\xi_0) = g(\xi_0)$ for the initial point.

To see that $g(\xi) \leqslant q(\xi)$, we need to show that Algorithm 5.1 does not skip any relevant bases. To this end, assume to the contrary that there exists a $\xi' \in (\xi_0, \xi_T]$ such that $q(\xi') < g(\xi')$. Without loss of generality, there exists a value ξ' such that that $q(\xi) = g(\xi)$ for all breakpoints $\xi \leqslant \xi'$ of q; this can always be achieved by choosing a sufficiently small value of ξ' where q and g differ. Let ξ_l be the largest element in $\{\xi_t \mid t = 0, \ldots, T\}$ such that $\xi_l < \xi'$, that is, we have $\xi_l < \xi' \leqslant \xi_{l+1}$. Such ξ_l exists because $\xi' > \xi_0$ and $q(\xi_0) = g(\xi_0)$. Let B_l be the basis chosen by Algorithm 5.1 for the line segment connecting ξ_l and ξ_{l+1}. We then observe that

$$\dot{q}(\xi') \;=\; \frac{q(\xi') - q_l}{\xi' - \xi_l} \;<\; \frac{g(\xi') - q_l}{\xi' - \xi_l} \;=\; \frac{q_{l+1} - q_l}{\xi_{l+1} - \xi_l} \;=\; \dot{g}(\xi') \ ,$$

where the first identity follows from our choice of ξ', the inequality directly follows from $q(\xi') < g(\xi')$, and the last two identities hold since B_l is selected by Algorithm 5.1 for the line segment connecting ξ_l and ξ_{l+1}. However, by Lemmas 5.4 and 5.5, B_l is the basis with the minimal slope between ξ_l and ξ_{l+1}, and it thus satisfies

$$\frac{q_{l+1} - q_l}{\xi_{l+1} - \xi_l} \;\leqslant\; \dot{q}(\xi) \ ,$$

which contradicts the strict inequality above. The correctness of the last value $\xi_{T+1} = \infty$, finally, follows since q is constant for large ξ as the constraint $\boldsymbol{w}^\top \boldsymbol{l} = \xi$ is inactive. ■

5.3 Complexity Analysis

A naive implementation of Algorithm 5.1 has a computational complexity of $O(S^2 \log S)$ because it sorts all pairs of indexes $(i, j) \in \mathcal{S} \times \mathcal{S}$ according to their derivatives $\dot{q}$. Although this already constitutes a significant improvement over the theoretical $O(S^{4.5})$ complexity of solving (5.2) using a generic LP solver, we observed numerically that the naive implementation performs on par with state-of-the-art LP solvers. In this section, we describe a simple structural property of the parametric problem (5.2) that allows us to dramatically speed up Algorithm 5.1.

Our improvement is based on the observation that a component $i \in \mathcal{S}$ cannot be a receiver in an optimal basis if there exists another component j that has both a smaller objective coefficient z_j and weight w_j. We call such components i *dominated*, and any dominated receivers can be eliminated from further consideration without affecting the correctness of Algorithm 5.1.

Proposition 5.7. *Consider a component $i \in \mathcal{S}$ such that there is another component $j \in \mathcal{S}$ satisfying $(z_j, w_j) \leqslant (z_i, w_i)$ as well as $(z_j, w_j) \neq (z_i, w_i)$. Then for any basis B in which i acts as receiver, Algorithm 5.1 selects the stepsize $\Delta\xi = 0$.*

Proof. Assume to the contrary that in iteration l, the basis B_l contains i as receiver and Algorithm 5.1 selects a stepsize $\Delta\xi > 0$. Consider $(\xi_{l-1}, \boldsymbol{p}_{l-1}, q_{l-1})$, the parameters at the beginning of iteration l, as well as $(\xi_l, \boldsymbol{p}_l, q_l)$, the parameters at the end of iteration l. To simplify the notation, we use $\mathbf{1}_i$, $i = 1, \ldots, S$ to denote the i-th unit basis vector in $\mathbb{R}^S$.

Let $k \in \mathcal{S}$ be the donor in iteration l. Note that $k \neq j$ as otherwise $\dot{q} \geqslant 0$, which would contradict the construction of the list $\mathcal{D}$. Define δ via $\boldsymbol{p}_l = \boldsymbol{p}_{l-1} + \delta[\mathbf{1}_i - \mathbf{1}_k]$, and note that $\delta > 0$ since $\Delta\xi > 0$. We claim that the alternative parameter setting $(\xi'_l, \boldsymbol{p}'_l, q'_l)$ with $\boldsymbol{p}'_l = \boldsymbol{p}_{l-1} + \delta[\mathbf{1}_j - \mathbf{1}_k]$, $\xi'_l = \|\boldsymbol{p}'_l - \bar{\boldsymbol{p}}\|_{1,\boldsymbol{w}}$ and $q'_l = \boldsymbol{z}^\top \boldsymbol{p}'_l$ satisfies $(\xi'_l, q'_l) \leqslant (\xi_l, q_l)$ and $(\xi'_l, q'_l) \neq (\xi_l, q_l)$. Since this would correspond to a line segment with a steeper decrease than the one constructed by Algorithm 5.1, this contradicts the optimality of Algorithm 5.1 proved in Theorem 5.6. To see that $(\xi'_l, q'_l) \leqslant (\xi_l, q_l)$, note that

$$\xi'_l \;=\; \left\|\boldsymbol{p}'_l - \bar{\boldsymbol{p}}\right\|_{1,\boldsymbol{w}} \;\leqslant\; \|\boldsymbol{p}_l - \bar{\boldsymbol{p}}\|_{1,\boldsymbol{w}} \;=\; \xi_l$$

since $w_j \leqslant w_i$ and $p_i \geqslant \bar{p}_i$ (otherwise, i could not be a receiver). Likewise, we have

$$q'_l \;=\; \boldsymbol{z}^\top \boldsymbol{p}'_l \;\leqslant\; \boldsymbol{z}^\top \boldsymbol{p}_l \;=\; q_l$$

since $z_j \leqslant z_i$. Finally, since $(w_i, z_i) \neq (w_j, z_j)$, at least one of the previous two inequalities must be strict, which implies that $(\xi_l, \boldsymbol{p}_l, q_l)$ is not optimal, a contradiction. ■

One readily verifies that if there are two potential receivers i and j satisfying $w_i = w_j$ and $z_i = z_j$, either one of the receivers can be removed from further consideration without affecting the correctness of Algorithm 5.1. We thus arrive at Algorithm 5.2, which constructs a minimal set of receivers to be considered by Algorithm 5.1 in time $O(S \log S)$.

```
Input: Objective coefficients $z_i$ and weights $w_i$ for all components $i \in \mathcal{S}$
Sort the elements $z_i$ and $w_i$ in non-decreasing order of $z_i$; break ties in
  non-decreasing order of $w_i$ ;
Initialize the set of possible receivers as $\mathcal{R} \leftarrow \{1\}$ ;
for  $i = 2 \dots S$ do
    if  $w_i < \min\{w_k \mid k \in \mathcal{R}\}$ then
        Update $\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$ ;
    end
end
return Possible receivers mapped back to their original positions in $\mathcal{R}$
```

Algorithm 5.2: Identify non-dominated receivers $i \in \mathcal{S}$.

Proposition 5.7 implies that for a uniform $\boldsymbol{w}$, only $i \in \mathcal{S}$ with a minimal component z_i can serve as a receiver, and our homotopy method can be adapted to run in time $O(S \log S)$. This matches the computational complexity of existing fast algorithms for unweighted sa-rectangular L_1-norms (Iyengar, 2005; Petrik and Subramanian, 2014). More generally, if there are C different weight values, then we need to consider at most one receiver for each of the C values.

The following corollary summarizes the combined time complexity of Algorithms 5.1 and 5.2, which are combinatorial. That is they are strongly polynomial and their runtimes that are independent of any optimality tolerance.

Corollary 5.8. *If $|\{w_i \mid i \in \mathcal{S}\}| = C$, then Algorithms 5.1 and 5.2 can be combined to run in time $O(CS \log CS)$ and produce an output of length $T \leqslant CS$.*

6. Computing the Bellman Operator: S-Rectangular Sets

We now develop a bisection scheme to compute the s-rectangular robust Bellman optimality operator $\mathfrak{L}$ defined in (3.7). Our bisection scheme builds on the homotopy method for the sa-rectangular Bellman optimality operator described in the previous section.

The remainder of the section is structured as follows. We first describe the bisection scheme for computing $\mathfrak{L}$ in Section 6.1. Our method does not directly compute the greedy policy required for our PPI from Section 4 but computes the optimal values of some dual variables instead. Section 6.2 describes how to extract the optimal greedy policy from these dual variables. Since our bisection scheme for computing $\mathfrak{L}$ cannot be used to compute the s-rectangular robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ for a fixed policy $\boldsymbol{\pi} \in \Pi$, we describe a different bisection technique for computing $\mathfrak{L}_{\boldsymbol{\pi}}$ in Section 6.3. We use this technique to solve the robust policy evaluation MDP defined in Section 4.

6.1 Bisection Scheme for Robust Bellman Optimality Operator

To simplify the notation, we fix a state $s \in \mathcal{S}$ throughout this section and drop the associated subscripts whenever the context is unambiguous. In particular, we denote the nominal transition probabilities under action a as $\bar{\boldsymbol{p}}_a \in \Delta^S$, the rewards under action a as $\boldsymbol{r}_a \in \mathbb{R}^S$, the L_1-norm weight vector as $\boldsymbol{w}_a \in \mathbb{R}^S$, and the budget of ambiguity as κ. We also fix a value function $\boldsymbol{v}$ throughout this section. We then aim to solve the optimization problem

$$\max_{\boldsymbol{d} \in \Delta^A} \min_{\boldsymbol{\xi} \in \mathbb{R}^A_+} \left\{ \sum_{a \in \mathcal{A}} d_a \cdot q_a(\xi_a) \;\middle|\; \sum_{a \in \mathcal{A}} \xi_a \leqslant \kappa \right\}, \tag{6.1}$$

where $q_a(\xi)$ is defined in (5.1) with subscript $a \in \mathcal{A}$ to identify the associated action. Note that problem (6.1) exhibits a very specific structure: It has a single constraint, and the function q_a is piecewise linear with at most S^2 pieces. We will use this structure to derive an efficient solution scheme that outperforms the naive solution of (6.1) via an LP solver.

Our bisection scheme employs the following reformulation of (6.1):

$$\min_{u \in \mathbb{R}} \left\{ u \;\middle|\; \sum_{a \in \mathcal{A}} q_a^{-1}(u) \leqslant \kappa \right\}, \tag{6.2}$$

where the inverse functions q_a^{-1} are defined as

$$q_a^{-1}(u) = \min_{\boldsymbol{p} \in \Delta^S} \left\{ \|\boldsymbol{p} - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \;\middle|\; \boldsymbol{p}^\top \boldsymbol{z} \leqslant u \right\} \qquad \forall a \in \mathcal{A}. \tag{6.3}$$

Before we formally show that (6.1) and (6.2) are indeed equivalent, we discuss the intuition that underlies the formulation (6.2). In problem (6.1), the adversarial nature chooses the transition probabilities $\boldsymbol{p}_a$, $a \in \mathcal{A}$, to minimize the value of $\sum_{a \in \mathcal{A}} d_a \cdot (\boldsymbol{p}_a^\top \boldsymbol{z})$ while adhering to the ambiguity budget via $\sum_{a \in \mathcal{A}} \xi_a \leqslant \kappa$ for $\xi_a = \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a}$. In problem (6.3), $q_a^{-1}(u)$ can be interpreted as the minimum ambiguity budget $\|\boldsymbol{p} - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a}$ assigned to the action $a \in \mathcal{A}$ that allows nature to ensure that taking an action a results in a robust value $\boldsymbol{p}^\top \boldsymbol{z}$ not exceeding u. Any value of u that is feasible in (6.2) thus implies that within the specified overall ambiguity budget of κ, nature can ensure that *every* action $a \in \mathcal{A}$ results

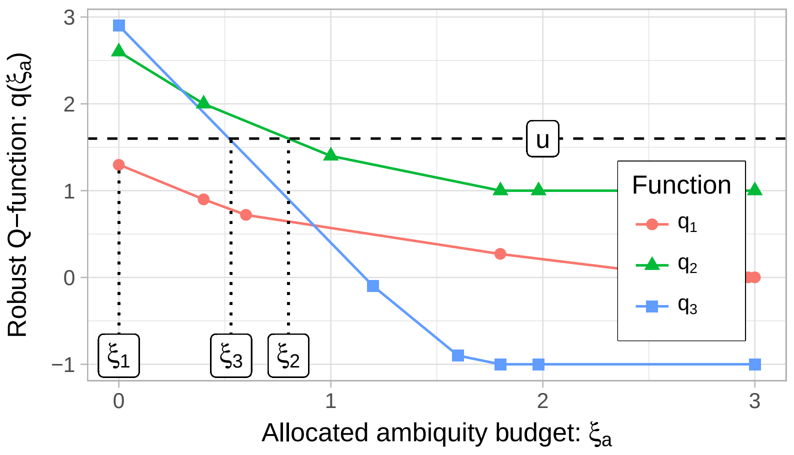

Figure 2: Visualization of the s-rectangular Bellman update with the response functions q_1, q_2, q_3 for 3 actions.

in a robust value not exceeding u. Minimizing u in (6.2) thus determines the transition probabilities that lead to the lowest robust value under *any* policy, which is the same as computing the robust Bellman optimality operator (6.1).

The following example demonstrates the relationship between q and q^{-1} as well as how they are related to the optimization in (6.2).

Example 6.1. *Figure 2 shows an example q-functions q_1, q_2, q_3 for 3 actions. To achieve the robust value of u depicted in the figure, the* smallest *action-wise budgets ξ_a that guarantee $q(\xi_a) \leqslant u$, $i = 1, 2, 3$, are indicated at ξ_1, ξ_2 and ξ_3, resulting in an overall budget of $\kappa = \xi_1 + \xi_2 + \xi_3$.*

We are now ready to state the main result of this section.

Theorem 6.2. *The optimal objective values of* (6.1) *and* (6.2) *coincide.*

Theorem 6.2 relies on the following auxiliary result, which we state first.

Lemma 6.3. *The functions q_a and q_a^{-1} are convex in ξ and u, respectively.*

Proof. The convexity of q_a is immediate from the LP formulation (5.2). The convexity of q_a^{-1} can be shown in the same way by linearizing the objective function in (6.3). ∎

Proof of Theorem 6.2. Since the functions q_a, $a \in \mathcal{A}$, are convex (see Lemma 6.3), we can exchange the maximization and minimization operators in (6.1) to obtain

$$\min_{\boldsymbol{\xi} \in \mathbb{R}_+^A} \left\{ \max_{\boldsymbol{d} \in \Delta^A} \left(\sum_{a \in \mathcal{A}} d_a \cdot q_a(\xi_a) \right) \;\middle|\; \sum_{a \in \mathcal{A}} \xi_a \leqslant \kappa \right\}.$$

Since the inner maximization is linear in $\boldsymbol{d}$, it is optimized at an extreme point of Δ^A. This allows us to re-express the optimization problem as

$$\min_{\boldsymbol{\xi}\in\mathbb{R}^A_+} \left\{ \max_{a\in\mathcal{A}} \{q_a(\xi_a)\} \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa \right\} .$$

We can linearize the objective function in this problem by introducing the epigraphical variable $u \in \mathbb{R}$:

$$\min_{u\in\mathbb{R}} \min_{\boldsymbol{\xi}\in\mathbb{R}^A_+} \left\{ u \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa,\;\; u \geqslant \max_{a\in\mathcal{A}} \{q_a(\xi_a)\} \right\} . \tag{6.4}$$

It can be readily seen that for a fixed u in the outer minimization, there is an optimal $\boldsymbol{\xi}$ in the inner minimization that minimizes each ξ_a individually while satisfying $q_a(\xi_a) \leqslant u$ for all $a \in \mathcal{A}$. Define g_a as the a-th component of this optimal $\boldsymbol{\xi}$:

$$g_a(u) = \min_{\xi_a\in\mathbb{R}_+} \{\xi_a \;\mid\; q_a(\xi_a) \leqslant u\}. \tag{6.5}$$

We show that $g_a(u) = q_a^{-1}(u)$. To see this, we substitute q_a in (6.5) to get:

$$g_a(u) = \min_{\xi_a\in\mathbb{R}_+} \min_{\boldsymbol{p}_a\in\Delta^S} \left\{ \xi_a \;\mid\; \boldsymbol{p}_a^\top \boldsymbol{z}_a \leqslant u,\; \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \leqslant \xi_a \right\} .$$

The identity $g_a = q_a^{-1}$ then follows by realizing that the optimal $\xi_a^\star$ in the equation above must satisfy $\xi_a^\star = \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a}$. Finally, substituting the definition of g_a in (6.5) into the problem (6.4) shows that the optimization problem (6.1) is indeed equivalent to (6.2). ■

```
Input: Desired precision $\epsilon$, functions $q_a^{-1}$, $a \in \mathcal{A}$
    $u_{\min}$: maximum known $u$ for which (6.2) is infeasible,
    $u_{\max}$: minimum known $u$ for which (6.2) is feasible
Output: $\hat{u}$ such that $|u^\star - \hat{u}| \leqslant \epsilon$, where $u^\star$ is optimal in (6.2)
while $u_{\max} - u_{\min} > 2\,\epsilon$ do
    Split interval $[u_{\min}, u_{\max}]$ in half: $u \leftarrow (u_{\min} + u_{\max})/2$;
    Calculate the budget required to achieve the mid point $u$: $s \leftarrow \sum_{a\in\mathcal{A}} q_a^{-1}(u)$ ;
    if $s \leqslant \kappa$ then
        $u$ is feasible: update the feasible upper bound: $u_{\max} \leftarrow u$;
    else
        $u$ is infeasible: update the infeasible lower bound: $u_{\min} \leftarrow u$;
    end
end
return $(u_{\min} + u_{\max})/2$;
```

Algorithm 6.1: Bisection scheme for the robust Bellman optimality operator (3.7)

The bisection scheme for solving problem (6.2) is outlined in Algorithm 6.1. Bisection is a natural and efficient approach for solving the one-dimensional optimization problem. This algorithm is simple and works well in practice, but it can be further improved by leveraging the fact that the functions q_a^{-1}, $a \in \mathcal{A}$, are piecewise linear. In fact, Algorithm 6.1 only solves problem (6.2) to ϵ-optimality, and it requires the choice of a suitable precision ϵ.

We outline how to adapt Algorithm 6.1 to determine the optimal solution to problem (6.2) in quasi-linear time independent of the precision ϵ; please see Algorithm B.1 in Appendix B for details. Recall that Algorithm 5.1 computes the breakpoints $(\xi_t^a)_{t=0,\ldots,T_a+1}$, and objective values $(q_t^a)_{t=0,\ldots,T_a+1}$, $T_a \leqslant S^2$, of each function q_a, $a \in \mathcal{A}$. Then each inverse function q_a^{-1} is also piecewise linear with breakpoints $(q_t^a)_{t=0,\ldots,T_a+1}$, and corresponding function values $\xi_t^a = q_a^{-1}(q_t^a)$. Recall that we define $q_a^{-1}(u) = \infty$ for $u < q_{T_a+1}^a$. We now combine all breakpoints q_t^a, $a \in \mathcal{A}$, to a single list $\mathcal{K}$ in ascending order. We then execute a variant of Algorithm 6.1 in which both $u_{\min}$ and $u_{\max}$ are always set to some breakpoints from $\mathcal{K}$. Instead of choosing the midpoint $u \leftarrow (u_{\min} + u_{\max})/2$ in each iteration of the bisection, we choose the *median* breakpoint between $u_{\min}$ and $u_{\max}$. Using the median instead of the mean breakpoint reduces the method to a *binary search* which runs in logarithmic time. We stop once $u_{\min}$ and $u_{\max}$ are consecutive breakpoints in $\mathcal{K}$, in which case the optimal solution of (6.2) can be computed by basic algebra.

The following statement follows from the discussion above and the results in Appendix B.

Theorem 6.4. *Algorithms 5.1 and 6.1 can be adapted (see Appendix B) to run jointly in $O(S^2 A \log SA)$ time, independent of the optimality tolerance.*

Because each execution of Algorithm 6.1 requires that Algorithm 5.1 is executed to produce its inputs, Theorem 6.4 states the joint complexity of the two algorithms. Using reasoning similar to Corollary 5.8, the bound in Theorem 6.4 can be tightened as follows.

Corollary 6.5. *If $|\{w_i \mid i \in \mathcal{S}\}| = C$, then Algorithms 5.1, 5.2 and 6.1 can be adapted to run jointly in $O(CSA \log CSA)$ time, independent of the optimality tolerance.*

We emphasize that general (interior-point) algorithms for the linear programming formulation of the robust Bellman optimality operator have a theoretical worst-case complexity of $O(S^{4.5} A^{4.5})$ (Karmarkar, 1984); see Appendix C. In addition, the spatial complexity of our algorithms is $O(S^2 A)$ because we need to store at most S^2 breakpoints for each action. Thus, the spatial complexity of our algorithms is linear in the input size since representing a dense transition function also takes $O(S^2 A)$ space.

6.2 Recovering the Greedy Policy

Since Algorithm 6.1 only computes the value of the robust Bellman optimality operator $\mathfrak{L}$ and not an optimal greedy policy $\boldsymbol{d}^\star$ achieving this value, it cannot be used in PPI or related robust policy iteration methods (Iyengar, 2005; Kaufman and Schaefer, 2013) as is. This section describes how to compute an optimal solution $\boldsymbol{d}^\star$ to problem (6.1) from the output of Algorithm 6.1. We again fix a state $s \in \mathcal{S}$ and drop the associated subscripts whenever the context is unambiguous. We also fix a value function $\boldsymbol{v}$ throughout this section. Finally, we assume that $\kappa > 0$; the limiting case $\kappa = 0$ is trivial since the robust Bellman optimality operator then reduces to the nominal Bellman optimality operator.

Recall that Algorithm 6.1 computes the optimal solution $u^\star \in \mathbb{R}$ to problem (6.2), which according to Theorem 6.2 equals the optimal value of problem (6.1). The same argument as in the proof of Theorem 6.2 thus implies that

$$u^\star = \max_{\boldsymbol{d}\in\Delta^A} \min_{\boldsymbol{\xi}\in\mathbb{R}^A_+} \left\{ \sum_{a\in\mathcal{A}} d_a \cdot q_a(\xi_a) \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa \right\} \tag{6.6}$$

$$= \min_{\boldsymbol{\xi}\in\mathbb{R}^A_+} \left\{ \max_{\boldsymbol{d}\in\Delta^A} \sum_{a\in\mathcal{A}} d_a \cdot q_a(\xi_a) \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa \right\} . \tag{6.7}$$

To compute an optimal $\boldsymbol{d}^\star$ from $u^\star$, we first use the definition (6.3) of q_a^{-1} to compute $\boldsymbol{\xi}^\star$ defined as

$$\xi^\star_a = q_a^{-1}(u^\star) \qquad \forall a \in \mathcal{A} . \tag{6.8}$$

Intuitively, the components $\xi^\star_a$ of this vector represent the action-wise uncertainty budgets required to ensure that no greedy policy achieves a robust value that exceeds $u^\star$. The set $\mathcal{C}(\boldsymbol{\xi}^\star) = \{a \in \mathcal{A} \mid q_a(\xi^\star_a) = u^\star\}$ of all actions achieving the optimal robust value plays an important role in the construction of an optimal greedy policy $\boldsymbol{d}^\star$. To this end, the following result collects important properties of $\boldsymbol{\xi}^\star$ and $\mathcal{C}(\boldsymbol{\xi}^\star)$.

Lemma 6.6. *The vector $\boldsymbol{\xi}^\star$ defined in* (6.8) *is optimal in* (6.7). *Moreover, $\mathcal{C}(\boldsymbol{\xi}^\star) \neq \emptyset$ and*

(i) $q_a(\xi^\star_a) = u^\star$ for all $a \in \mathcal{C}(\boldsymbol{\xi}^\star)$;

(ii) $\xi^\star_a = 0$ and $q_a(\xi^\star_a) = \bar{\boldsymbol{p}}_a^\top \boldsymbol{z} \leqslant u^\star$ for all $a \in \mathcal{A}\setminus\mathcal{C}(\boldsymbol{\xi}^\star)$.

Proof. We first argue that $\boldsymbol{\xi}^\star$ is optimal in (6.7). To see that $\boldsymbol{\xi}^\star$ is feasible in (6.7), fix any optimal solution $\bar{\boldsymbol{\xi}} \in \mathbb{R}^A$ in (6.7). The value $q_{a'}(\bar{\xi}_{a'})$ can be upper bounded by the objective in (6.7) as

$$q_{a'}(\bar{\xi}_{a'}) \;\leqslant\; \max_{\boldsymbol{d}\in\Delta^A} \sum_{a\in\mathcal{A}} d_a \cdot q_a(\bar{\xi}_a) \;\leqslant\; u^\star$$

for all $a' \in \mathcal{A}$. The definition of q_a in (5.1) implies that there are $\boldsymbol{p}_a \in \Delta^S$, $a \in \mathcal{A}$, such that

$$\boldsymbol{p}_a^\top \boldsymbol{z} \leqslant u^\star \quad \text{and} \quad \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \leqslant \bar{\xi}_a .$$

The definition of q_a^{-1} in (6.3) implies that each $\boldsymbol{p}_a$ is feasible in $q_a^{-1}(u^\star)$. Thus, each $\xi^\star_a$ is bounded from above by $\bar{\xi}_a$, and we observe that

$$\sum_{a\in\mathcal{A}} \xi^\star_a \;\leqslant\; \sum_{a\in\mathcal{A}} \bar{\xi}_a \;\leqslant\; \kappa . \tag{6.9}$$

Since the definition of q_a^{-1} also implies that $\xi^\star_a = q_a^{-1}(u^\star) \geqslant 0$, $\boldsymbol{\xi}^\star$ is indeed feasible in (6.7). The optimality of $\boldsymbol{\xi}^\star$ in (6.7) then follows from the fact that $q_a(\xi^\star_a) \leqslant u^\star$ because there exists $\boldsymbol{p}^\star_a$ such that $\boldsymbol{z}^\top \boldsymbol{p}^\star_a \leqslant u^\star$ and $\|\boldsymbol{p}^\star_a - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \leqslant \xi^\star_a$, by the definition of $\xi^\star_a$, $a \in \mathcal{A}$.

Next, to show that $\mathcal{C}(\boldsymbol{\xi}^\star) \neq \emptyset$, note that for all $a \in \mathcal{A}$, we have

$$q_a(\xi^\star_a) = q_a(q_a^{-1}(u^\star)) = \min_{\boldsymbol{p}_1\in\Delta^S} \left\{ \boldsymbol{p}_1^\top \boldsymbol{z} \;\middle|\; \|\boldsymbol{p}_1 - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \leqslant \min_{\boldsymbol{p}_2\in\Delta^S} \left\{ \|\boldsymbol{p}_2 - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \;\middle|\; \boldsymbol{p}_2^\top \boldsymbol{z} \leqslant u^\star \right\} \right\}$$

by the definitions of q_a and q_a^{-1} in (5.1) and (6.3), respectively. Recognizing that any optimal solution $\boldsymbol{p}^\star_2$ to the inner minimization is feasible in the outer minimization leads to

$$q_a(\xi^\star_a) \;\leqslant\; (\boldsymbol{p}^\star_2)^\top \boldsymbol{z} \;\leqslant\; u^\star .$$

Given the inequality above, $\mathcal{C}(\boldsymbol{\xi}^\star) = \varnothing$ only if $q_a(\xi_a^\star) < u^\star$, for all $a \in \mathcal{A}$. Imagine now that $\mathcal{C}(\boldsymbol{\xi}^\star) = \varnothing$. Then combining the equality in (6.6), the inequality in $\xi^\star$ (6.9) and $q_a(\xi_a^\star) < u^\star$ leads to

$$u^\star \;=\; \max_{\boldsymbol{d}\in\Delta^A} \min_{\boldsymbol{\xi}\in\mathbb{R}_+^A} \left\{ \sum_{a\in\mathcal{A}} d_a \cdot q_a(\xi_a) \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa \right\} \;\leqslant\; \max_{\boldsymbol{d}\in\Delta^A} \sum_{a\in\mathcal{A}} d_a \cdot q_a(\xi_a^\star) \;<\; u^\star \;,$$

which is a contradiction.

The statement *(i)* that $q_a(\xi_a^\star) = u^\star$ for all $a \in \mathcal{C}(\boldsymbol{\xi}^\star)$ now follows immediately from the definition of $\mathcal{C}(\boldsymbol{\xi}^\star)$. To see that $\xi_a^\star = 0$ for $a \in \mathcal{A}\backslash\mathcal{C}(\boldsymbol{\xi}^\star)$ in statement *(ii)*, assume to the contrary that $\xi_a^\star > 0$ for some $a \in \mathcal{A}\backslash\mathcal{C}(\boldsymbol{\xi}^\star)$. Since $q_a(\xi_a^\star) < u^\star$, there is $\boldsymbol{p}_a^\star \in \Delta^S$ optimal in (6.3) satisfying $(\boldsymbol{p}_a^\star)^\top \boldsymbol{z} < u^\star$ and $\|\boldsymbol{p}_a^\star - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} \leqslant \xi_a^\star$. At the same time, since $\xi_a^\star > 0$, we have $\|\boldsymbol{p}_a^\star - \bar{\boldsymbol{p}}_a\|_{1,\boldsymbol{w}_a} > 0$ as well. This implies, however, that there is $\epsilon > 0$ such that $\boldsymbol{p}_a^\star + \epsilon \cdot (\bar{\boldsymbol{p}}_a - \boldsymbol{p}_a^\star)$ is feasible in (6.3) and achieves a lower objective value than $\boldsymbol{p}_a^\star$, which contradicts the optimality of $\boldsymbol{p}_a^\star$ in (6.3). We thus conclude that $\xi_a^\star = 0$ for $a \in \mathcal{A}\backslash\mathcal{C}(\boldsymbol{\xi}^\star)$. This immediately implies that $q_a(\xi_a^\star) = \bar{\boldsymbol{p}}_a^\top \boldsymbol{z}$ for all $a \in \mathcal{A}\backslash\mathcal{C}(\boldsymbol{\xi}^\star)$ as well.

Finally, the fact stated in *(ii)* that $q_a(\xi_a^\star) \leqslant u^\star$ for all $a \in \mathcal{A}\backslash\mathcal{C}(\boldsymbol{\xi}^\star)$ has already been shown earlier in the proof. ■

The construction of $\boldsymbol{d}^\star \in \Delta^A$ relies on the slopes of q_a, which are piecewise constant but discontinuous at the breakpoints of q_a. However, the functions q_a are convex by Lemma 6.3, and therefore their subdifferentials (Rockafellar, 1970) $\partial q_a(\xi_a)$ exist for all $\xi_a \geqslant 0$. Using these subdifferentials, we construct optimal action probabilities $\boldsymbol{d}^\star \in \Delta^A$ from $\boldsymbol{\xi}^\star$ as follows.

(i) If $0 \in \partial q_{\bar{a}}(\xi_{\bar{a}}^\star)$ for some $\bar{a} \in \mathcal{C}(\boldsymbol{\xi}^\star)$, define $\boldsymbol{d}^\star$ as

$$d_a^\star = \begin{cases} 1 & \text{if } a = \bar{a} \\ 0 & \text{otherwise} \end{cases} \qquad \forall a \in \mathcal{A}\ . \tag{6.10a}$$

(ii) If $0 \notin \partial q_a(\xi_a^\star)$ for all $a \in \mathcal{C}(\boldsymbol{\xi}^\star)$, define $\boldsymbol{d}^\star$ as

$$d_a^\star = \frac{e_a}{\sum_{a'\in\mathcal{A}} e_{a'}} \quad \text{with} \quad e_a = \begin{cases} -\frac{1}{f_a} & \text{if } a \in \mathcal{C}(\boldsymbol{\xi}^\star) \\ 0 & \text{otherwise} \end{cases} \qquad \forall a \in \mathcal{A}\ , \tag{6.10b}$$

where f_a can be any element from $\partial q_a(\xi_a^\star)$, $a \in \mathcal{A}$.

The choice of $\boldsymbol{d}^\star$ may not be unique as there may be multiple $\bar{a} \in \mathcal{C}(\boldsymbol{\xi}^\star)$ that satisfy the first condition, and the choice of $f_a \in \partial q_a(\xi_a^\star)$ in the second condition may not be unique either.

Theorem 6.7. *Any vector $\boldsymbol{d}^\star$ satisfying* (6.10a) *or* (6.10b) *is optimal in problem* (6.1). *Moreover, for $\boldsymbol{\xi}^\star$ defined in* (6.8)*, $(\boldsymbol{d}^\star, \boldsymbol{\xi}^\star)$ is a saddle point in* (6.1).

Proof. One readily verifies that $\boldsymbol{d}^\star$ satisfying (6.10a) is contained in Δ^A. To see that $\boldsymbol{d}^\star \in \Delta^A$ for $\boldsymbol{d}^\star$ satisfying (6.10b), we note that $\mathcal{C}(\boldsymbol{\xi}^\star)$ is non-empty due to Lemma 6.6 and that $f_a < 0$ and thus $e_a > 0$ since q_a is non-increasing. To see that $\boldsymbol{d}^\star$ satisfying (6.10a) or (6.10b) is optimal in (6.1), we show that it achieves the optimal objective value $u^\star$:

$$\min_{\boldsymbol{\xi}\in\mathbb{R}_+^A} \left\{ \sum_{a\in\mathcal{A}} d_a^\star \cdot q_a(\xi_a) \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa \right\} \geqslant u^\star\ . \tag{6.11}$$

Observe that $u^\star$ is indeed achieved for $\boldsymbol{\xi} = \boldsymbol{\xi}^\star$ since

$$\sum_{a\in\mathcal{A}} d_a^\star \cdot q_a(\xi_a^\star) \;=\; \sum_{a\in\mathcal{C}(\xi^\star)} d_a^\star \cdot q_a(\xi_a^\star) \;=\; \sum_{a\in\mathcal{C}(\xi^\star)} d_a^\star \cdot u^\star = u^\star \ .$$

Here, the first equality holds since $d_a^\star = 0$ for $a \notin \mathcal{C}(\boldsymbol{\xi}^\star)$, the second equality follows from the definition of $\mathcal{C}(\boldsymbol{\xi}^\star)$, and the third equality follows from $\boldsymbol{d}^\star \in \Delta^A$.

To establish the inequality (6.11), we show that $\boldsymbol{\xi}^\star$ is optimal in (6.11). This also proves that $(\boldsymbol{d}^\star, \boldsymbol{\xi}^\star)$ is a saddle point of problem (6.1). We denote by $\partial_{\boldsymbol{\xi}}(f)[\boldsymbol{\xi}^\star]$ the subdifferential of a convex function f with respect to $\boldsymbol{\xi}$, evaluated at $\boldsymbol{\xi} = \boldsymbol{\xi}^\star$. The KKT conditions for non-differentiable convex programs (see, for example, Theorem 28.3 of Rockafellar 1970), which are sufficient for the optimality of $\boldsymbol{\xi}^\star$ in the minimization on the left-hand side of (6.11), require the existence of a scalar $\lambda^\star \geqslant 0$ and a vector $\boldsymbol{\alpha}^\star \in \mathbb{R}_+^A$ such that

$$\mathbf{0} \in \partial_{\boldsymbol{\xi}} \left(\sum_{a\in\mathcal{A}} d_a^\star \cdot q_a(\xi_a) - \lambda^\star \left(\kappa - \sum_{a\in\mathcal{A}} \xi_a \right) - \sum_{a\in\mathcal{A}} \alpha_a^\star \cdot \xi_a \right) [\boldsymbol{\xi}^\star] \qquad \text{[Stationarity]}$$

$$\lambda^\star \cdot \left(\kappa - \sum_{a\in\mathcal{A}} \xi_a^\star \right) = 0, \quad \alpha_a^\star \cdot \xi_a^\star = 0 \;\; \forall a \in \mathcal{A} \qquad \text{[Compl. Slackness]}$$

The stationarity condition simplifies using the chain rule to

$$0 \in d_a^\star \cdot \partial q_a(\xi_a^\star) + \lambda^\star - \alpha_a^\star \qquad \forall a \in \mathcal{A} \ . \tag{6.12}$$

If $\boldsymbol{d}^\star$ satisfies (6.10a), then both (6.12) and complementary slackness are satisfied for $\lambda^\star = 0$ and $\boldsymbol{\alpha}^\star = \mathbf{0}$. On the other hand, if $\boldsymbol{d}^\star$ satisfies (6.10b), we set

$$\lambda^\star = \frac{1}{\sum_{a\in\mathcal{C}(\boldsymbol{\xi}^\star)} e_a}, \qquad \alpha_a^\star = 0 \quad \forall a \in \mathcal{C}(\boldsymbol{\xi}^\star), \qquad \alpha_a^\star = \lambda^\star \quad \forall a \in \mathcal{A}\backslash\mathcal{C}(\boldsymbol{\xi}^\star) \ ,$$

where e_a is defined in (6.10b). This solution satisfies $\lambda^\star \geqslant 0$ and $\boldsymbol{\alpha} \geqslant \mathbf{0}$ because $f_a \leqslant 0$ and therefore $e_a \geqslant 0$. This solution satisfies (6.12), and Lemma 6.6 implies that the second complementary slackness condition is satisfied as well. To see that the first complementary slackness condition is satisfied, we argue that $\sum_{a\in\mathcal{A}} \xi_a^\star = \kappa$ under the conditions of (6.10b). Assume to the contrary that $\sum_{a\in\mathcal{A}} \xi_a^\star < \kappa$. Since $0 \notin \partial q_a(\xi_a^\star)$ and the sets $\partial q_a(\xi_a^\star)$ are closed for all $a \in \mathcal{C}(\boldsymbol{\xi}^\star)$ (see page 215 and Theorem 23.4 of Rockafellar 1970), we have

$$\exists \bar{\beta}_a > 0 \quad \text{such that} \quad q_a(\xi_a^\star + \beta_a) < q_a(\xi_a) \;\; \forall \beta_a \in (0, \bar{\beta}_a)$$

for all $a \in \mathcal{C}(\boldsymbol{\xi}^\star)$. We can thus marginally increase each component $\xi_a^\star$, $a \in \mathcal{C}(\boldsymbol{\xi}^\star)$, to obtain a new solution to problem (6.7) that is feasible and that achieves a strictly lower objective value than $u^\star$. This, however, contradicts the optimality of $u^\star$. We thus conclude that $\sum_{a\in\mathcal{A}} \xi_a^\star = \kappa$, that is, the first complementary slackness condition is satisfied as well. ■

The values $\boldsymbol{\xi}^\star$ and $\boldsymbol{d}^\star$ can be computed in time $O(A \log S)$ since they rely on the quantities $q_a(\xi_a^\star)$ and $q_a^{-1}(u^\star)$ that have been computed previously by Algorithm 5.1 and Algorithm 6.1, respectively. The worst-case transition probabilities can also be retrieved from the minimizers of q_a defined in (5.1) since, as Theorem 6.7 implies, $\boldsymbol{\xi}^\star$ is optimal in the minimization problem in (6.1).

6.3 Bisection Scheme for Robust Bellman Policy Update

Recall that the robust policy evaluation MDP $(\mathcal{S}, \bar{\mathcal{A}}, \boldsymbol{p}_0, \bar{\boldsymbol{p}}, \bar{\boldsymbol{r}}, \gamma)$ defined in Section 4 has continuous action sets $\bar{\mathcal{A}}(s) = \mathcal{P}_s$, $s \in \mathcal{S}$, and the transition function $\bar{\boldsymbol{p}}$ and the rewards $\bar{\boldsymbol{r}}$ defined as

$$\bar{\boldsymbol{p}}_{s,\boldsymbol{\alpha}} = \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{\alpha}_a \quad \text{and} \quad \bar{r}_{s,\boldsymbol{\alpha}} = -\sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{\alpha}_a^\top \boldsymbol{r}_{s,a} \ .$$

To solve this MDP via value iteration or (modified) policy iteration, we must compute the Bellman optimality operator $\mathfrak{L}$ defined as

$$\begin{aligned}
(\mathfrak{L}\boldsymbol{v})_s &= \max_{\boldsymbol{\alpha}\in\mathcal{P}_s} \left\{ \bar{r}_{s,\boldsymbol{\alpha}} + \gamma \cdot \bar{\boldsymbol{p}}_{s,\boldsymbol{\alpha}}^\top \boldsymbol{v} \right\} \\
&= \max_{\boldsymbol{\alpha}\in(\Delta^S)^A} \left\{ \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{\alpha}_a^\top (\gamma \cdot \boldsymbol{v} - \boldsymbol{r}_{s,a}) \;\middle|\; \sum_{a\in\mathcal{A}} \|\boldsymbol{\alpha}_a - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_s \right\} \\
&= -\min_{\boldsymbol{\alpha}\in(\Delta^S)^A} \left\{ \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{\alpha}_a^\top (\boldsymbol{r}_{s,a} - \gamma \cdot \boldsymbol{v}) \;\middle|\; \sum_{a\in\mathcal{A}} \|\boldsymbol{\alpha}_a - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_s \right\} \ . \qquad (6.13)
\end{aligned}$$

The continuous action space in this MDP makes it impossible to compute $\mathfrak{L}\boldsymbol{v}$ by simply enumerating the actions. The ordinary Bellman operator could be solved as a linear program, but this suffers from the same computational limitations as its application to the robust Bellman operator described earlier. Using similar ideas as in Section 6.1, we can re-express the minimization problem in (6.13) as

$$\min_{\boldsymbol{\xi}\in\mathbb{R}_+^A} \left\{ \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot q_{s,a}(\xi_a) \;\middle|\; \sum_{a\in\mathcal{A}} \xi_a \leqslant \kappa_s \right\}, \qquad (6.14)$$

where the function $q_{s,a} : \mathbb{R}_+ \to \mathbb{R}$ is defined for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$ as

$$q_{s,a}(\xi) \;=\; \min_{\boldsymbol{p}\in\Delta^S} \left\{ \boldsymbol{p}^\top (\boldsymbol{r}_{s,a} - \gamma \cdot \boldsymbol{v}) \;\middle|\; \|\boldsymbol{p} - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \xi \right\} \ .$$

Note that this definition of $q_{s,a}$ corresponds to (5.1) with a different choice of $\boldsymbol{z}$.

At the first glance, problem (6.14) seems to be a special case of problem (6.1) from Section 6.1, and one may posit that it can be solved using Algorithm 6.1. Unfortunately, this is not the case. The lack of optimization over $\boldsymbol{d}$ precludes the transformations employed in Theorem 6.2. However, problem (6.14) can still be solved efficiently by taking advantage of the fact that it only contains a single constraint on $\boldsymbol{\xi}$ and that the functions $q_{s,a}$ are piecewise linear and convex. To see this, note that the Lagrangian of (6.14) is

$$\max_{\lambda\in\mathbb{R}_+} \min_{\boldsymbol{\xi}\in\mathbb{R}_+^A} \left\{ \sum_{a\in\mathcal{A}} (\pi_{s,a} \cdot q_{s,a}(\xi_a)) + \lambda \cdot \boldsymbol{1}^\top \boldsymbol{\xi} - \lambda\, \kappa_s \right\},$$

where the use of strong duality (Vanderbei, 1998) is justified since (6.14) can be reformulated as a feasible linear program. The minimization can now be decomposed by actions:

$$\max_{\lambda\in\mathbb{R}_+} u(\lambda) = \max_{\lambda\in\mathbb{R}_+} \left\{ \sum_{a\in\mathcal{A}} \min_{\xi_a\in\mathbb{R}_+} \left\{ \pi_{s,a} \cdot q_{s,a}(\xi_a) + \lambda \xi_a \right\} - \lambda\, \kappa_s \right\} \qquad (6.15)$$

The inner minimization problems over ξ_a, $a \in \mathcal{A}$, are convex and can be solved exactly by bisection since the functions $q_{s,a}$ are piecewise linear. Likewise, the maximization over λ can be solved exactly by bisection since the function u is concave and piecewise linear. The lower bound on λ for the bisection method is 0. A sufficient upper bound on λ is a value ν such that $-\nu \leqslant \partial_\xi \, q_{s,a}(0)$ for all $a \in \mathcal{A}$. As can be readily seen from (6.15), $u(\lambda) \leqslant u(\nu)$ for any $\lambda \geqslant \nu$. Finally, the optimal ξ in (6.14) can be recovered using a method similar to the one described in Section 6.2.

7. Numerical Evaluation

We now compare the runtimes of PPI (Algorithm 4.1) combined with the homotopy method (Algorithm 5.1) and the bisection method (Algorithm 6.1) with the runtime of a naive approach that combines the robust value iteration with a computation of the robust Bellman optimality operator $\mathfrak{L}$ using a general LP solver. We use Gurobi 9.0, a state-of-the-art commercial optimization package. All algorithms were implemented in C++, parallelized using the OpenMP library, and used the Eigen library to perform linear algebra operations. The algorithms were compiled with GCC 9.3 and executed on an AMD Ryzen 9 3900X CPU with 64GB RAM. The source code of the implementation is available at `http://github.com/marekpetrik/craam2`.

7.1 Experimental Setup

Our experiments involve two problems from different domains with a fundamentally different structure. The two domains are the *inventory management* problem (Zipkin, 2000; Porteus, 2002) and the *cart-pole* problem (Lagoudakis and Parr, 2003). The inventory management problem has many actions and dense transition probabilities. The cart-pole problem, on the other hand, has only two actions and sparse transition probabilities. More actions and dense transition probabilities make for much more challenging computation of the Bellman update compared to policy evaluation.

Next, we give a high-level description of both problems as well as our parameter choice. Because the two domains serve simply as benchmark problems and their full description would be lengthy, we only outline their motivation, construction, and properties. To facilitate the reproducibility of the domains, CSV files with the precise specification of the RMDPs being solved are available at `http://github.com/marekpetrik/PPI_paper`.

In our inventory management problem, a retailer orders, stores and sells a single product over an infinite time horizon. Any orders submitted in a time period t are fulfilled at the beginning of time period $t+1$ and are subject to deterministic fixed and variable costs. Any items held in the limited-capacity inventory incur deterministic per-period holding costs. The per-unit sales price is deterministic, but the per-period demand is stochastic. All accrued demand in time period t is satisfied up to the available inventory. Any remaining unsatisfied demand is backlogged at a per-unit backlogging penalty up to a given limit. The states and actions of our MDP represent the inventory levels and the order quantities in any given time period, respectively. The stochastic demands drive the stochastic state transitions. The rewards are the sales revenue minus the purchase costs in each period.

In our experiments, the fixed and variable ordering costs are 5.99 and 1.0, respectively. The inventory holding and backlogging costs are 0.1 and 0.15, respectively. We vary the inventory capacity I to study the impact of the problem's size on the runtimes, while the backlog limit is $I/3$. We also impose an upper limit of $I/2$ on each order. The corresponding MDP thus has $I + I/3 = 4/3 \cdot I$ states and $I/2$ actions. Note that due to the inventory capacity limits, not all actions are available at every state. The unit sales price is 1.6. The demand in each period follows the Gaussian distribution with a mean of $I/2$ and a standard deviation of $I/5$ and is rounded to the closest integer. We use a discount factor of 0.995.

In our cart-pole problem, a pole has to be balanced upright on top of a cart that moves along a single dimension. At any point in time, the state of the system is described by four continuous quantities: the cart's position and velocity, as well as the pole's angle and angular velocity. To balance the pole, one can apply a force to the cart from the left or from the right. The resulting MDP thus accommodates a 4-dimensional continuous state space and two actions. Several different implementations of this problem can be found in the literature; in the following, we employ the deterministic implementation from the OpenAI Gym. Again, we use a discount factor of 0.995.

Since the state space of our cart-pole problem is continuous, we discretize it to be amenable to our solution methods. The discretization follows a standard procedure in which random samples from the domain are subsampled to represent the discretized state space. The transitions are then estimated from samples that are closest to each state. In other words, the probability of transitioning from a discretized state s to another discretized state s' is proportional to the number of sampled transitions that originate near s and end up near s'. The discretized transition probabilities are no longer deterministic, even though the original problem transitions are.

The ambiguity sets are modified slightly in this section to ensure a more realistic evaluation. Assuming that the robust transition can be positive to any state of the RMDP can lead to overly conservative policies. To obtain less conservative policies, we restrict our ambiguity sets $\mathcal{P}_{s,a}$ and $\mathcal{P}_s$ from Section 3 to probability distributions that are *absolutely continuous* with respect to the nominal distributions $\bar{\boldsymbol{p}}_{s,a}$. Our sa-rectangular ambiguity sets $\mathcal{P}_{s,a}$ thus become

$$\mathcal{P}_{s,a} = \left\{ \boldsymbol{p} \in \Delta^S \;\middle|\; \|\boldsymbol{p} - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_{s,a},\;\; p_{s'} \leqslant \left\lceil \bar{p}_{s,a,s'} \right\rceil \;\; \forall s' \in \mathcal{S} \right\} ,$$

and we use a similar construction for our s-rectangular ambiguity sets $\mathcal{P}_s$. We set the ambiguity budget to $\kappa_{s,a} = 0.2$ and $\kappa_s = 1.0$ in the sa-rectangular and s-rectangular version of our inventory management problem, respectively, and we set $\kappa_{s,a} = \kappa_s = 0.1$ in our cart-pole problem. Anecdotally, the impact of the ambiguity budget on the runtimes is negligible. We report separate results for uniform weights $\boldsymbol{w}_{s,a} = \mathbf{1}$ and non-uniform weights $\boldsymbol{w}_{s,a}$ that are derived from the value function $\boldsymbol{v}$. In the latter case, we choose weights $(\boldsymbol{w}_{s,a})_{s'}$ that are proportional to $|v_{s'} - \mathbf{1}^\top \boldsymbol{v}/S|$, which have been shown to optimize the percentile criterion for uncertain MDPs (Behzadian et al., 2021). All weights $\boldsymbol{w}_{s,a}$ are normalized so that their values are contained in $[0, 1]$. Note that the simultaneous scaling of $\boldsymbol{w}_{s,a}$ and $\kappa_{s,a}$ does not affect the solution.

Recall that the policy evaluation step in PPI can be accomplished by any MDP solution method. In our inventory management problem, whose instances have up to 1,000 states, we use policy iteration and solve the arising systems of linear equations via the LU decomposition of the Eigen library (Puterman, 2005). This approach does not scale well to MDPs with $S \gg 1,000$ states as the policy iteration manipulates matrices of di-

			SA-rectangular		S-rectangular	
Problem	Ambiguity	States	LP Solver	Alg. 5.1	LP Solver	Alg. 6.1
Inventory	Uniform	100	13.96	0.02	24.67	0.06
Inventory	Weighted	100	13.85	0.75	21.36	0.86
Inventory	Uniform	500	583.20	0.36	1,715.94	19.65
Inventory	Weighted	500	440.35	20.69	655.00	36.24
Inventory	Uniform	1,000	> 10,000.00	20.00	> 10,000.00	51.97
Inventory	Weighted	1,000	4,071.47	109.27	3,752.21	163.32
Cart-pole	Uniform	1,000	9.50	0.18	19.85	1.94
Cart-pole	Weighted	1,000	12.70	1.93	32.80	1.90
Cart-pole	Uniform	2,000	12.81	1.90	13.33	1.88
Cart-pole	Weighted	2,000	12.04	2.03	13.08	1.95
Cart-pole	Uniform	4,000	23.39	1.91	23.29	1.76
Cart-pole	Weighted	4,000	19.96	2.05	21.16	2.14

Table 2: Runtime (in seconds) required by different algorithms to compute 200 steps of the robust Bellman optimality operator.

mension $S \times S$. Therefore, in our cart-pole problem, whose instances have $1,000$ or more states, we use modified policy iteration (Puterman, 2005) instead. We compare the performance of our algorithms to the robust value iteration as well as the robust modified policy iteration (RMPI) of Kaufman and Schaefer (2013). Recall that in contrast to PPI, RMPI evaluates robust policies through a fixed number of value iteration steps. Since the impact of the number of value iteration steps on the overall performance of RMPI is not well understood, we fix this number to $1,000$ throughout our experiments. Finally, we set $\epsilon_{k+1} = \min\{\gamma^2 \epsilon_k, 0.5/(1-\gamma) \cdot \|\mathfrak{L}_{\boldsymbol{\pi}_k} \boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty\}$ in Algorithm 4.1, which satisfies the convergence condition in Theorem 4.5.

7.2 Results and Discussion

Table 2 reports the runtimes required by our homotopy method (Algorithm 5.1), our bisection method (Algorithm 6.1) and Gurobi (LP Solver) to compute 200 steps of the robust Bellman optimality operator $\mathfrak{L}$ across all states $s \in \mathcal{S}$. We fixed the number of Bellman evaluations in this experiment to clearly separate the speedups achieved by a quicker evaluation of the Bellman operator itself, studied in this experiment, from the speedups obtained by using PPI in place of value iteration, studied in the next experiment. The computations are parallelized over all available threads via OpenMP using Jacobi-style value iteration (Puterman, 2005). By construction, all algorithms identify the same optimal solutions in each application of the Bellman operator. The computations were terminated after $10,000$ seconds.

There are several important observations we can make from the results in Table 2. First of all, that our algorithms outperform Gurobi by an order of magnitude for weighted ambiguity sets and by two orders of magnitude for uniform (unweighted) ambiguity sets,

			SA-rectangular			S-rectangular	
Problem	Ambiguity	States	VI	RMPI	PPI	VI	PPI
Inventory	Uniform	100	0.12	0.03	0.01	3.52	0.15
Inventory	Weighted	100	10.28	0.94	0.14	15.02	1.02
Inventory	Uniform	500	1.39	0.06	0.14	24.69	2.71
Inventory	Weighted	500	140.53	5.69	2.11	276.63	16.76
Inventory	Uniform	1,000	8.65	0.23	0.59	217.90	13.98
Inventory	Weighted	1,000	393.90	14.36	6.90	519.21	163.18
Cart-pole	Uniform	1,000	0.03	0.06	0.03	0.80	0.15
Cart-pole	Weighted	1,000	0.25	0.17	0.04	0.98	0.28
Cart-pole	Uniform	10,000	0.32	0.26	0.13	8.40	1.06
Cart-pole	Weighted	10,000	1.72	1.13	0.21	13.43	3.52
Cart-pole	Uniform	20,000	0.44	0.54	0.29	16.24	2.40
Cart-pole	Weighted	20,000	6.37	3.22	0.62	28.50	9.30

Table 3: Runtime (in seconds) required by different algorithms to compute an approximately optimal robust value function.

independent of the type of rectangularity. This impressive performance is because the inventory management problem has many actions, which makes computing the Bellman operator particularly challenging. The computation time also reflects that homotopy and bisection methods have quasi-linear time complexities when used with uniform L_1 norms. It is remarkable that even with the simple cart-pole problem our algorithms are about 10 to 20 times faster than a state-of-the-art LP solver. In fact, our results indicate that even moderately-sized RMDPs may be practically intractable when solved with generic LP solvers.

Table 3 reports the runtimes required by the parallelized versions of the robust value iteration (VI), the robust modified policy iteration (RMPI) and our partial policy iteration (PPI) to solve our inventory management and cart-pole problems to approximate optimality. To this end, we choose a precision of $\delta = 40$ (that is, $\|\mathfrak{L}_{\boldsymbol{\pi}_k}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty \leqslant 0.1$), as defined in Algorithm 4.1, for our inventory management problem, as well as a smaller precision of $\delta = 4$ (that is, $\|\mathfrak{L}_{\boldsymbol{\pi}_k}\boldsymbol{v}_k - \boldsymbol{v}_k\|_\infty \leqslant 0.01$) for our cart-pole problem, to account for the smaller rewards in this problem. All algorithms use the homotopy (Algorithm 5.1) and the bisection method (Algorithm 6.1) to compute the robust Bellman optimality operator. Note that RMPI is only applicable to sa-rectangular ambiguity sets. The computations were terminated after $10,000$ seconds.

There are also several important observations we can make from the results in Table 3. As one would expect, PPI in RMDPs behaves similarly to policy iteration in MDPs. It outperforms value iteration in essentially all benchmarks, being almost up to 100 times faster, but the margin varies significantly. The improvement margin depends on the relative complexity of policy improvements and evaluations. In the sa-rectangular cart-pole problem, for example, the policy improvement step is relatively cheap, and thus the benefit of employing a policy evaluation is small. The situation is reversed in the s-rectangular inventory management problem, in which the policy improvement step is very time-consuming. PPI outperforms the robust value iteration most significantly in the sa-rectangular inventory management problem since the policy evaluation step is much cheaper than the policy improvement step due to the large number of available actions. RMPI's performance, on the other hand, is more varied: while it sometimes outperforms the other methods, it is usually dominated by at least one of the competing algorithms. We attribute this fact to the inefficient value iteration that is employed in the robust policy evaluation step of RMPI. It is important to emphasize that PPI has the same theoretical convergence rate as the robust value iteration, and thus its performance relative to the robust value iteration and RMPI will depend on the specific problem instance and as well as the employed parameter settings.

In conclusion, our empirical results show that our proposed combination of PPI and the homotopy or bisection method achieves a speedup of up to four orders of magnitude for both sa-rectangular and s-rectangular ambiguity sets when compared with the state-of-the-art solution approach that combines a robust value iteration with a computation of the robust Bellman operator via a commercial LP solver. Since our methods scale more favorably with the size of the problem, their advantage is likely to only increase with larger problems that what we considered here.

8. Conclusion

We proposed three new algorithms to solve robust MDPs over L_1-ball uncertainty sets. Our homotopy algorithm computes the robust Bellman operator over sa-rectangular L_1-ball uncertainty sets in quasi-linear time and is thus almost as efficient as computing the ordinary Bellman operator. Our bisection scheme utilizes the homotopy algorithm to compute the robust Bellman operator over s-rectangular L_1-ball uncertainty sets, again in quasi-linear time. Both algorithms can be combined with PPI, which generalizes the highly efficient modified policy iteration scheme to robust MDPs. Our numerical results show significant speedups of up to four orders of magnitude over a leading LP solver for both sa-rectangular and s-rectangular ambiguity sets.

Our research opens up several promising avenues for future research. First, our homotopy method sorts the bases of problem (5.2) in quasi-linear time. This step could also be implemented in linear time using a variant of the *quickselect* algorithm, which has led to improvements in a similar context (Condat, 2016). Second, we believe that the techniques presented here can be adapted to other uncertainty sets, such as L_∞- and L_2-balls around the nominal transition probabilities or uncertainty sets based on ϕ-divergences. Both the efficient implementation of the resulting algorithms as well as the empirical comparison of different uncertainty sets on practical problem instances would be of interest. Third, it is important to study how our methods generalize to robust value function approximation methods (Tamar et al., 2014). Finally, we believe that the study of robust MDPs under the average reward setting is another interesting direction for future research.

Acknowledgments

We thank Bruno Scherrer for pointing out the connections between policy iteration and algorithms for solving zero-sum games and Stephen Becker for insightful comments. This work was supported by the National Science Foundation under Grants No. IIS-1717368 and IIS-1815275, by the Engineering and Physical Sciences Research Council under Grant No. EP/R045518/1, by the City University of Hong Kong (Project No. 9610481 and 7005534), and by the National Natural Science Foundation of China (Project No. 72032005). Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the funding bodies.

Appendix A. Properties of Robust Bellman Operator

We prove several fundamental properties of the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ and the robust Bellman optimality operator $\mathfrak{L}$ over s-rectangular and sa-rectangular ambiguity sets.

Lemma A.1. *For both s-rectangular and sa-rectangular ambiguity sets, the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ and the robust Bellman optimality operator $\mathfrak{L}$ are γ-contractions under the L_∞-norm, that is*

$$\|\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{x} - \mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{y}\|_\infty \leqslant \gamma \|\boldsymbol{x}-\boldsymbol{y}\|_\infty \qquad \text{and} \qquad \|\mathfrak{L}\boldsymbol{x} - \mathfrak{L}\boldsymbol{y}\|_\infty \leqslant \gamma \|\boldsymbol{x}-\boldsymbol{y}\|_\infty \ .$$

The equations $\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{v} = \boldsymbol{v}$ and $\mathfrak{L}\boldsymbol{v} = \boldsymbol{v}$ have the unique solutions $\boldsymbol{v}_{\boldsymbol{\pi}}$ and $\boldsymbol{v}^\star$, respectively.

Proof. See Theorem 3.2 of Iyengar (2005) for sa-rectangular sets and Theorem 4 of Wiesemann et al. (2013) for s-rectangular sets. ■

Lemma A.2. *For both s-rectangular and sa-rectangular ambiguity sets, the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ and the robust Bellman optimality operator $\mathfrak{L}$ are monotone:*

$$\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{x} \geqslant \mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{y} \quad \text{and} \quad \mathfrak{L}\boldsymbol{x} \geqslant \mathfrak{L}\boldsymbol{y} \qquad \forall \boldsymbol{x} \geqslant \boldsymbol{y} \ .$$

Proof. We show the statement for s-rectangular ambiguity sets; the proof of sa-rectangular uncertainty sets is analogous. Consider $\boldsymbol{\pi} \in \Pi$ as well as $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^S$ such that $\boldsymbol{x} \geqslant \boldsymbol{y}$ and define

$$F_s(\boldsymbol{p}, \boldsymbol{x}) = \sum_{a\in\mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{x}) \ .$$

The monotonicity of the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ follows from the fact that

$$(\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{x})_s = \min_{\boldsymbol{p}\in\mathcal{P}_s} F_s(\boldsymbol{p}, \boldsymbol{x}) = F_s(\boldsymbol{p}^\star, \boldsymbol{x}) \geqslant F_s(\boldsymbol{p}^\star, \boldsymbol{y}) \overset{\text{(a)}}{\geqslant} (\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{y})_s \qquad \forall s \in \mathcal{S} \ ,$$

where $\boldsymbol{p}^\star \in \arg\min_{\boldsymbol{p}\in\mathcal{P}_s} F_s(\boldsymbol{p}, \boldsymbol{x})$. The inequality (a) holds because $F_s(\boldsymbol{p}^\star, \cdot)$ is monotone since $\boldsymbol{p}^\star \geqslant \boldsymbol{0}$.

To prove the monotonicity of the robust Bellman optimality operator $\mathfrak{L}$, consider again some $\boldsymbol{x}$ and $\boldsymbol{y}$ with $\boldsymbol{x} \geqslant \boldsymbol{y}$ and let $\boldsymbol{\pi}^\star$ be the greedy policy satisfying $\mathfrak{L}\boldsymbol{y} = \mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{y}$. We then have that

$$(\mathfrak{L}\boldsymbol{y})_s = (\mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{y})_s \leqslant (\mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{x})_s \leqslant (\mathfrak{L}\boldsymbol{x})_s,$$

where the inequalities follow from the (previously shown) monotonicity of $\mathfrak{L}_{\boldsymbol{\pi}^\star}$ and the fact that $(\mathfrak{L}\boldsymbol{x})_s = (\max_{\boldsymbol{\pi}\in\Pi} \mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{x})_s \geqslant (\mathfrak{L}_{\boldsymbol{\pi}^\star}\boldsymbol{x})_s$. ■

Lemmas A.1 and A.2 further imply the following two properties of $\mathfrak{L}_{\boldsymbol{\pi}}$ and $\mathfrak{L}$.

Corollary A.3. *For both s-rectangular and sa-rectangular ambiguity sets, the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ and the robust Bellman optimality operator $\mathfrak{L}$ satisfy $\boldsymbol{v}^\star \geqslant \boldsymbol{v}_{\boldsymbol{\pi}}$ for each $\boldsymbol{\pi} \in \Pi$.*

Proof. The corollary follows from the monotonicity (Lemma A.2) and contraction properties (Lemma A.1) of $\mathfrak{L}$ and $\mathfrak{L}_{\boldsymbol{\pi}}$ using standard arguments. See, for example, Proposition 2.1.2 in Bertsekas (2013). ■

Corollary A.4. *For both s-rectangular and sa-rectangular ambiguity sets, the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ and the robust Bellman optimality operator $\mathfrak{L}$ satisfy for any $\boldsymbol{v} \in \mathbb{R}^S$ that*

$$\|\boldsymbol{v}^\star - \boldsymbol{v}\|_\infty \leqslant \frac{1}{1-\gamma}\,\|\mathfrak{L}\boldsymbol{v} - \boldsymbol{v}\|_\infty \quad \textit{and} \quad \|\boldsymbol{v}_{\boldsymbol{\pi}} - \boldsymbol{v}\|_\infty \leqslant \frac{1}{1-\gamma}\,\|\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{v} - \boldsymbol{v}\|_\infty\ .$$

Proof. The corollary follows from the monotonicity (Lemma A.2) and contraction properties (Lemma A.1) of $\mathfrak{L}$ and $\mathfrak{L}_{\boldsymbol{\pi}}$ using standard arguments. See, for example, Proposition 2.1.1 in Bertsekas (2013). ■

We next show that both $\mathfrak{L}_{\boldsymbol{\pi}}$ and $\mathfrak{L}$ are invariant when adding a constant to the value function.

Lemma A.5. *For both s-rectangular and sa-rectangular ambiguity sets, the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ and the robust Bellman optimality operator $\mathfrak{L}$ are translation invariant for each $\boldsymbol{\pi} \in \Pi$:*

$$\mathfrak{L}_{\boldsymbol{\pi}}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) = \mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1} \quad \textit{and} \quad \mathfrak{L}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) = \mathfrak{L}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1} \qquad \forall \boldsymbol{v} \in \mathbb{R}^S,\ \forall \epsilon \in \mathbb{R}$$

Proof. We show the statement for s-rectangular ambiguity sets; the proof of sa-rectangular uncertainty sets is analogous. Fixing $\boldsymbol{\pi} \in \Pi$, $\boldsymbol{v} \in \mathbb{R}^S$ and $\epsilon \in \mathbb{R}$, we have

$$\begin{aligned}
(\mathfrak{L}_{\boldsymbol{\pi}}(\boldsymbol{v} + \epsilon\mathbf{1}))_s &= \min_{\boldsymbol{p}\in\mathcal{P}_s} \sum_{a\in\mathcal{A}} \pi_{s,a}\cdot \boldsymbol{p}_a^\top(\boldsymbol{r}_{s,a} + \gamma\cdot[\boldsymbol{v} + \epsilon\cdot\mathbf{1}]) \\
&= \min_{\boldsymbol{p}\in\mathcal{P}_s} \sum_{a\in\mathcal{A}} \pi_{s,a}\cdot (\boldsymbol{p}_a^\top(\boldsymbol{r}_{s,a} + \gamma\cdot\boldsymbol{v}) + \gamma\epsilon) \\
&= \gamma\epsilon + \min_{\boldsymbol{p}\in\mathcal{P}_s} \sum_{a\in\mathcal{A}} \pi_{s,a}\cdot \boldsymbol{p}_a^\top(\boldsymbol{r}_{s,a} + \gamma\cdot\boldsymbol{v})\ ,
\end{aligned}$$

where the first identity holds by definition of $\mathfrak{L}_{\boldsymbol{\pi}}$, the second is due to the fact that $\boldsymbol{p}_a^\top\mathbf{1} = 1$ since $\mathcal{P}_s \subseteq (\Delta^S)^A$, and the third follows from the fact that $\sum_{a\in\mathcal{A}} \pi_{s,a} = 1$.

To see that $\mathfrak{L}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) = \mathfrak{L}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1}$, we note that

$$\mathfrak{L}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) = \mathfrak{L}_{\boldsymbol{\pi}^1}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) = \mathfrak{L}_{\boldsymbol{\pi}^1}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1} \leqslant \mathfrak{L}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1}\ ,$$

where $\boldsymbol{\pi}^1 \in \Pi$ is the greedy policy that satisfies $\mathfrak{L}_{\boldsymbol{\pi}^1}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) = \mathfrak{L}(\boldsymbol{v} + \epsilon\cdot\mathbf{1})$, as well as

$$\mathfrak{L}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1} = \mathfrak{L}_{\boldsymbol{\pi}^2}\boldsymbol{v} + \gamma\epsilon\cdot\mathbf{1} = \mathfrak{L}_{\boldsymbol{\pi}^2}(\boldsymbol{v} + \epsilon\cdot\mathbf{1}) \leqslant \mathfrak{L}(\boldsymbol{v} + \epsilon\cdot\mathbf{1})\ ,$$

where $\boldsymbol{\pi}^2 \in \Pi$ is the greedy policy that satisfies $\mathfrak{L}_{\boldsymbol{\pi}^2}\boldsymbol{v} = \mathfrak{L}\boldsymbol{v}$. ■

Our last result in this section shows that the difference between applying the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ to two value functions can be bounded from below by a linear function.

Lemma A.6. *For both s-rectangular and sa-rectangular ambiguity sets, there exists a stochastic matrix $\boldsymbol{P}$ such that the robust Bellman policy update $\mathfrak{L}_{\boldsymbol{\pi}}$ satisfies*

$$\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{x} - \mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{y} \geqslant \gamma \cdot \boldsymbol{P}(\boldsymbol{x} - \boldsymbol{y}) \ ,$$

for each $\boldsymbol{\pi} \in \Pi$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^S$.

Proof. We show the statement for s-rectangular ambiguity sets; the proof of sa-rectangular uncertainty sets is analogous. We have that

$$\begin{aligned}
(\mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{x} - \mathfrak{L}_{\boldsymbol{\pi}}\boldsymbol{y})_s &= \min_{\boldsymbol{p} \in \mathcal{P}_s} \left\{ \sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{x}) \right\} - \min_{\boldsymbol{p} \in \mathcal{P}_s} \left\{ \sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{y}) \right\} \\
&\geqslant \min_{\boldsymbol{p} \in \mathcal{P}_s} \left\{ \sum_{a \in \mathcal{A}} \left(\pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{x}) \right) - \sum_{a \in \mathcal{A}} \left(\pi_{s,a} \cdot \boldsymbol{p}_a^\top (\boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{y}) \right) \right\} \\
&= \min_{\boldsymbol{p} \in \mathcal{P}_s} \left\{ \sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \gamma \cdot \boldsymbol{p}_a^\top (\boldsymbol{x} - \boldsymbol{y}) \right\} \ .
\end{aligned}$$

The result follows by constructing a stochastic matrix $\boldsymbol{P}$ such that its s-th row is $\sum_{a \in \mathcal{A}} \pi_{s,a} \cdot \boldsymbol{p}_a^\top$ where $\boldsymbol{p}_a$ is the optimizer in the last minimization above. ■

Appendix B. Bisection Algorithm with Quasi-Linear Time Complexity

We adapt Algorithm 6.1 to determine the optimal solution to problem (6.2) in quasi-linear time without dependence on any precision ϵ. Recall that Algorithm 5.1 computes the breakpoints $(\xi_t^a)_t$, $t = 0, \ldots, T_a + 1$ and objective values $(q_t^a)_t$, $t = 0, \ldots, T_a + 1$, $T_a \leqslant S^2$, of each function q_a, $a \in \mathcal{A}$. Moreover, each inverse function q_a^{-1} is also piecewise linear with breakpoints $(q_t^a)_t$, $t = 0, \ldots, T_a + 1$ and corresponding function values $\xi_t^a = q_a^{-1}(q_t^a)$, as well as $q_a^{-1}(u) = \infty$ for $u < q_{T_a+1}^a$. We use this data as input for our revised bisection scheme in Algorithm B.1.

Algorithm B.1 first combines all breakpoints q_t^a, $t = 0, \ldots T_a + 1$ and $a \in \mathcal{A}$, of the inverse functions q_a^{-1}, $a \in \mathcal{A}$, to a single list $\mathcal{K}$ in ascending order. It then bisects on the indices of these breakpoints. The result is a breakpoint pair $(k_{\min}, k_{\max})$ satisfying $k_{\max} = k_{\min} + 1$ as well as $\kappa \in \left[\sum_{a \in \mathcal{A}} q_a^{-1}(\hat{q}_{k_{\min}}), \sum_{a \in \mathcal{A}} q_a^{-1}(\hat{q}_{k_{\max}})\right]$. Since none of the functions q_a^{-1} have a breakpoint between $\hat{q}_{k_{\min}}$ and $\hat{q}_{k_{\max}}$, finding the optimal solution $u^\star$ to problem (3.7) then reduces to solving a single linear equation in one unknown, which is done in the last part of Algorithm B.1.

The complexity of Algorithm B.1 is dominated by the merging of the sorted lists $(q_t^a)_{t=0,\ldots T_a+1}$, $a \in \mathcal{A}$, as well as the computation of s inside the while-loop. Merging A sorted lists, each of size less than or equal to CS, can be achieved in time $O(CSA \log A)$. However, each one of these lists needs to be also sorted in Algorithm 5.1 giving the overall complexity of $O(CSA \log CSA)$. Then, computing q_a^{-1} at a given point can be achieved in

```
**Input:** Breakpoints $(q^a_t)_{t=0,\ldots,T_a+1}$, of all functions $q_a$, $a \in \mathcal{A}$
**Output:** The optimal solution $u^\star$ to the problem (6.2)
Combine $q^a_t$, $t = 0, \ldots, T_a$ and $a \in \mathcal{A}$, to a single list $\mathcal{K} = (\hat{q}_1, \ldots, \hat{q}_K)$ in ascending
  order, omitting any duplicates ;

// Bisection search to find the optimal line segment $(k_{\min}, k_{\max})$
$k_{\min} \leftarrow 1$; $k_{\max} \leftarrow K$ ;
**while** $k_{\max} - k_{\min} > 1$ **do**
    Split $\{k_{\min}, \ldots, k_{\max}\}$ in half: $k \leftarrow \text{round}((k_{\min} + k_{\max})/2)$ ;
    Calculate the budget required to achieve $u = \hat{q}_k$: $s \leftarrow \sum_{a \in \mathcal{A}} q_a^{-1}(\hat{q}_k)$ ;
    **if** $s \leqslant \kappa$ **then**
        $u = \hat{q}_k$ is *feasible*: update the feasible upper bound: $k_{\max} \leftarrow k$ ;
    **else**
        $u = \hat{q}_k$ is *infeasible*: update the infeasible lower bound: $k_{\min} \leftarrow k$ ;
    **end**
**end**

// All $q_a^{-1}$ are affine on $(\hat{q}_{k_{\min}}, \hat{q}_{k_{\max}})$
$u_{\min} \leftarrow \hat{q}_{k_{\min}}$;                $u_{\max} \leftarrow \hat{q}_{k_{\max}}$ ;
$s_{\min} \leftarrow \sum_{a \in \mathcal{A}} q_a^{-1}(u_{\min})$; $s_{\max} \leftarrow \sum_{a \in \mathcal{A}} q_a^{-1}(u_{\max})$ ;
$\alpha \leftarrow (\kappa - s_{\min})/(s_{\max} - s_{\min})$ ;
$u^\star \leftarrow (1 - \alpha) \cdot u_{\min} + \alpha \cdot u_{\max}$;
**return** $u^\star$
```

Algorithm B.1: Quasi-linear time bisection scheme for solving (3.7)

time $O(\log CS)$, so that s in an individual iteration of the while-loop can be computed in time $O(A \log CS)$. Since the while-loop is executed $O(\log CSA)$ many times, computing s has an overall complexity of $O(A \log CS \log CSA)$. We thus conclude that Algorithm B.1 has a complexity of $O(CSA \log A + A \log CS \log CSA)$.

Appendix C. Computing the Bellman Operator via Linear Programming

In this section we present an LP formulation for the robust s-rectangular Bellman optimality operator $\mathfrak{L}$ defined in (3.7):

$$(\mathfrak{L}\boldsymbol{v})_s = \max_{\boldsymbol{d}\in\Delta^A} \min_{\boldsymbol{p}\in(\Delta^S)^A} \left\{ \sum_{a\in\mathcal{A}} d_a \cdot \boldsymbol{p}_a^\top \boldsymbol{z}_a \;\middle|\; \sum_{a\in\mathcal{A}} \|\boldsymbol{p}_a - \bar{\boldsymbol{p}}_{s,a}\|_{1,\boldsymbol{w}_{s,a}} \leqslant \kappa_s \right\}$$

Here, we use $\boldsymbol{z}_a = \boldsymbol{r}_{s,a} + \gamma \cdot \boldsymbol{v}$ in the objective function. Employing an epigraph reformulation, the inner minimization problem can be re-expressed as the following linear program:

$$\begin{array}{lll}
\underset{\boldsymbol{p}\in\mathbb{R}^{A\times S},\,\boldsymbol{\theta}\in\mathbb{R}^{A\times S}}{\text{minimize}} & \sum_{a\in\mathcal{A}} d_a \cdot \boldsymbol{z}_a^\top \boldsymbol{p}_a & \\
\text{subject to} & \boldsymbol{1}^\top \boldsymbol{p}_a = 1 & \forall a\in\mathcal{A} \quad [x_a] \\
& \boldsymbol{p}_a - \bar{\boldsymbol{p}}_a \geqslant -\boldsymbol{\theta}_a & \forall a\in\mathcal{A} \quad [y_a^n] \\
& \bar{\boldsymbol{p}}_a - \boldsymbol{p}_a \geqslant -\boldsymbol{\theta}_a & \forall a\in\mathcal{A} \quad [y_a^p] \\
& -\sum_{a\in\mathcal{A}} \boldsymbol{w}_a^\top \boldsymbol{\theta}_a \geqslant -\kappa & \qquad\quad\; [\lambda] \\
& \boldsymbol{p} \geqslant \boldsymbol{0}, \quad \boldsymbol{\theta} \geqslant \boldsymbol{0} &
\end{array}$$

For ease of exposition, we have added the dual variables corresponding to each constraint in brackets. This linear program is feasible by construction, which implies that its optimal value coincides with the optimal value of its dual. We can thus dualize this linear program and combine it with the outer maximization to obtain the following linear programming reformulation of the the robust s-rectangular Bellman optimality operator $\mathfrak{L}$:

$$\begin{array}{lll}
\underset{\substack{\boldsymbol{d}\in\mathbb{R}^A,\boldsymbol{x}\in\mathbb{R}^A,\,\lambda\in\mathbb{R} \\ \boldsymbol{y^p}\in\mathbb{R}^{S\times A},\boldsymbol{y^n}\in\mathbb{R}^{S\times A}}}{\text{maximize}} & \sum_{a\in\mathcal{A}} \left(x_a + \bar{\boldsymbol{p}}_a^\top [\boldsymbol{y}_a^n - \boldsymbol{y}_a^p] \right) - \kappa \cdot \lambda & \\
\text{subject to} & \boldsymbol{1}^\top \boldsymbol{d} = 1, \quad \boldsymbol{d} \geqslant \boldsymbol{0} & \\
& -\boldsymbol{y}_a^p + \boldsymbol{y}_a^n + x \cdot \boldsymbol{1} \leqslant d_a z_a & \forall a\in\mathcal{A} \\
& \boldsymbol{y}_a^p + \boldsymbol{y}_a^n - \lambda \cdot \boldsymbol{w}_a \leqslant \boldsymbol{0} & \forall a\in\mathcal{A} \\
& \boldsymbol{y}^p \geqslant \boldsymbol{0} \quad \boldsymbol{y}^n \geqslant \boldsymbol{0} & \\
& \lambda \geqslant 0 &
\end{array}$$

This problem has $O(SA)$ variables and an input bitlength of $O(SA)$. As such, its theoretical runtime complexity is $O(S^{4.5}A^{4.5})$ (Karmarkar, 1984).

References

M. S. Asif and J. Romberg. Dantzig selector homotopy with dynamic measurements. In *IS&T/SPIE Computational Imaging*, 2009.

B. Behzadian, R. Russel, M. Petrik, and C. P. Ho. Optimizing percentile criterion using robust MDPs. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

D. P. Bertsekas. *Abstract Dynamic Programming.* Athena Scientific, 2013.

D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997.

L. Condat. Fast projection onto the Simplex and the l1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

A. Condon. On algorithms for simple stochastic games. *Advances in Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–71, 1993.

K. V. Delgado, L. N. De Barros, D. B. Dias, and S. Sanner. Real-time dynamic programming for Markov decision processes with imprecise probabilities. *Artificial Intelligence*, 230: 192–223, 2016.

E. Derman, D. Mankowitz, T. Mann, and S. Mannor. A Bayesian approach to robust reinforcement learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2019.

I. Drori and D. Donoho. Solution of l1 minimization problems by LARS/homotopy methods. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2006.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *International Conference of Machine Learning (ICML)*, 2008.

P. J. Garrigues and L. El Ghaoui. A homotopy algorithm for the lasso with online observations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 489–496, 2009.

M. Geist and B. Scherrer. Anderson acceleration for reinforcement learning. In *European Workshop on Reinforcement Learning*, 2018.

R. Givan, S. Leach, and T. Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122(1):71–109, 2000.

V. Goyal and J. Grand-Clement. Robust Markov decision process: Beyond rectangularity, 2018.

V. Goyal and J. Grand-Clement. A first-order approach to accelerated value iteration, 2019.

G. Hanasusanto and D. Kuhn. Robust data-driven dynamic programming. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

T. Hansen, P. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM)*, 60(1):1–16, 2013.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2nd edition, 2009.

C. P. Ho, M. Petrik, and W. Wiesemann. Fast Bellman updates for robust MDPs. In *International Conference on Machine Learning (ICML)*, pages 1979–1988, 2018.

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

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(1):1563–1600, 2010.

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

D. L. Kaufman and A. J. Schaefer. Robust modified policy iteration. *INFORMS Journal on Computing*, 25(3):396–410, 2013.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Y. Le Tallec. *Robust, Risk-Sensitive, and Data-driven Control of Markov Decision Processes.* PhD thesis, MIT, 2007.

S. Mannor, O. Mebel, and H. Xu. Lightning does not strike twice: Robust MDPs with coupled uncertainty. In *International Conference on Machine Learning (ICML)*, 2012.

S. Mannor, O. Mebel, and H. Xu. Robust MDPs with k-rectangular uncertainty. *Mathematics of Operations Research*, 41(4):1484–1509, 2016.

K. Murphy. *Machine Learning: A Probabilistic Perspective.* Springer, 2012.

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

M. Petrik. Approximate dynamic programming by minimizing distributionally robust bounds. In *International Conference of Machine Learning (ICML)*, 2012.

M. Petrik and D. Subramanian. RAAM: The benefits of robustness in approximating aggregated MDPs in reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2014.

M. Petrik, Mohammad Ghavamzadeh, and Y. Chow. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

E. L. Porteus. *Foundations of Stochastic Inventory Theory.* Stanford Business Books, 2002.

I. Post and Y. Ye. The simplex method is strongly polynomial for deterministic Markov decision processes. *Mathematics of Operations Research*, 40(4):859–868, 2015.

M. Puterman and M. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11):1127–1137, 1978.

M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming.* Wiley-Interscience, 2005.

M. L. Puterman and S. L. Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4(1):60–69, 1979.

R. T. Rockafellar. Convex Analysis, 1970.

R. H. Russell and M. Petrik. Beyond confidence regions: Tight Bayesian ambiguity sets for robust MDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

J. Satia and R. Lave. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21:728–740, 1973.

A. L. Strehl, L. Li, and M. Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.

M. A. Taleghan, T. G. Dietterich, M. Crowley, K. Hall, and H. J. Albers. PAC Optimal MDP Planning with Application to Invasive Species Management. *Journal of Machine Learning Research*, 16:3877–3903, 2015.

A. Tamar, S. Mannor, and H. Xu. Scaling up Robust MDPs Using Function Approximation. In *International Conference of Machine Learning (ICML)*, 2014.

J. Thai, C. Wu, A. Pozdnukhov, and A. Bayen. Projected sub-gradient with l1 or simplex constraints via isotonic regression. In *IEEE Conference on Decision and Control (CDC)*, pages 2031–2036, 2015.

E. van den Berg and M. P. Friedlander. Sparse Optimization with Least-Squares Constraints. *SIAM Journal on Optimization*, 21(4):1201–1229, 2011.

R. J. Vanderbei. *Linear Programming: Foundations and Extensions.* Springer, 1998.

T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger. Inequalities for the L1 deviation of the empirical distribution, 2003.

C. White and H. Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42(4):739–749, 1994.

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

R. J. R. Williams and L. C. L. Baird. Tight performance bounds on greedy policies based on imperfect value functions. In *Yale Workshop on Adaptive and Learning Systems.* Northeastern University, 1993.

H. Xu and S. Mannor. The robustness-performance tradeoff in Markov decision processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

H. Xu and S. Mannor. Parametric regret in uncertain Markov decision processes. In *IEEE Conference on Decision and Control (CDC)*, pages 3606–3613, 2009.

J. Zhang, B. O'Donoghue, and S. Boyd. Globally convergent type-I Anderson acceleration for nonsmooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4):3170–3197, 2020.

P. H. Zipkin. *Foundations of Inventory Management.* McGraw-Hill, 2000.