Computers and Operations Research 126 (2021) 105108

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/caor

Policy-based branch-and-bound for infinite-horizon Multi-model Markov decision processes

Vinayak S. Ahluwalia[a], Lauren N. Steimle[b,*], Brian T. Denton[c]

[a] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA
[b] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[c] Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA

A R T I C L E I N F O

Article history:
Received 7 January 2020
Revised 14 August 2020
Accepted 23 September 2020
Available online 11 October 2020

Keywords:
Markov decision processes
Parameter uncertainty
Branch-and-bound

A B S T R A C T

Markov decision processes (MDPs) are models for sequential decision-making that inform decision making in many fields, including healthcare, manufacturing, and others. However, the optimal policy for an MDP may be sensitive to the reward and transition parameters which are often uncertain because parameters are typically estimated from data or rely on expert opinion. To address parameter uncertainty in MDPs, it has been proposed that multiple models of the parameters be incorporated into the solution process, but solving these problems can be computationally challenging. In this article, we propose a policy-based branch-and-bound approach that leverages the structure of these problems and numerically compare several important algorithmic designs. We demonstrate that our approach outperforms existing methods on test cases from the literature including randomly generated MDPs, a machine maintenance MDP, and an MDP for medical decision making.

1. Introduction

MDPs are used to model sequential decision-making under uncertainty in many fields, including healthcare, machine maintenance, inventory control, and finance (Boucherie and Van Dijk, 2017; Puterman, 1994). MDPs are stochastic control processes whereby a decision maker (DM) seeks to maximize rewards over a planning horizon. In this article, we consider expected discounted rewards over an infinite horizon, where the Expectation depends on the DM's decisions, the transition probabilities, and rewards which describe the stochastic reward process. However, the transition probability and reward parameters are uncertain because they are typically estimated from data, synthesized from systematic reviews of the literature, or based on expert opinion. The optimal decisions may depend on such parameters used in the optimization process, leading to ambiguity in what the DM should do.

Recent efforts have sought to improve sequential decision-making by directly incorporating parameter uncertainty into MDPs. One proposed approach is the Multi-model Markov decision process (MMDP) wherein the DM considers multiple models of the MDP's parameters in the solution (Steimle et al., 2018; Buchholz and Scheftelowitsch, 2019). Solving an MMDP involves finding the policy that performs the best with respect to the weighted average of the policy's performance in each model, and this problem has been shown to be NP-hard for infinite-horizon MMDPs (Buchholz and Scheftelowitsch, 2019) and finite-horizon MMDPs (Steimle et al., 2018). Both Steimle et al. (2018) and Buchholz and Scheftelowitsch (2019) proposed mixed-integer program (MIP) formulations to solve these problems when considering the class of stationary (Markov) deterministic policies. However, it has been shown that this formulation struggles to scale to larger problem instances.

To address the computational challenge of infinite-horizon MMDPs, Buchholz and Scheftelowitsch (2019) considered heuristics. Meraklı and Küçükyavuz (2019) proposed a MIP formulation for an extension of the MMDP in which the DM may be risk-averse to parameter ambiguity. In the finite-horizon setting, Steimle et al. (2019) designed a policy-based branch-and-bound (B&B) algorithm that can solve MMDPs faster than the previously proposed MIP formulation (Steimle et al., 2018). The B&B approach for solving finite-horizon MMDPs relies on exploring promising partial Markov deterministic policies until the optimal solution is found. To obtain bounds for partial policies, their algorithm solves a relaxation on each node in the B&B tree in which each model is independently solved using backwards induction. Nodes are enumerated in the tree by fixing certain actions in certain state-time pairs and then employing traditional B&B pruning methods until the optimal solution is found. Although this approach worked well for finite-horizon MDPs with parameter ambiguity, it is an open question whether a similar B&B approach would also outperform MIP-based solution methods for infinite-horizon MMDPs.

* Corresponding author.
E-mail addresses: vahluw@umich.edu (V.S. Ahluwalia), steimle@gatech.edu (L.N. Steimle), btdenton@umich.edu (B.T. Denton).

https://doi.org/10.1016/j.cor.2020.105108
0305-0548/© 2020 Elsevier Ltd. All rights reserved.

In this article, we consider a policy-based branch-and-bound (PB-B&B) approach for solving infinite-horizon MMDPs and examine different options for the algorithmic design. We consider different node selection strategies including depth-first search (DFS), breadth-first search (BrFS), or best-first search (BeFS). Different strategies for branching in the PB-B&B tree are also considered. We also explore the use of exact and approximate solution methods for solving the relaxation and obtaining upper bounds at each node, such as policy iteration and value iteration (see Chapter 6 of Puterman (1994) for a description of these methods). We analyze the computational performance of the PB-B&B with respect to the stopping criteria for these alternative approaches. More specifically, we consider the trade-off between solving the relaxation using a very small convergence parameter to obtain a tighter upper bound and the additional computational effort required to do so.

We then compare the best-performing PB-B&B design to the current standard for solving these problems, a MIP formulation implemented using a commercial solver. We compare the PB-B&B and MIP-based solution methods on three sets of test instances. The first set is comprised of MMDPs in which all parameters are sampled randomly (Buchholz and Scheftelowitsch, 2019). The second set is a study of the optimal time to repair a deteriorating machine under uncertainty in the transition probabilities which describe the deterioration process (Delage and Mannor, 2010). The third set is based on a model of the optimal time to start treatment for a patient with HIV under multiple plausible models of the natural history of HIV and the remaining life expectancy after beginning treatment (Shechter et al., 2008). We show that the PB-B&B approach outperforms the MIP-based approach on larger problems, but that using a MIP formulation may be sufficient for small scale instances.

In summary, the main contributions of this work are as follows:

- We propose the first PB-B&B method for solving infinite-horizon MDPs with multiple models of the parameters.
- We propose and evaluate algorithmic designs for the PB-B&B method that are unique to infinite-horizon MDPs.
- We demonstrate the performance of our PB-B&B on 3 different case studies of MMDPs and we show that the best algorithmic design outperforms the existing MIP approaches for solving these instances.
- We illustrate that the PB-B&B algorithm is easily modified to support a percentile optimization objective and outperforms the corresponding MIP formulation.

The remainder of this article is organized as follows. In Section 2, we state the infinite-horizon MMDP after we provide background on infinite-horizon MDPs relevant to the PB-B&B algorithm. In Section 3, we describe our algorithmic approach and several design considerations, and in Section 4, we compare these designs numerically. In Section 5, we demonstrate the effectiveness of our PB-B&B algorithm on three different MDPs from the literature. In Section 6, we conclude with a discussion of the main contributions of our study, and we propose ideas for future work.

2. Problem statement

In this section, we present the problem setting in which we apply our PB-B&B algorithm. We begin by describing the standard infinite-horizon MDP, and we then describe the infinite-horizon MMDP. For brevity, we refer to these as the MDP and MMDP from this point forward.

The stochastic process for a standard MDP is described by a discrete-time Markov chain over a set of *states*, $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$. The initial state of the system is determined by the initial state distribution vector, $\mu \in [0,1]^{|\mathcal{S}|}$, such that $\mu(s)$ represents the probability that the MDP will start in state s. At each *decision epoch* $t \in \mathcal{T} := \{1, 2, \ldots\}$, the DM observes the state of the Markov chain, s, and specifies a control *action* $a \in \mathcal{A}$. Then, the Markov chain will evolve stochastically such that the state of the system at time $t+1$ will be s' with probability $p(s'|s,a) \in [0,1]$. When implementing action a in state s, the DM receives a reward of $r(s,a)$, and rewards are discounted at a rate $\lambda \in [0,1)$. We consider the setting where all rewards and transition probabilities are stationary meaning that they are independent of time. We summarize the parameters of an MDP using the tuple $(\mathcal{S}, \mathcal{A}, R, P, \mu)$ where $R \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ and $P \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}$ denote the rewards and transition probabilities, respectively.

The DM uses a *policy* to determine which actions to take. In general, a policy may depend on the entire history of the MDP, but it is well-known that stationary Markov policies are optimal for infinite-horizon MDPs with stationary transition probabilities and rewards (Puterman, 1994, §6.2). Therefore, we consider the class of stationary deterministic policies of the form $\pi : \mathcal{S} \mapsto \mathcal{A}$. When the DM implements a policy π, the expected discounted reward to the DM will be:

$$\mathbb{E}^{\mu,P,R,\pi}\left[\sum_{t=1}^{\infty} \lambda^{t-1} r(s,a)\right], \tag{1}$$

which depends on the MDP parameters, R and P. The optimal policy is obtained as a solution to the following well-known *optimality equations*:

$$v(s) = \max_{a\in\mathcal{A}} \left\{ r(s,a) + \lambda \sum_{s'\in\mathcal{S}} p(s'|s,a)\, v(s') \right\}, \quad \forall s \in \mathcal{S}. \tag{2}$$

In this article, we specifically focus on infinite-horizon MDPs. Infinite-horizon MDPs are widely used to model controlled stochastic processes with stationary rewards and transition probabilities and long time-horizons relative to the decision epoch (Puterman, 1994, Ch. 6). Thus, infinite-horizon models are often appropriate for stochastic control processes such as inventory control and machine maintenance. Because the search space is limited to stationary policies, infinite-horizon MDPs tend to be easier to solve than their finite-horizon counterparts. Reducing the search space offers computational benefits and several algorithms leverage this when determining an optimal solution to the set of equations in (2), such as value iteration, policy iteration, modified policy iteration, and linear programming. We refer the reader to Chapter 6 of Puterman (1994) for the details of these methods.

2.1. The multi-model Markov decision process

The MMDP framework includes a finite collection of models of the MDP, $\mathcal{M} = \{1, \ldots, |\mathcal{M}|\}$. Each model is itself an MDP defined on the same state space and action space, with reward and transition probabilities possibly dependent on the model: $(\mathcal{S}, \mathcal{A}, R^m, P^m, \mu)$. We let $r^m(s,a)$ denote the model-specific reward of taking action a in state s in the MDP m. Similarly, $p^m(s'|s,a)$ denotes the model-specific probability of transitioning from state s to s' by taking action a.

Multiple objectives can be considered in the MMDP framework to design policies that account for multiple models of the parameters including maximizing minimum model value, minimizing maximum model regret, and percentile optimization (Steimle et al., 2019; Meraklı and Küçükyavuz, 2019). In this article, we focus on the *weighted value problem* (WVP). In the WVP, each model m is assigned a weight $\alpha_m \in (0,1)$ such that $\sum_{m\in\mathcal{M}} \alpha_m = 1$. The goal of the DM in this setting is to select a policy that maximizes the weighted average of the objective functions from each model. That is, if policy π achieves the following value function in model m:

$$v_\pi^m = \mathbb{E}^{\mu,P^m,R^m,\pi}\left[\sum_{t=1}\lambda^{t-1}r^m(s,a)\right],$$

then the weighted value of policy π is given by:

$$W(\pi) = \sum_{m\in\mathcal{M}}\alpha_m v_\pi^m. \tag{3}$$

The goal is to find a policy $\pi \in \Pi$ that maximizes the weighted value:

$$W^* = \max_{\pi\in\Pi} W(\pi)$$

where Π is the class of stationary deterministic policies. Stationary deterministic policies are desirable due to their ease of implementation and interpretability, and others have similarly searched over this policy class to design policies for MMDPs (Buchholz and Scheftelowitsch, 2019; Meraklı and Küçükyavuz, 2019). The WVP has been shown to be computationally difficult:

Theorem 1 Buchholz and Scheftelowitsch (2019). *Solving the weighted value problem is NP-hard.*

Although we discuss the development of our solution methods in the context of the weighted value problem, our framework easily extends to other proposed multi-model objectives such as those discussed in Steimle et al. (2019) and Meraklı and Küçükyavuz (2019) which reflect DMs that are risk-averse to ambiguity in the MDP parameters. We show how a modification of our framework can be used to solve the percentile optimization objective for infinite-horizon MMDPs proposed in Meraklı and Küçükyavuz (2019) and discuss the results in a numerical study. The goal of the DM in percentile maximization is to maximize the value, z, such that $\mathbb{P}(v_\pi^m \geq z) \geq 1-\eta$. On the other hand, the goal of the DM in percentile minimization is to minimize the value, z, such that $\mathbb{P}(v_\pi^m \leq z) \geq 1-\eta$.

2.2. Related work

We now describe related work on mitigating parameter ambiguity in MDPs and focus on related methods where multiple models of MDPs are considered. As demonstrated in Mannor et al. (2007), parameter ambiguity can negatively impact decision-making in two ways. First, the DM may choose to operate under a policy that is not actually optimal because the optimization process was done with respect to parameter estimates that differ from the true parameters of the MDP. Second, the DM may get a false sense of confidence in that the value function found via the optimization process is higher than the true optimal value functions. Recently, there has been a stream of research dedicated to methods for addressing parameter ambiguity in MDPs. Robust optimization has been a common approach for addressing parameter ambiguity in MDPs. In the *robust MDP* setting, the DM seeks to select a policy that performs the best when the transition probability parameters are allowed to vary within an *ambiguity set*. Nilim and El Ghaoui (2005) and Iyengar (2005) showed that the max–min objective is tractable so long as the ambiguity set has a special structure called (s,a)-rectangularity, which means that the ambiguity set is constructed as the Cartesian product of ambiguity sets corresponding to rows for each $(s,a) \in \mathcal{S}\times\mathcal{A}$. The rectangularity assumption often leads to conservative solutions that perform poorly in expectation with respect to parameter uncertainty (Zhang et al., 2019). Work in this stream of research has sought to either find ambiguity sets that remain tractable while relaxing the (s,a)-rectangularity requirement (Mannor et al., 2016; Goyal and Grand-Clement, 2018; Zhang et al., 2019) or assume distributional information about the model parameters (Delage and Mannor, 2010; Xu and Mannor, 2012). We refer the interested reader to Mannor and Xu (2019) for more details on robust optimization approaches for MDPs with parameter ambiguity.

In contrast to the work above, the MMDP approach characterizes parameter uncertainty via multiple sets of model parameters with no rectangularity assumption. The MMDP approach under the weighted value objective was concurrently developed by Buchholz and Scheftelowitsch (2019) and Steimle et al. (2018) in the infinite-horizon and finite-horizon settings, respectively. Meraklı and Küçükyavuz (2019) noted that the weighted value objective may not be appropriate for DMs that are risk-averse to parameter ambiguity in MDPs and proposed an extension of the MMDPs using a percentile optimization objective. In general, the MMDP approach is NP-hard for the weighted value problems (Buchholz and Scheftelowitsch, 2019; Steimle et al., 2018) and the percentile optimization problem (Delage and Mannor, 2010).

Exact solution methods for solving these hard problems have relied on MIP formulations. Buchholz and Scheftelowitsch (2019) proposed a MIP formulation to find the optimal deterministic stationary policy to solve infinite-horizon MMDPs. Steimle et al. (2018) also proposed a MIP formulation in the finite-horizon setting, and Meraklı and Küçükyavuz (2019) proposed a MIP formulation for the percentile optimization objective. The formulation in (4a) is based on the primal linear programming formulation that can be used to solve a single MDP Puterman (1994, §9.4), with additional binary variables and logic-based constraints to enforce that each model of the MDP is operating under the same policy

$$\max_{\pi,v} \sum_{m\in\mathcal{M}}\sum_{s\in\mathcal{S}}\alpha_m\mu^m(s)v^m(s) \tag{4a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}}\pi(a|s) = 1, \qquad \forall s\in\mathcal{S}, \tag{4b}$$

$$M\pi(a|s) + v^m(s) - \lambda\sum_{s'\in\mathcal{S}}p^m(s'|s,a)v^m(s') \leq r^m(s,a) + M,$$

$$\forall\, m\in\mathcal{M}, s\in\mathcal{S}, a\in\mathcal{A}, \tag{4c}$$

$$\pi(a|s)\in\{0,1\}, \quad \forall a\in\mathcal{A}, \quad s\in\mathcal{S}, \tag{4d}$$

$$v^m(s)\in\mathbb{R}, \quad \forall m\in\mathcal{M}, \quad s\in\mathcal{S} \tag{4e}$$

where the binary variables are defined as follows:

$$\pi(a|s) = \begin{cases} 1 & \text{if the optimal policy takes action } a \text{ in state s} \\ 0 & \text{otherwise} \end{cases}, \quad \forall s\in\mathcal{S}, \quad a\in\mathcal{A}.$$

In (4a), the continuous variables, $v^m(s)$, represent the value to go in model m from state s under the policy described by the π variables. Constraints (4b) ensure that only one action is taken in each state. Constraints (4c) ensure that the value functions take on their maximum values so long as they correspond to the policy π and rely on the use of "big-M"s to enforce the logical relationship between the value function variables and the policy variables.

Although this formulation is valid, early work has shown that it does not scale well (Buchholz and Scheftelowitsch, 2019; Steimle et al., 2019). Most approaches for solving practical sized instances have depended on heuristic methods (Buchholz and Scheftelowitsch, 2019; Steimle et al., 2018; Meraklı and Küçükyavuz, 2019). Steimle et al. (2019) were recently successful in solving finite-horizon MMDPs using a B&B approach that relies on backward induction to generate bounds; however, their approach does not translate directly to infinite-horizon MDPs. In the next section we propose policy-based B&B algorithms for infinite-horizon MMDPs.

3. Methods

The PB-B&B framework we present takes advantage of the decomposable nature of the MMDP while leveraging specialized algorithms for solving MDPs to obtain bounds for each partial solution in the B&B tree. The main difficulty in solving MMDPs is that the optimal policy, π^*, must maximize the weighted performance which is known to be NP-hard. In contrast, given a policy, π, the objective value corresponding to that policy is easily found by evaluating π independently in each of the $|\mathscr{M}|$ MDPs.

We now present Algorithm 1 (PB-B&B) which leverages the decomposable structure of the MMDP. The algorithm starts by solving each model independently and subsequently adds restrictions that policies must agree across all models. These restrictions are are added incrementally based on *partial policies* which specify the actions to be taken in some, but not necessarily all, of the states. In other words, a subset of the states have a valid action $a \in \mathscr{A}$ fixed for all models, while all other states do not have any such restrictions. In order to obtain an upper bound on the objective value corresponding to the best possible completion of a partial policy, the algorithm solves a relaxed version of the MMDP. In this version, the states that do not have their actions specified by the partial policy are allowed to have different actions in each model. To solve this relaxation, each model in the MMDP can be solved independently using standard methods so long as states follow the actions prescribed by the partial policy if they have been specified.

Solving the relaxed MMDP at a given node provides an upper bound for the given partial policy. If the bound is worse than the best known completed policy (the *incumbent*), then the node can be pruned. *Warm-starting* the algorithm with a good incumbent policy may help accelerate the pruning of nodes by bound. A natural approach to warm-start is the solution to the mean value problem (MVP), which is a single MDP wherein each parameter takes on its weighted value across all the models of the MMDP. For instance, the MVP's transition probability parameters are specified as $\bar{p}(s'|s,a) = \sum_{m\in\mathscr{M}} \alpha_m p^m(s'|s,a)$ for all $s, s' \in \mathscr{S}$ and $a \in \mathscr{A}$. If the optimal completion of the partial policy at a node is the same in each model, the node can be pruned and the incumbent updated if appropriate. The upper bound on the weighted value is obtained in Line 6. If the DM would like to consider another objective function, such as maximizing the $(100 \times \eta)$ th-percentile as Meraklı and Küçükyavuz (2019) do, one would replace Line 6 with the upper bound on the $100 \times \eta$-percentile.

Pending nodes are examined by branching to further define the node's partial policy. To do so, we select a state s that is not already fixed under the partial policy and create $|\mathscr{A}|$ children nodes, one for each policy action that will be required by this new node's partial policy. In Algorithm 1, branching is accomplished in Steps 18 and the "for loop" starting in step 19.

Algorithm 1: Policy-based branch-and-bound (PB-B&B)

```
   Data: An MMDP
   Result: The optimal weighted value $W^*$, an optimal policy $\pi^*$
1  Lower bound: $W^{LB} \leftarrow -\infty$
2  Let $\hat{\pi}(s) \leftarrow \emptyset, \quad \forall s \in \mathcal{S}$
3  Let $Q$ be the set of pending nodes and $Q = \{\hat{\pi}\}$ the corresponding set of partial policies
     while $Q \neq \emptyset$ do
4        Remove a pending node from $Q$ and let $\hat{\pi}$ be the corresponding partial policy
5        Solve relaxation with $\hat{\pi}$ to obtain $(\pi^1, \ldots, \pi^m)$ and $(v^1, \ldots, v^m)$
6        $W \leftarrow \sum_{m\in\mathcal{M}} \alpha_m v^m_{\pi^m}$
7        if $W < W^{LB}$ then
8            Prune node $Q$ by bound
9        end
10       if $\pi^m = \pi^{m'}, \forall (m, m') \in \mathcal{M} \times \mathcal{M}$ then
11           if $W > W^{LB}$ then
12               $W^{LB} \leftarrow W$
13               $\pi^* \leftarrow \pi^1$
14           else
15               Prune node $Q$ by bound
16           end
17       else
18           Select a state $\bar{s}$ such that $\bar{\pi}(\bar{s})$ is empty
19           for each $a \in \mathcal{A}$ do
20               Let $\pi_a(s) = \pi(s) \quad \forall s \neq \bar{s}$
21               Let $\pi_a(\bar{s}) = a$
22               $Q \leftarrow Q \cup \pi_a$
23           end
24       end
25 end
26 $W^* \leftarrow W^{LB}$
```

In the discussion that follows, we present several key considerations for the design of the PB-B&B algorithm. We summarize these design considerations in Table 1.

3.1. Node selection strategies

There are multiple strategies for selecting pending nodes to efficiently explore the space of all possible policies. There are three main node selection strategies: 1) best-first search (BeFS) 2) depth-first search (DFS) and 3) breadth-first search (BrFS). In BeFS, the algorithm prioritizes the completion of policies with the highest upper bounds. In DFS, the algorithm wishes to obtain complete policies as soon as possible. Lastly, in BrFS, all actions for a particular state are explored before fixing actions in another state. We explore the best choice of node selection design computationally in Section 4.

3.2. Branching strategies

Another important design consideration in the PB-B&B algorithm is the branching strategy. In the context of the PB-B&B algorithm, the branching strategy will determine which state should be added to the subset of states for which actions are fixed according to the partial policy. We consider a branching strategy that follows from the high-level idea behind branching on the "most-fractional variable" in B&B for integer programming (Wolsey, 1998, p. 99) because we branch on the state for which the relaxation is furthest from a completed policy. The idea is that if many models disagree about the appropriate action to take for a particular state, branching on this state may reduce the number of future nodes explored. We consider two other types of branching strategies that consider discrepancies between the individual models. The first, value-based disagreement branching (VDB), measures disagreement using the standard deviation of individual models' value functions found solving the relaxation in Step 5. The second, policy-based disagreement branching (PDB), branches on the state where there is the largest number of unique actions specified by individual models' policies found solving the relaxation. As a reference branching strategy, we also consider arbitary branching (AB) in which states are branched on in a sequential order, starting with s_1 and ending with $s_{|S|}$.

3.3. Bounding strategies

In Step 5 of Algorithm 1, we obtain an upper bound on the best possible completion of a given partial policy by solving the node relaxation of the MMDP. We consider four methods for solving the relaxation: value iteration (VI), policy iteration (PI), modified policy iteration (MPI), and linear programming (LP).

Table 1
Algorithmic design choices for PB-B&B.

Algorithm Design Choice	Designs Considered
Node selection strategy	Best-first search (BeFS)
	Breadth-first search (BrFS)
	Depth-first search (DFS)
Branching strategy	Arbitary branching (AB)
	Value-based disagreement branching (VDB)
	Policy-based disagreement branching (PDB)
	Monotone branching (MB)[a]
Bounding strategy	Linear programming (LP), Exact
	Policy iteration (PI), Exact
	Value iteration (VI), Approximation
	Modified policy iteration (MPI), Approximation

[a] Monotone branching is considered in special cases.

The choice of using an exact or approximate method for solving the relaxation at each node presents an interesting potential trade-off in computational performance. On the one hand, we could use exact solution methods like PI or LP to obtain tighter upper bounds than the approximate methods and could potentially prune nodes by bound earlier in the search process; however, the tighter upper bounds may come at the cost of additional computational effort to solve each node relaxation. On the other hand, approximation algorithms like VI and MPI may generate looser upper bounds, leading the algorithm to potentially explore more nodes, but the bounds at each node could be generated more quickly.

For the approximation algorithms (VI and MPI), there are well-known stopping criteria that will guarantee ϵ-optimal solutions (Puterman, 1994, §6.3, §6.5). We hypothesized that increasing ϵ would result in a trade-off in the time required to solve node relaxations versus the total number of nodes explored. We explore this trade-off computationally in Section 4.

3.4. Enhancements for monotone policies

In this section, we show how PB-B&B can be enhanced when only searching over the space of monotone policies. For standard MDPs, there exist sufficient conditions that guarantee the existence of an optimal monotone policy (Puterman, 1994, §6.11.2). Monotone policies are often desirable to DMs due to their interpretability.

First, we show that if each of the MMDP's models satisfies some sufficient conditions for a monotone policy to be optimal, then the MVP is guaranteed to have a monotone policy that is optimal.

Proposition 1. If each model, $m \in \mathscr{M}$, satisfies the following conditions:

1. $r^m(s,a)$ is non-decreasing in s for all $a \in \mathscr{A}$,
2. $q^m(k|s,a) \equiv \sum_{j=k}^{|\mathscr{S}|} p^m(j|s,a)$ is non-decreasing in s for all $k \in \mathscr{S}$ and $a \in \mathscr{A}$,
3. $r^m(s,a)$ is a superadditive function on $\mathscr{S} \times \mathscr{A}$,
4. $q^m(k|s,a)$ is a superadditive function on $\mathscr{S} \times \mathscr{A}$ for all $k \in \mathscr{S}$,

then there is guaranteed to be a non-increasing policy that is optimal for the MVP of the MMDP.

The result of Proposition 1 is useful because it implies that we can use the solution to the MVP as a warm-start for the PB-B&B algorithm when searching over only the class of monotone policies for an MMDP where this restriction is natural.

Empirically, we have observed that if each model in the MMDP has a monotone policy that is optimal, the optimal policy for the WVP is also monotone; Meraklı and Küçükyavuz (2019) have also observed this phenomenon. However, sufficient conditions for the MMDP optimal policy to be monotone have not been identified. Nevertheless, monotone policies have desirable properties such as their interpretability and ease of implementation. Further, the DM may find it appealing that the solution to the WVP is no more complex than the solution to the individual MDPs or the MVP.

We consider two enhancements to the B&B strategy that exploit monotone policies. In the first enhancement, we modify the branching strategy to omit examining certain actions that would violate the monotone nature of the optimal policy. Specifically, in Step 19 of Algorithm 1, only partial policies that could be completed with a monotone policy are added to the set of pending partial policies. In the second enhancement, we modify the method for solving node relaxations. For instance, PI in Step 5 could be replaced with with monotone PI (Puterman, 1994, §6.11.2) which achieves computational efficiencies by eliminating non-monotone policies from consideration.

4. Numerical study of PB-B&B designs

In this section, we compare the different approaches for solving the MMDP. We generated random test instances of MDPs to compare the PB-B&B under the different designs specified in Section 3 and summarized in Table 1. After determining the best designs for the PB-B&B algorithm, we will compare the B&B approach to the MIP-based approach in Section 5.

4.1. Design of test instances

We compared our algorithmic designs on a set of randomly generated MMDP instances which were generated using the procedure described in Buchholz and Scheftelowitsch (2019). The rewards, model weights, initial distributions, and transition probability matrices in these instances are randomly sampled from uniform distributions, and all vectors describing probability distributions are scaled so that their sum is equal to one. We compared our different designs on 30 MMDP instances with 8 states, 8 actions, and 8 models each. The discount factor for each instance was set to 0.97.

All experiments were run on a Windows Server 2012 R2 Standard with a 3.4 GHz Intel processor and 128 GB of RAM. The PB-B&B was implemented in C++. The MIP formulation and the linear programming relaxation solution were implemented using Gurobi 7.5.1. The number of threads for Gurobi was set to be 1. We recorded the computation time in CPU seconds and the optimality gap. We attempted to solve each test instance to within 1% of optimality subject to a time limit of 300 s.

4.2. Algorithm design results

Table 2 compares the BeFS, DFS, and BrFS node selection strategies in terms of the computation time, number of nodes visited, and the optimality gap. We found that these strategies performed similarly in terms of median computation time. However, BeFS noticeably outperformed the other two node selection strategies in mean computation time. Moreover, BeFS was able to solve all but one of the thirty test instances, whereas BrFS was unable to solve four instances, and DFS could not solve six instances within the time limit. The single instance where best-first search (BeFS) did not terminate within 300 seconds had an optimality gap of 1.4%, while the worst case optimality gaps for BrFS and DFS were 5.6% and 26.4%, respectively. Although all three node selection strategies may perform similarly on average, the poor worst case performances of BrFS and DFS suggest that BeFS may be the best choice of node selection strategy.

Table 3 compares the VBD, PBD, and AB branching strategies in terms of the number of instances solved, computation time, and number of nodes visited. We found that PBD works the best among the three branching strategies in terms of number of instances solved, mean and worst-case gaps, and mean number of nodes explored. We found that VBD performed worse than AB, leading us to believe that the value functions are not a meaningful measure of policy disagreement among the models. VBD had a significantly higher average and median computation time and worst-case optimality gap relative to the other two branching strategies. These findings suggest that branching on states in which there is disagreement in the optimal policy can lead to computational gains relative to simply branching in an arbitrary fashion.

We now present the trade-off between obtaining tight upper bounds and solving relaxations quickly when using approximation methods to generate bounds. Fig. 1 demonstrates the influence of increasing the stopping criteria, ϵ, on total computation time for two different values of the discount factor, λ. For each value of discount factor, we observe that the mean computation time initially decreases in ϵ and then begins to increase once ϵ grows sufficiently large. Our findings suggest that solving each node very close to optimality led to overall increases in computation time due to the additional time spent processing each node. On the other hand, while large values of ϵ allowed the relaxation at a given node to terminate quickly, these values led to poor upper bounds, leading the algorithm to explore more nodes in the tree. We found that selecting an ϵ in the middle of these two extremes can decrease overall computation time and that a good choice of ϵ will be problem-dependent. Empirically, we observed that using a stopping criteria of $\epsilon = \frac{2\times 0.001\times W\times\lambda}{1-\lambda}$ tended to perform well across a wide number of instances, where W is the value of the wait-and-see objective of the instance. For our final PB-B&B design, we use this approach to pre-compute a value of the stopping criteria ϵ before beginning the branching.

Finally, Table 4 shows the computation time, number of nodes visited, and the optimality gap for the VI, MPI, LP, and PI bounding

Table 2
Computational comparison of the three node selection strategies: Best-first search (BeFS), Breadth-first search (BrFS), and Depth-first search (DFS). Policy-based disagreement branching (PDB) was used for the branching strategy and value iteration (VI) was used as the method for solving the relaxation. 30 instances were solved with a discount factor of 0.97.

Selection Strategy	Instances Solved	Solution Time (CPU secs.)			Nodes Visited (1000s)		Optimality Gap (%)	
		Avg.	Med.	Max.	Avg.	Max.	Avg.	Max.
BeFS	96.7%	97.4	67.1	>300.0	134.9	349.0	<1.0	1.4
BrFS	86.7%	101.5	68.7	>300.0	164.5	369.4	<1.0	5.6
DFS	80.0%	106.4	71.7	>300.0	138.3	350.3	3.2	26.4

Table 3
Computational comparison of three branching strategies: Value-based disagreement branching (VDB), Policy-based disagreement branching (PDB), and Arbitary branching (AB) on the test instances. Best-first search (BeFS) was the node selection strategy and value iteration (VI) was the method for solving the relaxation. 30 instances were solved with a discount factor of 0.97.

Branching Strategy	Instances Solved	Solution Time (CPU secs.)			Nodes Visited (1000 s)		Optimality Gap (%)	
		Avg.	Med.	Max.	Avg.	Max.	Avg.	Max.
AB	90.0%	112.3	71.1	>300.0	140.0	393.5	<1.0	1.4
PBD	96.7%	97.6	67.8	>300.0	134.8	349.1	<1.0	1.4
VBD	83.3%	123.9	84.8	>300.0	156.2	308.5	1.1	2.2

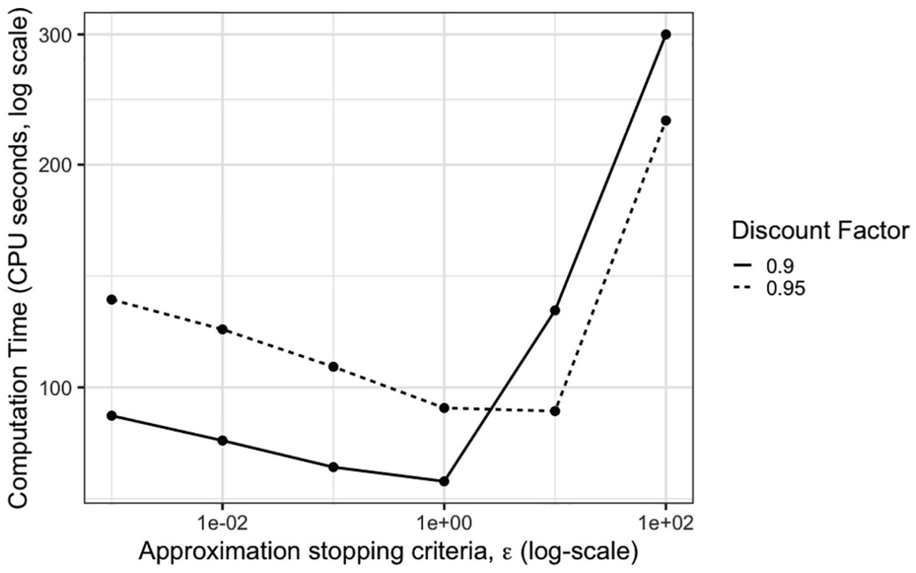


Fig. 1. Comparison of mean runtimes for 6 different values of the approximation algorithm's stopping criteria, ϵ, and two different discount factors.

Table 4
Computational comparison of four methods for solving the relaxation: Value iteration (VI), Modified policy iteration (MPI), Policy iteration (PI), and Linear programming (LP) on the test instances. Best-first search (BeFS) was the node selection strategy and Policy-based disagreement branching (PDB) was the branching strategy. 30 instances were solved with a discount factor 0f 0.97. MPI used an $m_n = 5$; for more details, consult (Puterman, 1994).

Relaxation Strategy	Instances Solved	Solution Time (CPU secs.)			Nodes Visited (1000 s)		Optimality Gap (%)	
		Avg.	Med.	Max.	Avg.	Max.	Avg.	Max.
VI	96.7%	102.9	69.3	> 300.0	134.6	349.0	<1.0	1.55
MPI	96.7%	96.6	61.7	> 300.0	142.1	367.9	<1.0	1.55
PI	0%	> 300.0	> 300.0	> 300.0	0.7	0.8	15.0	18.8
LP	0%	> 300.0	> 300.0	> 300.0	12.7	13.1	7.8	11.8

strategies. We found that the approximation algorithms, VI and MPI, significantly outperformed the exact algorithms, LP and PI, in terms of computation time. The approximation algorithms were able to solve 96.7% of the instances, whereas the exact methods could not solve a single instance within the time limit. Since PI could only examine around 720 nodes on average, as opposed to VI and MPI, which examined 100,000 nodes on average, the time to solve the relaxation at a given node using PI was significantly longer than the approximation algorithms. At the same time, solving the relaxation exactly clearly did not yield bounds that facilitated increased node pruning, demonstrating the ineffectiveness of PI for solving large MMDPs.

The approximation algorithms, VI and MPI outperform the exact methods, suggesting that generating an ϵ-optimal solution to the relaxation quickly will yield sufficiently tight bounds that lead to faster solution time. Between the approximation algorithms, MPI outperformed VI in terms of mean and median computation time, despite examining more nodes on average. Therefore, our results suggest that MPI is the most effective method for solving the relaxation in PB-B&B.

5. Case studies

In this section, we present numerical experiments comparing the best implementation of the B&B algorithm with the MIP formulation presented in (4). We used the following three test cases:

- **Random instances (RI) from** Buchholz and Scheftelowitsch (2019)**:** These MMDP instances are generated using the same procedure from Buchholz and Scheftelowitsch (2019) which was used in Section 4. In this section, we vary the number of actions, states, and models, and we generate 30 instances of each problem size. We generated test instances by sampling the transition probability and reward parameters from the distributions described in the article and used a discount factor of 0.97.
- **Machine maintenance (MM) from** Delage and Mannor (2010)**:** This MMDP was generated from the description of a machine maintenance MDP described in Delage and Mannor (2010). The state space is comprised of eight states representing different levels of quality the machine's operation while two states correspond to different types of maintenance. There are four actions corresponding to different repair options that influence which type of maintence the machine will undergo. Parameter ambiguity in the transition probabilities among the different states under the repairs options is modeled using a Dirichlet distribution. The nominal transition probability parameters were scaled by a *concentration parameter*, denoted *c*, to generate the parameters for the Dirichlet distribution. We considered three values of concentration parameters, *c* = 1, 5, and 9. We also considered MMDPs with 10, 30, and 100 models. We still use a discount factor of 0.97.
- **HIV therapy (HIV) from** Shechter et al. (2008)**:** This MMDP was generated from the description of an MDP used to determine the optimal time to initiate therapy for HIV from Shechter et al. (2008). There are six states in the model: four transient states corresponding to the patient's CD4 count (an indicator of the disease's severity) and two absorbing states correspond-

ing to death and post-treatment. The actions are whether or not to start therapy in order to maximize a quality-adjusted measure of life expectancy. The authors present multiple plausible models of this decision making process due to the different methods for estimating utilities, natural history of the disease, and prognosis after treatment. The different methods for estimating the parameters lead to 72 different models that define the infinite-horizon MMDP. We found that each model of the MDP, when solved independently, satisfied the requirements for having an optimal non-decreasing monotone policy (Puterman, 1994, §6.11.2). Therefore, we also used a version of this MDP, which we refer to as HIV-M, to test the enhancements for monotone policies that we discussed in Section 3.4.

For RI and MM, we generated 30 instances for each set of problem characteristics by sampling from the distributions described in these articles. For all problem instances, we instituted an optimality gap of 1% and a time limit of 300 s. For the MM case study, we also considered a version of the MMDP with a percentile optimization objective (Meraklı and Küçükyavuz, 2019). The goal of the DM in the MM case study is percentile minimization, or to minimize the value, z, such that $\mathbb{P}(v_\pi^m \leq z) \geq 1-\eta$. We denote this objective using PercOpt($100\times\eta$%), which signifies optimizing the $100\times\eta$th percentile. We compared the PB-B&B approach to a MIP formulation and analyzed the policies resulting from the weighted value and PercOpt objective functions. We provide details about the MIP implementation of this problem in Appendix B.2. We have created a repository of the data for these test instances which is available at: `https://doi.org/10.7302/2frp-2m36`.

All experiments were run on a Windows Server 2012 R2 Standard with a 3.4 GHz Intel processor and 128 GB of RAM. The PB-B&B was implemented in C++. The MIP formulation was implemented using Gurobi 7.5.1. The number of threads for Gurobi was set to be 1. The big-Ms. used in the MIP were tightened according to the procedure described in Appendix B.1. We recorded the computation time in CPU seconds and the optimality gap for both the PB-B&B and the MIP formulation.

The results of our experiments comparing PB-B&B to the MIP formulation for the weighted value objective are summarized in Table 5. For each case study and for each set of problem characteristics, the PB-B&B performed at least as well as the MIP in terms of the percentage of instances solved. In addition, the PB-B&B significantly outperformed the MIP in terms of average computation time and the average optimality gap for large problem instances.

The PB-B&B and MIP both solved the small problems very quickly. The first four sets of problem instances shown in Table 5 were solved in under one second on average. For slightly larger problem instances, such as the 5-model, 5-state, 5-action MMDPs, the PB-B&B begins to outperform the MIP. For 2-model, 10-state, 3-action problems, the PB-B&B shows greater improvements over the MIP formulation, with a mean computation time of 0.04 s versus 11.66 s, respectively. The PB-B&B continues to significantly outperform the MIP as the problem size increases. When the MMDP has 2 models, 10 states, and 10 actions, the MIP formulation is unable to solve a single test instance within the time limit. Meanwhile, the PB-B&B solved every such problem with a mean of 9.18 s. Moreover, once this problem size was reached, the MIP had an optimality gap at termination of over 1600% on average. For a much larger problem size (3 models, 20 states, 4 actions), the PB-B&B was still able to solve a majority of the test instances while the MIP still could not solve a single one of these instances. Among the four test instances of this size that the PB-B&B could not solve, the maximum optimality gap after 300 s was less than 3%. Lastly, for the largest problem instances, neither the PB-B&B nor the MIP were able to solve the test instances within the time limit. However, the highest mean optimality gap for the PB-B&B was 16%, versus 2200% for the MIP.

In the MM problem instances, the PB-B&B significantly outperformed the MIP formulation for each of the 270 test instances. The PB-B&B solved all instances of the MM decision-making problem within the time limit, whereas the MIP formulation only solved 37% of the instances. For very large problem instances, such as with 100 models, the MIP formulation could not solve any of the

Table 5
Computational comparison of the PB-B&B and MIP for the RI, MM, and HIV MDPs. For each problem size, the instance was solved to within 1% of optimality. Each solution method was warmstarted with the mean-value policy and had a time limit of 300 s. The discount factor was 0.97 for all cases. The PB-B&B design included BeFS as the node selection strategy, PBD as the branching strategy, and MPI as the method for solving the relaxation. The PB-B&B used the monotone branching strategy for the HIV-M case.

MDP	Characteristics				Instances Solved		Solution Time Avg., (CPU Secs.)		Optimality Gap Avg., (%)	
	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	c	PB-B&B	MIP	PB-B&B	MIP	PB-B&B	MIP
RI	2	2	2	NA	100%	100%	0.03	0.02	<1	<1
	3	3	3	NA	100%	100%	0.09	0.04	<1	<1
	2	5	2	NA	100%	100%	0.02	0.03	<1	<1
	10	5	2	NA	100%	100%	0.08	0.08	<1	<1
	5	5	5	NA	100%	100%	0.72	1.07	<1	<1
	2	10	3	NA	100%	100%	0.04	11.66	<1	<1
	2	10	10	NA	100%	0%	9.18	>300.0	<1	1666.9
	3	20	4	NA	86.7%	0%	98.84	>300.0	1.2	1385.6
	3	50	3	NA	0%	0%	>300.0	>300.0	7.0	1327.7
	5	50	5	NA	0%	0%	>300.0	>300.0	16.0	2586.8
	3	100	4	NA	0%	0%	>300.0	>300.0	12.2	2315.2
	3	200	4	NA	0%	0%	>300.0	>300.0	11.9	2200.0
	3	300	4	NA	0%	0%	>300.0	>300.0	12.4	2244.9
MM	10	10	4	1	100%	83.3%	1.37	177.14	<1	77.0
	10	10	4	5	100%	10.0%	1.72	287.90	<1	470.8
	10	10	4	9	100%	6.7%	1.54	292.81	<1	495.6
	30	10	4	1	100%	23.3%	8.62	298.13	<1	185.1
	30	10	4	5	100%	0%	4.57	> 300.0	<1	931.1
	30	10	4	9	100%	0%	4.46	> 300.0	<1	1019.8
	100	10	4	1	100%	0%	50.02	>300.0	<1	952.4
	100	10	4	5	100%	0%	11.54	>300.0	<1	1855.0
	100	10	4	9	100%	0%	5.84	>300.0	<1	1970.1
HIV	72	6	2	NA	NA	NA	0.14	0.10	<1	<1
HIV-M	72	6	2	NA	NA	NA	0.05	0.10	<1	<1

instances within the 300-s time limit. Similar results were noted for the large RI instances.

When solving the MM problem instances, the solution time for the PB-B&B increased as the number of models increased and as the concentration parameter decreased. The expected time required for solving the relaxation grows linearly in the number of models, which explains why increasing the number of models also increases overall computation time. We suspect that the computation time increases as the concentration parameter decreases because there is more variation in each of the model's parameters. When the models are increasingly different from one another, each model's individual policy is different, leading to increased disagreement among policies and the exploration of more nodes. Interestingly, the performance of the MIP was better when the concentration parameter decreased. However, as the concentration parameter increased and the models became more similar, the MIP required more computation time and overall had much higher optimality gaps at termination.

For the HIV case study, we observed that a non-decreasing monotone policy was optimal for the WVP based on Theorem 6.11.6 of Puterman (1994). For these instances, we tested a separate branching strategy, Monotone branching (MB), in which we limited our search space to monotone partial policies. This resulted in a significant relative reduction in the number of nodes, 15 to 5 nodes, and computation time, from 0.14 to 0.05 s, respectively. Although this is a small problem instance, this finding suggests that monotone branching may be useful for reducing the number of nodes explored within the PB-B&B algorithm, generating significant computational savings on very large problems.

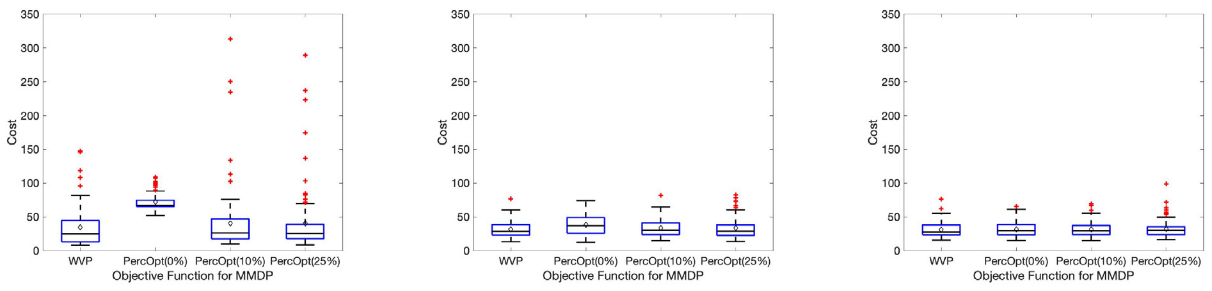
Table 6 shows a comparison of the PB-B&B and MIP solution methods in terms of instances solved, solution time, and optimality gap for the MM problem instances with a percentile optimization objective for η = 0.0 (PercOpt(0%)), 0.10 (PercOpt(10%)), 0.25 (PercOpt(25%)). In this case, η = 0.0 is equivalent to minimizing the worst-case cost.

As observed with the WVP, the PB-B&B approach outperforms the MIP formulation, with the PB-B&B solving 807 of the 810 test instances within the time limit, whereas the MIP was only able to solve 312 out of 810. The PB-B&B also outperformed the MIP in terms of average solution time for all problem sizes and percentile values; it also had a higher percentage of instances solved and a better average optimality gap in all but one problem size (PercOpt(0%), 30 models, 10 states, 4 actions, $c = 1$). In general, the solution time required to solve the percentile optimization objective decreased with the percentile for the PB-B&B but increased with the percentile for the MIP. As previously seen with the WVP, the solution time increased for the MIP as the concentration parameter increased and variance subsequently decreased. We observed the opposite trend for the PB-B&B, but only when there were 100 models, since the average solution time considerably decreased as the variance increased. We conjecture that the behavior of the MIP for PercOpt(10%) and PercOpt(25%) may be attributed to the fact that when model variances are large, there is a clear stratification of the value functions, allowing the algorithm to more easily find the desired percentile value. Yet when the value functions are closer to each other, it is more difficult for the MIP to pinpoint the model value for the desired percentile. Overall, these findings demonstrate that the PB-B&B approach outperforms MIP-based solution methods when extended to another multi-model objective function that is relevant for risk-averse DMs.

Fig. 2 demonstrates how the WVP and percentile optimization objectives compare in terms of mitigating parameter ambiguity in MDPs. The boxplots show the distribution of the models' value functions for a 100-model instance of the MM MMDP under different optimal policies. Boxplots for additional instances are shown in Appendix C. The WVP policy is obtained via PB-B&B for the WVP objective function. The other policies are found using PB-B&B for the percentile optimization objective that optimized the 0th percentile (worst-case), 10th percentile, and 25th percentile. As

Table 6
Computational comparison of the PB-B&B and MIP for the MM MMDPs using the percentile optimization objective for the corresponding percentile, η. For each problem size, the instance was solved to within 1% of optimality. Each solution method was warmstarted with the mean-value policy and both had a time limit of 300 s. The discount factor was 0.97 for all cases. The PB-B&B design included BeFS as the node selection strategy, PBD as the branching strategy, and MPI as the method for solving the relaxation.

η	Characteristics				Instances Solved		Solution Time Avg., (CPU secs.)		Optimality Gap Avg., (%)	
	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	c	PB-B&B	MIP	PB-B&B	MIP	PB-B&B	MIP
0.0	10	10	4	1	100.0%	100.0%	1.48	50.33	<1	<1
	10	10	4	5	100.0%	100.0%	2.22	62.58	<1	<1
	10	10	4	9	100.0%	100.0%	2.78	71.55	<1	<1
	30	10	4	1	96.7%	100.0%	23.62	97.36	1.1	<1
	30	10	4	5	100.0%	100.0%	9.85	159.43	<1	<1
	30	10	4	9	100.0%	100.0%	10.12	170.99	<1	<1
	100	10	4	1	96.7%	23.3%	88.85	288.73	1.1	51.2
	100	10	4	5	96.7%	0.0%	59.48	>300.0	1.0	191.7
	100	10	4	9	100.0%	0.0%	32.83	>300.0	<1	237.0
0.1	10	10	4	1	100.0%	100.0%	0.78	154.82	<1	<1
	10	10	4	5	100.0%	93.3%	1.47	192.23	<1	<11.8
	10	10	4	9	100.0%	96.7%	1.75	183.01	<1	2.9
	30	10	4	1	100.0%	0.0%	4.40	>300.0	<1	417.0
	30	10	4	5	100.0%	0.0%	5.44	>300.0	<1	570.6
	30	10	4	9	100.0%	0.0%	5.53	>300.0	<1	580.8
	100	10	4	1	100.0%	0.0%	33.11	>300.0	<1	714.5
	100	10	4	5	100.0%	0.0%	20.40	>300.0	<1	1101.1
	100	10	4	9	100.0%	0.0%	10.25	>300.0	<1	1285.8
0.25	10	10	4	1	100.0%	36.7%	0.68	>284.93	<1	223.6
	10	10	4	5	100.0%	26.7%	1.04	290.58	<1	199.1
	10	10	4	9	100.0%	63.3%	1.03	278.62	<1	74.3
	30	10	4	1	100.0%	0.0%	3.58	>300.0	<1	852.5
	30	10	4	5	100.0%	0.0%	3.25	>300.0	<1	866.3
	30	10	4	9	100.0%	0.0%	3.28	>300.0	<1	853.5
	100	10	4	1	100.0%	0.0%	28.21	>300.0	<1	1223.8
	100	10	4	5	100.0%	0.0%	9.65	>300.0	<1	1827.5
	100	10	4	9	100.0%	0.0%	5.82	>300.0	<1	1918.7



(a) $c = 1$ (Highest variance) (b) $c = 5$ (c) $c = 9$ (Lowest Variance)

Fig. 2. Boxplots showing the distribution of an MMDP's models' value functions for a 100-model instance of the MM MMDP under different policies. The WVP policy is obtained via PB-B&B for the WVP objective function. The other policies are found using PB-B&B for the percentile optimization formulation that optimized the 0th percentile (worst-case), 10th percentile, and 25th percentile. In the MM MMDP, we aim to minimize cost.

expected, we observe that the WVP policy performs the best in terms of the weighted value, the PercOpt(0%) performs the best in terms of worst-case model value, and the other percentile optimization objectives achieve their respective percentiles. For a concentration parameter of 1, Fig. 2(a) illustrates that while the WVP optimal policy did not have the lowest worst-case cost, it did have the lowest weighted-value cost as well as the second lowest worst-case cost, which was better than the policies of both PercOpt(10%) and PercOpt(25%). Moreover, both PercOpt(10%) and PercOpt(25%) had multiple individual models whose value functions were outliers, many of which exceeded 200.

The fact that the WVP policy outperforms a policy optimized for the 10th or 25th percentiles can likely be ascribed to the binary nature of the objective in terms of whether a model has a cost below the optimized percentile value. That is, the percentile optimization tends to sacrifice the performance of models that tend to have higher cost in order to achieve a policy that improve the desired percentile. We also observe that as the concentration parameter increases, and in turn as the variance among the models' parameters decreases, the value functions for the various models tend to be more similar under the various models. This suggests that the objective function that is specified to mitigate parameter ambiguity is most important when the variance among the models' parameters is high. As a result, when there is high variance in the model parameters, the DM is advised to solve the WVP as opposed to PercOpt because this objective function effectively balances the trade-offs of the worst case performance with its weighted value performance.

In summary, our experiments show that the PB-B&B significantly outperforms the MIP in terms of computation time and optimality gaps for several large, existing decision-making problems across different domains. As a result, we recommend that the PB-B&B be used to solve large infinite-horizon MMDPs due to the algorithm's ability to exploit the specific problem structure and the MIP's inability to quickly generate favorable bounds. We also demonstrated the flexibility of the PB-B&B algorithm by solving a percentile optimization version of the MMDP. The PB-B&B outperformed a MIP formulation of the percentile optimization version and demonstrated that the WVP objective can perform better than the percentile optimization objective in terms of weighted and worst-case performance.

6. Conclusion

We considered a new solution method for solving infinite-horizon MDPs when the DM has multiple plausible models of the parameters. This problem has been shown to be NP-hard and existing solution methods based on the state-of-the-art MIP have struggled to scale to larger instances of these problems.

In this article, we proposed a custom PB-B&B as a solution method which leverages the ability to optimize and evaluate policies quickly for single-model MDPs. By branching on the policy-space, we are able to use specialized solution methods for single-model MDPs to quickly obtain bounds. We considered several important design considerations for the PB-B&B algorithm, including the node selection strategy, branching strategy, bounding strategy, and choice of stopping criteria, and explored their performance numerically using a test set of previously-published random instances described in Buchholz and Scheftelowitsch (2019). We found that BeFS node selection and a PBD branching strategy tended to perform the best in our computational experiments. We also found that using MPI, an approximation method, can obtain bounds when solving the relaxation much faster than exact methods such as LP or PI. Lastly, we found a good rule-of-thumb for pre-computing the stopping criteria, ϵ, which can enhance the performance of the approximation algorithms in generating bounds. Overall, our PB-B&B can successfully solve large problems by exploiting the specific problem structure of the MMDP as well as by solving subproblems approximately, instead of exactly.

We compared the performance of PB-B&B against MIP-based methods on previously-published MDP formulations where there are multiple models of the parameters: a set of random MMDP instances reported in Buchholz and Scheftelowitsch (2019), a set of machine maintenance MDPs with parameter ambiguity presented in Delage and Mannor (2010), and a medical decision-making instance to determine optimal treatment of HIV when there are multiple plausible models of the MDP (Shechter et al., 2008). PB-B&B significantly outperformed the MIP formulation on each of these case studies when the MDPs had many states, actions, and/or models. We also demonstrated that, if the best monotone policy is acceptable for the DM, PB-B&B can be modified for further computational gains. For very large problem instances, the PB-B&B could not solve a problem within the 300 s time limit, but its optimality gap was reasonably small, whereas the MIP sometimes had gaps greater than 2000% for these same instances. We presented the PB-B&B algorithm in the context of the WVP. One may view the WVP as an appropriate objective for a DM who is risk-neutral towards ambiguity in the model's parameters. However, others have proposed alternative objectives that may be more appropriate for a DM that is risk-averse to ambiguity in the model's parameters, such as the percentile optimization objective proposed in Meraklı and Küçükyavuz (2019). We demonstrated that the PB-B&B is easily adapted to consider other objectives, such as percentile optimization, by simply changing the calculation for the upper bound at each node in the branch-and-bound tree. Using numerical experiments, we demonstrated that the PB-B&B algorithm outperformed a MIP-based approach for percentile optimization. Further, our experiments revealed that a WVP approach can yield policies that outperform policies generated from certain percentile optimization objectives in terms of both weighted and worst-case performance. This finding suggests that the WVP may lead to more robust solutions than a percentile optimization objective, depending on the percentile optimized. We also found that the choice of objective function for the MMDP is most important when the variance in transition probabilities among the models' is high. In these situations, the WVP objective function tended to offer both good weighted value and worst-case performance relative to the percentile optimization objectives. Meanwhile, when variance in the model parameters is low, the policies generated by the different objective functions tend to perform similarly.

Our study is not without limitations. In our computational study, we found the PB-B&B approach was unable to solve some of the MMDPs with 50 or more states within a time limit of 300 s. However, the PB-B&B still fared significantly better on these large instances than the MIP approach. In addition, the computational gains associated with monotone policy structure were relevant to only one of our test instances and it was not a large problem. Nevertheless, our results do demonstrate the feasibility of exploiting this problem structure and showed significant relative improvement in computation times. We compared our PB-B&B algorithm to a MIP-based approach to solve the percentile optimization version of the MMDP. Although we did our best to replicate the proposed formulation from Meraklı and Küçükyavuz (2019), there may be small differences between our implementation of the proposed formulation and the implementation that is presented in Meraklı and Küçükyavuz (2019), which would lead to differences in performance.

There are opportunities to build upon the ideas presented in this paper. First, there may be other enhancements to the PB-B&B algorithm. We considered the most well-known node selection strategies but other more specialized node selection strategies may perform better. Similarly, there may be methods that incorporate aspects of different algorithmic design choices, such as combining PBD and VBD. Second, we presented our algorithms in the context of the WVPs and PercOpt problems, but the best search strategies for other objective functions (e.g. regret) could vary. Finally, we showed how our PB-B&B algorithm could be modified to exploit monotonicity of policies. Future work may explore how other side constraints that add structure to the policy can be incorporated into such a framework.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Vinayak S. Ahluwalia: Methodology, Software, Validation, Data curation, Investigation, Formal analysis, Writing - original draft, Visualization. **Lauren N. Steimle:** Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Visualization, Funding acquisition. **Brian T. Denton:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Acknowledgment

This work was supported by the National Science Foundation under Grant Nos. DGE-1256260 (Steimle) and CMMI-1462060 (Denton); any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Appendix A. Proof of Proposition 1

Proof. Proof of Proposition 1 Theorem 6.11.6 of Puterman (1994) states that if an MDP satisfies Conditions 1–4 in the statement of the proposition, then there exists a monotone nonincreasing policy that is optimal for the MDP. We prove the proposition by showing that if each model in an MMDP satisfies Conditions 1–4, then there exists an optimal policy for the MVP that is monotone nonincreasing.First, we will show that the rewards and tailsums are nondecreasing if Condition 1 is satisfied for the MMDP. By Condition 1 of the assumption, each model's rewards are nondecreasing with respect to states for each action and model. Therefore, because the weights are non-negative, it follows that the MVP's rewards are also nondecreasing:

$$\begin{aligned} &r^m(s,a) \leq r^m(s+1,a), \quad \forall a \in \mathscr{A}, \quad m \in \mathscr{M} \\ \Rightarrow &\sum_{m \in \mathscr{M}} \alpha_m r^m(s,a) \leq \sum_{m \in \mathscr{M}} \alpha_m r^m(s+1,a), \quad \forall a \in \mathscr{A}. \end{aligned}$$

Thus, the MVP reward function satisfies Condition 1 of Puterman (1994) Theorem 6.11.6. A similar argument shows that the MVP's tailsums $\sum_{m \in \mathscr{M}} \alpha_m q^m(k|s,a)$ are nondecreasing in s for all $k \in \mathscr{S}$ and $a \in \mathscr{A}$ and thus Condition 2 is also satisfied. Next, we show that the rewards and tailsums of the MVP are superadditive. We show this for the rewards case. By Condition 3 and the definition of superadditivity, each model has superadditive rewards:

$$r^m(s^-,a^-) + r^m(s^+,a^+) \geq r^m(s^+,a^-) + r^m(s^-,a^+), \quad \forall m \in \mathscr{M},$$

where $s^- \leq s^+ \in \mathscr{S}$, and $a^- \leq a^+, \forall a \in \mathscr{A}$. Therefore, for $s^- \leq s^+$ and $a^- \leq a^+$, it follows that

$$\begin{aligned} &\alpha_m r^m(s^-,a^-) + \alpha_m r^m(s^+,a^+) \geq \alpha_m r^m(s^+,a^-) + \alpha_m r^m(s^-,a^+), \quad \forall m \in \mathscr{M} \\ \Rightarrow &\alpha_m (r^m(s^-,a^-) + r^m(s^+,a^+)) \geq \alpha_m (r^m(s^+,a^-) + r^m(s^-,a^+)), \quad \forall m \in \mathscr{M} \\ \Rightarrow &\sum_{m \in \mathscr{M}} \alpha_m (r^m(s^-,a^-) + r^m(s^+,a^+)) \geq \sum_{m \in \mathscr{M}} \alpha_m (r^m(s^+,a^-) + r^m(s^-,a^+)). \end{aligned}$$

This implies that the MVP also has superadditive rewards which satisfies Condition 3 of Theorem 6.11.6 in Puterman (1994). A similar argument shows the tailsums are superadditive satisfying Condition 4. Thus, Conditions 1–4 are satisfied. By Theorem 6.11.6 of Puterman (1994), it follows that if Conditions 1–4 are satisfied for all models that define an MMDP, then the corresponding MVP is guaranteed to have a monotone policy that is optimal. □

Appendix B. Implementation details

B.1. Tightening procedure for big-Ms

In the MIP formulation that was implemented in Section 5, we select the big-Ms. in the constraint (4c) corresponding to model m, state s, and action a to be sufficiently large so that these constraints become redundant if $\pi(a|s) = 1$. However, we desire that these big-M values are as small as possible so long as they stay sufficiently large. Tightening the big-Ms. improves the linear programming relaxation of the problem and has been shown to improve the running time of MIP solvers.

To tighten the big-Ms, we used a similar procedure to that described in Meraklı and Küçükyavuz (2019) and Steimle et al. (2018) by considering the value functions under the best possible policy:

$$\min_{v\in\mathbb{R}^{|\mathbb{S}|\cdot|\mathcal{M}|}} \sum_{s\in\mathcal{S}} \mu^m(s)\, v^m(s) \tag{5a}$$

$$r^m(s,a) + \lambda \sum_{s'\in\mathcal{S}} p^m(s'|s,a)\, v^m(s') \le v^m(s), \forall s\in\mathcal{S}, m\in\mathcal{M}, a\in\mathcal{A} \tag{5b}$$

and the value functions under the worst-possible policy:

$$\max_{v\in\mathbb{R}^{|\mathbb{S}|\cdot|\mathcal{M}|}} \sum_{s\in\mathcal{S}} \mu^m(s)\, v^m(s) \tag{6a}$$

$$r^m(s,a) + \lambda \sum_{s'\in\mathcal{S}} p^m(s'|s,a)\, v^m(s') \ge v^m(s), \quad \forall s\in\mathcal{S},\ m\in\mathcal{M},\ a\in\mathcal{A} \tag{6b}$$

Considering these value functions, we select the big-Ms. to be specific to each state s and each model m by selecting the big-M for all corresponding constraints in (4c) such that:

$$M_{m,s} \ge \overline{v}^m(s) - \underline{v}^m(s)$$

where $\bar{v}^m(s)$ is the optimal value of $v^m(s)$ in (5a) and that $\underline{v}^m(s)$ is the optimal value of $v^m(s)$ in (6a).

B.2. Implementation of the percentile optimization formulation

Below we outline the specific version of Meraklı and Küçükyavuz (2019) proposed a similar MIP-based formulation for solving the percentile optimization formulation of the MMDP:

$$\max_{\pi\in\Pi, v\in\mathbb{R}^{|\mathbb{S}|\cdot|\mathcal{M}|}, z\in\mathbb{R}} \quad z \tag{7a}$$

$$\mathbb{P}\left(\sum_{s\in\mathcal{S}} \mu^m(s)\, v^m(s) \ge z\right) \ge 1-\eta, \tag{7b}$$

$$r^m(s,\pi(s)) + \lambda \sum_{s'\in\mathcal{S}} p^m(s'|s,\pi(s))\, v^m(s',\pi(s')) = v^m(s,\pi(s)),$$

$$\forall s\in\mathcal{S},\ m\in\mathcal{M}. \tag{7c}$$

In short, the formulation above maximizes the value z so that there is at least $(1-\eta)$ probability that the value function is at least z. In other words, the $(100\times\eta)$ th percentile is maximized.

Keeping with our previous notation, we define π as binary decision variables representing the policy and v as continuous variables representing the value function for each model and state. In the percentile optimization formulation, we also introduce binary variables y_m for each model which take on a value of 1 if the value in model m is at least z and zero otherwise. The constraint (8d) ensures that the y_m value will take on a value equal to 1 for models with values greater than or equal to z and all models with a value less than z will have a y_m value of 0. Constraint (8e) requires that there is no more than η chance that the models have values less than z.

$$\max_{\pi,v,y,z} z \tag{8a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}} \pi(a|s) = 1, \quad \forall s\in\mathcal{S} \tag{8b}$$

$$M\pi(a|s) + v^m(s) - \lambda \sum_{s'\in\mathcal{S}} p^m(s'|s,a)\, v^m(s') \le r^m(s,a) + M,\ \forall m\in\mathcal{M}, s\in\mathcal{S}, a\in\mathcal{A}, \tag{8c}$$

$$z + M'y_m - \sum_{s\in\mathcal{S}} \mu_1^m(s)\, v^m(s) \le M', \quad \forall m\in\mathcal{M}, \quad s\in\mathcal{S}, \tag{8d}$$

$$\sum_{m\in\mathcal{M}} \alpha_m y_m \ge 1-\eta \quad \forall m\in\mathcal{M}, \tag{8e}$$

$$\pi(a|s)\in\{0,1\}, \quad \forall a\in\mathcal{A},\ s\in\mathcal{S}, \tag{8f}$$

$$y_m\in\{0,1\}, \quad \forall m\in\mathcal{M}, \tag{8g}$$

$$v^m(s)\in\mathbb{R}, \quad \forall m\in\mathcal{M},\ s\in\mathcal{S}, \tag{8h}$$

$$z\in\mathbb{R}. \tag{8i}$$

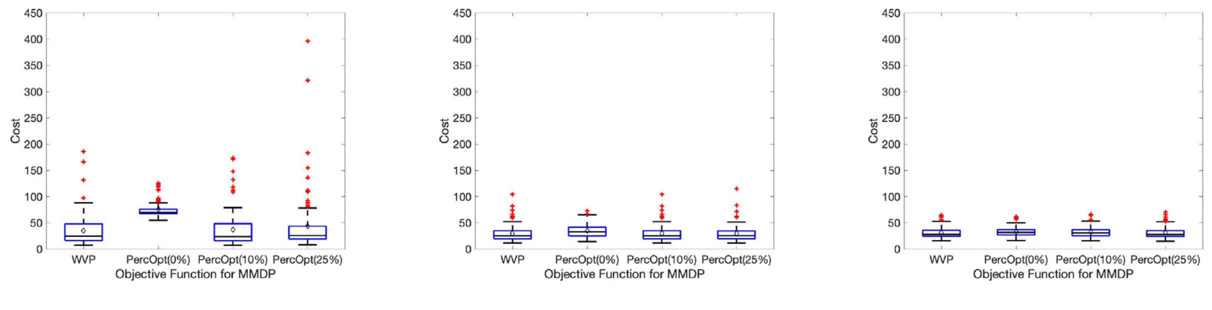
The big-Ms in (8c) are the same as those discussed in Appendix B.1 and there is a specific big-Ms for each state and model, which we denote $M_{m,s}$. The big-Ms. in constraint (8d) are tightened specific to each model, $M'_m = \sum_{s\in\mathcal{S}} M_{m,s}$.

Appendix C. Additional comparisons of WVP and PercOpt for the MM case study

In Figs. 3 and 4, we provide boxplots for two additional instances of the MM case study which compare the value functions for the WVP, PercOpt(0%), PercOpt(10%), and PercOpt(25%). We provide these boxplots to demonstrate that our findings were consistent across multiple instances.

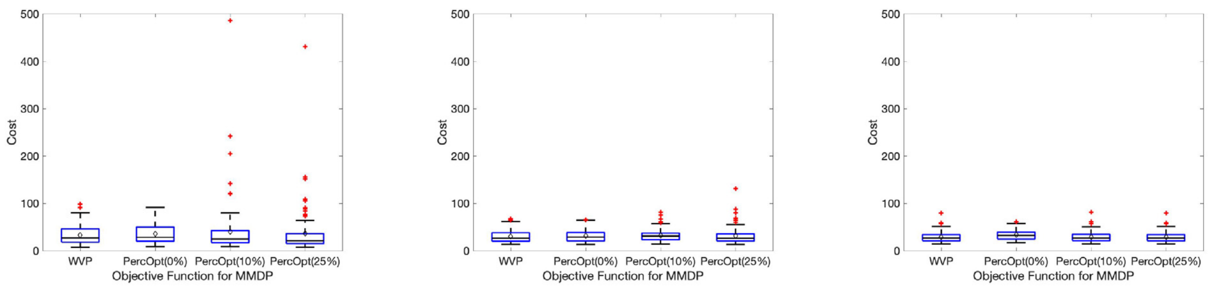
As with the instance shown in Fig. 2, we observe that the WVP policy tends to perform well with respect to the weighted performance across the MMDP models relative to the PercOpt formulations, especially when variance is high. In Fig. 3(a), the weighted performance of the WVP policy is substantially better than the weighted performance of the PercOpt(0%) policy, even though the latter has a better worst-case cost. In addition, the worst-case cost of the WVP policy is noticeably lower than that of PercOpt(25%), although the percentile optimization formulation for the 10th percentile slightly outperforms the WVP policy in the worst-case.

In the instance shown in Fig. 4, we observe that the WVP once again outperforms the 10th and 25th percentile optimization approaches in terms of both weighted and worst-case performance. The worst-case cost for the WVP is also quite similar to the worst-case cost for PercOpt(0%). As discussed in the main body of this article, the choice of MMDP objective function is more important when the variance across the models' parameters is high.



(a) $c=1$ (b) $c=5$ (c) $c=9$

Fig. 3. Boxplots showing the distribution of an MMDP's models' value functions for a second 100-model instance of the MM MMDP under different policies. The WVP policy is obtained via PB-B&B for the WVP objective function. The other policies are found using PB-B&B for the percentile optimization formulation that optimized the 0th percentile (worst-case), 10th percentile, and 25th percentile. In the MM MMDP, we aim to minimize cost.



(a) $c=1$ (b) $c=5$ (c) $c=9$

Fig. 4. Boxplots showing the distribution of an MMDP's models' value functions for a third 100-model instance of the MM MMDP under different policies. The WVP policy is obtained via PB-B&B for the WVP objective function. The other policies are found using PB-B&B for the percentile optimization formulation that optimized the 0th percentile (worst-case), 10th percentile, and 25th percentile. In the MM MMDP, we aim to minimize cost.

Appendix D. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.cor.2020.105108.

References

Boucherie, R.J., Van Dijk, N.M., 2017. Markov Decision Processes in Practice, vol. 248, Springer.

Buchholz, P., Scheftelowitsch, D., 2019. Computation of weighted sums of rewards for concurrent MDPs. Math. Methods Oper. Res. 89 (1), 1–42.

Delage, E., Mannor, S., 2010. Percentile optimization for Markov decision processes with parameter uncertainty. Oper. Res. 58 (1), 203–313.

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

Iyengar, G.N., 2005. Robust dynamic programming. Math. Oper. Res. 30 (2), 257–280.

Mannor, S., Mebel, O., Xu, H., 2016. Robust MDPs with k-rectangular uncertainty. Math. Oper. Res. 41 (4), 1484–1509.

Mannor, S., Simester, D., Sun, P., Tsitsiklis, J.N., 2007. Bias and variance approximation in value function estimates. Manage. Sci. 53 (2), 308–322.

Mannor, S., Xu, H., 2019. Data-driven methods for Markov decision problems with parameter uncertainty. Oper. Res. Manage. Sci. Age Anal., 101–129 (INFORMS).

Meraklı, M., Küçükyavuz, S., 2019. Risk aversion to parameter uncertainty in Markov decision processes with an application to slow-onset disaster relief. IISE Trans., 1–21. URL:https://doi.org/10.1080/24725854.2019.1674464.

Nilim, A., El Ghaoui, L., 2005. Robust control of Markov decision processes with uncertain transition matrices. Oper. Res. 53 (5), 780–798.

Puterman, M.L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York.

Shechter, S.M., Bailey, M.D., Schaefer, A.J., Roberts, M.S., 2008. The optimal time to initiate HIV therapy under ordered health states. Oper. Res. 56 (1), 20–33.

Steimle, L.N., Ahluwalia, V., Kamdar, C., Denton, B.T., 2019. Decomposition methods for solving Markov decision processes with multiple models of the parameters. Optim. Online URL: http://www.optimization-online.org/DB_FILE/2018/11/6958.pdf.

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

Wolsey, L.A., 1998. Integer Programming. Wiley.

Xu, H., Mannor, S., et al., 2012. Distributionally robust Markov decision processes. Math. Oper. Res. 37 (2), 288–300.

Zhang, Y., Steimle, L., Denton, B.T., 2019. Robust Markov decision processes for medical treatment decisions. Optim. Online.