Decomposition Methods for Solving Markov Decision Processes with Multiple Models of the Parameters

Lauren N. Steimle
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332
steimle@gatech.edu

Vinayak Ahluwalia, Charmee Kamdar
Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109,
vahluw@umich.edu,ckamdar@umich.edu

Brian T. Denton
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, btdenton@umich.edu

We consider the problem of decision-making in Markov decision processes (MDPs) when the reward or transition probability parameters are not known with certainty. We consider an approach in which the decision maker (DM) considers multiple models of the parameters for an MDP and wishes to find a policy that optimizes an objective function that considers the performance with respect to each model, such as maximizing the expected performance or maximizing worst-case performance. Existing solution methods rely on mixed-integer program (MIP) formulations, but have previously been limited to small instances due to the computational complexity. In this article, we present branch-and-cut (B&C) and policy-based branch-and-bound (PB-B&B) solution methods that leverage the decomposable structure of the problem and allow for the solution of MDPs that consider many models of the parameters. Numerical experiments show that a customized implementation of PB-B&B significantly outperforms the MIP-based solution methods and that the variance among model parameters can be an important factor in the value of solving these problems.

Key words: Markov decision processes, dynamic programming, parameter ambiguity, decomposition, stochastic programming

1. Introduction

Markov decision processes (MDPs) are mathematical optimization models that have found success in informing optimal decision-making under uncertainty when decisions are made sequentially over time. However, the optimal decisions can be sensitive to the MDP’s parameters. Unfortunately, parameter ambiguity is common in MDPs as parameters are usually estimated from data and may
rely on expert opinion. The ambiguity in the modeling process creates a dilemma for the decision maker as to the best choice of parameters to use when solving the MDP, and in turn the decision maker (DM) may be left unsure as to the best course of action.

Recently, there has been a line of research on mitigating the impact of parameter ambiguity by incorporating multiple scenarios of the parameters into the solution of the MDP. Each scenario is referred to as a “model” of the MDP where all models are defined on the same state space, action space, and set of decision epochs, but each model may vary in terms of its rewards, transition probabilities, and initial distribution. Adulyasak et al. (2015), Buchholz and Scheftelowitsch (2018), and Steimle et al. (2019) proposed designing policies that maximize a weighted value across multiple models. Ahmed et al. (2017) proposed minimizing the maximum regret with respect to each model for finite horizon MDPs. Adulyasak et al. (2015) and Meraklı and Küçükyavuz (2019) proposed a percentile optimization approach for finite-horizon and infinite-horizon MDPs, respectively. Thus far, the exact solution methods for this problem have relied on mixed-integer program (MIP) formulations where binary decision variables encode the policy and continuous variables encode the value functions for each model of the MDP. Although the MIP can be used to find exact solutions, these problems are NP-hard (Steimle et al. 2019, Buchholz and Scheftelowitsch 2018, Delage and Mannor 2009), and the MIP solution methods for this problem have been limited to small MDPs thus far.

In this article, we present new computational methods for solving MDPs with multiple models of the parameters and use these methods to generate new intuition behind when it is important to consider parameter ambiguity in MDPs. Our contributions are as follows:

- **We present new branch-and-cut (B&C) and policy-based branch-and-bound (PB-B&B) decomposition methods that leverage problem structure.** In this article, we take special consideration of the structure of these problems and are the first to consider decomposition methods for solving finite-horizon MDPs with multiple models, which present unique challenges due to the scale of the policy-space. For the weighted value case, we consider the extensive form
of a previously proposed MIP formulation. In the extensive form, the entire MIP is solved without any special consideration of the problem structure. We then present a B&C method which follows from decomposition algorithms in the stochastic programming literature and a custom PB-B&B method that exploits the decomposable nature of the problem. We also discuss how the custom PB-B&B procedure is easily modified to extend to other ambiguity-sensitive objective functions that incorporate multiple models of the parameters, such as maximizing the performance of the worst-case model.

- **We provide the first numerical study comparing the extensive form, B&C, and PB-B&B solution approaches.** We provide a numerical study that compares the three solution methods on various problem sizes and characteristics. Across the problem sizes and characteristics considered, the PB-B&B outperforms the other proposed methods and is able to solve test instances that would be previously unsolvable only using MIP-based approaches. These findings suggest that specialized algorithms may lead to computational gains for MDPs with multiple models.

- **We are the first to study the relationship between the variance in models’ parameters and the value of considering multiple models in MDPs.** Due to our ability to solve MDPs with multiple models for a variety of characteristics, we are able to conduct the first empirical analysis that investigates the value of the stochastic solution (VSS) in the context of MDPs. We are the first to show that variance among the models’ parameters can impact the VSS, computation time, and ambiguity-award objective function values. As we will show, instances with higher variance among the models’ parameters tend to have higher VSS, higher computation time, and larger differences between the objective function values that the DM might use to mitigate ambiguity. In other situations when the models are quite similar, solving a simpler mean value problem may lead to near-optimal solutions.

The remainder of this article is structured as follows: In Section 2, we provide background on parameter ambiguity in MDPs and related solution methods for solving MDPs with multiple models of parameters. In Section 3, we introduce the multi-model Markov decision process (MMDP), state
the Weighted Value Problem (WVP), and present other multi-model objectives for the MMDP. In Section 4, we describe two methods, a B&C procedure and a custom PB-B&B procedure, that leverage problem structure to solve MMDPs. We compare the solution methods in Section 5 using a set of test instances based on a medical decision-making problem. Finally, we conclude with a summary and discussion of the most important contributions of this article in Section 6.

2. Background

The MDP is a generalization of a Markov chain in which a DM can take actions to influence the stochastic evolution of the system (Puterman 1994, Boucherie and Van Dijk 2017). In this article, we consider a finite state-space \( S \equiv \{1, 2, \ldots, |S|\} \) and a finite action-space, \( A \equiv \{1, 2, \ldots, |A|\} \). The system begins in state \( s_1 \in S \) according to the initial distribution \( \mu_1 \in \Delta(S) \), where \( \Delta(\cdot) \) denotes the set of probability measures on the discrete set. At each \emph{decision epoch} \( t \in T \equiv \{1, \ldots, T\} \), the DM observes the state \( s_t \in S \), selects an action \( a_t \in A \), and receives a reward \( r_t(s_t, a_t) \in \mathbb{R} \). Then, the system transitions to a new state \( s_{t+1} \in S \) with probability \( p_t(s_{t+1} \mid s_t, a_t) \in [0, 1] \). The DM selects the last action at time \( T \) which may influence which state is observed at time \( T + 1 \) through the transition probabilities. Upon reaching \( s_{T+1} \in S \) at time \( T + 1 \), the DM receives a terminal reward of \( r_{T+1}(s_{T+1}) \in \mathbb{R} \). Future rewards are discounted at a rate of \( \alpha \in (0, 1] \) which accounts for the preference for rewards received now over rewards received in the future. In this article, we assume without loss of generality that the discount factor is already incorporated into the reward definition.

We will refer to the times at which the DM selects an action as the set of decision epochs, \( T \), the set of rewards as \( R \in \mathbb{R}^{[S \times A \times T]} \), and the set of transition probabilities as \( P \in \mathbb{R}^{[S \times A \times S \times T]} \) with elements satisfying \( p_t(s_{t+1} \mid s_t, a_t) \in [0, 1] \) and \( \sum_{s_{t+1} \in S} p_t(s_{t+1} \mid s_t, a_t) = 1, \forall t \in T, s_t \in S, a_t \in A \). Throughout the remainder of this article, we will use the tuple \( (T, S, A, R, P, \mu) \) to summarize the parameters of an MDP.

The strategy by which the DM selects the action for each state at decision epoch \( t \in T \) is called a \emph{decision rule}, \( \pi_t \in \Pi_t \), and the set of decision rules over the planning horizon is called a \emph{policy},
The goal of the DM is to specify a policy that prescribes the decision rules that maximize the expected rewards over the planning horizon:

$$\max_{\pi \in \Pi} E^{P, \pi, \mu_1} \left[ \sum_{t=1}^{T} r_t(s_t, a_t) + r_{T+1}(s_{T+1}) \right].$$ (1)

It has been shown that a Markov deterministic policy will achieve the optimum in (1) (Puterman 1994, Theorem 4.4.2), which means that the optimal action to take will depend only on the current state of the system and the decision epoch. In this article, we restrict our attention to decision rules of the form $\pi_t = (\pi_t(1), \ldots, \pi_t(|S|))$ where $\pi_t(s) : S \times T \rightarrow A$ because although in general history-dependent policies may be optimal for MDPs with multiple models (Steimle et al. 2019), they are more difficult to implement in practice (Steimle et al. 2019, Meraklı and Küçükyavuz 2019).

Standard MDPs have been carefully studied for many years. Numerous solution methods have been developed including value iteration, policy iteration, linear programming, and others for infinite-horizon MDPs (See Chapter 6 of Puterman (1994) for a review of these and other related methods). Backward induction is the most commonly used method for finite-horizon MDPs (see Chapter 4 of Puterman (1994)) which are the focus of this article. More recently, a number of approaches have been proposed to deal with the issue of parameter ambiguity in the context of MDPs. The most common is the “max-min” approach in which the DM seeks a policy that maximizes the worst-case performance selected by an “adversary” that can minimize rewards by controlling the transition probabilities that are allowed to vary within an uncertainty set. A common way to estimate the uncertainty set is to use confidence intervals around point estimates of the transition probabilities. Methods like this satisfy the well-known rectangularity property, which implies independence across rows of the transition probability matrix, making the problem easy to solve (Iyengar 2005, Nilim and El Ghaoui 2005). While such an approach is computationally tractable, it can give rise to overly-conservative policies because the DM must account for the possibility that parameters for each state-action-time triplet will take on their worst-case values simultaneously. Ways to mitigate this problem in the context of the rectangularity assumption have been explored in the recent work of Zhang et al. (2017). More recent work on robust dynamic

A recent stream of research on MDPs with parameter ambiguity assumes multiple models of the MDP parameters when determining the optimal policy. Steimle et al. (2019), Adulyasak et al. (2015), and Buchholz and Scheftelowitsch (2018) proposed the Weighted Value Problem (WVP) for MMDPs in which the DM seeks to find a policy that maximizes the weighted performance with respect to each model of the MDP. Adulyasak et al. (2015) and Meraklı and Küçükyavuz (2019) consider an ambiguity-averse DM through the use of a percentile optimization formulation of MMDPs. Ahmed et al. (2017) consider a min-max-regret approach to considering multiple models of the parameters wherein the DM wishes to minimize the worst-case gap between the best possible performance in a given model and the performance achieved by the specified policy. The exact solution methods proposed in the articles above all rely on MIP-based approaches, while the custom PB-B&B approach we present later does not rely on an MIP formulation of the MMDP. Steimle et al. (2019) and Buchholz and Scheftelowitsch (2018) demonstrate the computational complexity of these types of problems, making the case for specialized algorithms such as the ones we present in this article.

The WVP for an MMDP is an NP-hard problem and has close ties to two-stage stochastic integer programming. In a two-stage stochastic program, ambiguous problem parameters are treated as random variables where collective outcomes define scenarios. The DM must take some first-stage decisions before these random variables are realized. Upon the realization of the problem parameters, the DM may take some recourse actions in the second-stage to adapt to the information gained. In many cases, the random variables representing the problem data have finite support and the outcomes of these random variables are referred to as scenarios. We refer the reader to Birge and Louveaux (1997) and Shapiro et al. (2009) for more information on stochastic programming.
Through the lens of stochastic programming, the MMDP can be viewed as a two-stage stochastic integer program in which the DM specifies a policy that is encoded through the use of first stage binary variables to indicate whether or not to take a given action for each state-time pair. Then, the DM observes which model of the MDP is realized, which subsequently determines the corresponding value functions for each model, which are represented using continuous variables. This two-stage stochastic integer program has a fixed technology matrix and a random recourse matrix with a finite number of scenarios. The problem can be written in its extensive form (EF) which contains decision variables representing the first-stage variables as well as second-stage decision variables for each scenario. However, the EF of the problem can become quite large and potentially inefficient to solve as the number of models grows. Fortunately, the constraint matrix in the EF is block-separable except for the columns corresponding to the first-stage decision variables. Due to this structure, two-stage stochastic programs lend themselves well to divide-and-conquer type algorithms, such as Benders decomposition (also known as the L-shaped method) (Benders 1962, Van Slyke and Wets 1969).

The stochastic programming formulation of the MMDP has some features that require special algorithmic consideration. First, the binary first-stage decision variables require integer programming methods, such as branch-and-bound (B&B). Early work to address this problem includes that of Wollmer (1980), which proposed a cutting plane decomposition method, and Laporte and Louveaux (1993) which proposed a B&C scheme wherein the feasibility and optimality cuts are added within a B&B framework for solving the master problem. Second, the MMDP has relatively complete recourse and therefore, feasibility cuts are not required. Third, logical constraints are required to enforce that the value functions in each of the models of the MMDP correctly correspond to the policy encoded by the binary variables. The logical constraints are enforced through the introduction of notorious “big-Ms” which weaken the linear programming relaxation of the EF of the MIP and cause problems for potential decomposition methods. Logic-based cuts have been proposed to strengthen formulations and avoid the explicit use of the big-M values (Hooker and Ottosson 2003, Codato and Fischetti 2006, Ntaimo 2010, Luedtke 2014).
In this article, we propose a Benders-based B&C method to solve the MMDP, and we propose a custom PB-B&B method that does not explicitly consider an MIP formulation, eliminates the need to consider the big-M’s in the MMDP, and further exploits the unique structural properties of the MDP subproblems.

3. Problem statement

In this section, we introduce the MMDP and several objective functions that consider the performance of a policy with respect to the finite set of models of the MDP.

3.1. The Multi-model Markov decision process

The MMDP is a tuple \((T, S, A, M, \Lambda)\) where \(T\) is the set of decision epochs, \(S\) and \(A\) are the state and action spaces respectively, \(M\) is the finite discrete set of models, and \(\Lambda := \{\lambda_1, \ldots, \lambda_{|M|}\}\) is the set of exogenous model weights with \(\lambda_m \in (0, 1), \forall m \in M\) and \(\sum_{m \in M} \lambda_m = 1\). Each model \(m \in M\) is an MDP, \((T, S, A, R^m, P^m, \mu^m)\), with a unique combination of rewards, transition probabilities, and initial distributions. The requirement that \(\lambda_m \in (0, 1)\) is to avoid the trivial cases: If there exists a model \(m \in M\) such that \(\lambda_m = 1\), the MMDP would reduce to a standard MDP. If there exists a model \(m \in M\) such that \(\lambda_m = 0\), then the MMDP would reduce to an MMDP with a smaller set of models, \(M \setminus \{m\}\). In this article, we consider that the DM wishes to design a policy \(\pi\) that performs well with respect to the different models in the MMDP. In the following sections, we will describe several proposed objective functions for such DMs, beginning with the WVP.

3.2. Weighted value problem

In this section, we introduce the WVP and present a MIP formulation for solving it. In the WVP, the DM considers the expected rewards of the specified policy in the multiple models of the MDP. The value of a policy \(\pi \in \Pi\) in model \(m \in M\) is given by its expected rewards evaluated with model \(m\)’s parameters:

\[
v^m(\pi) := \mathbb{E}^\pi_{P^m, \mu^m} \left[ \sum_{t=1}^T r^m_t(s_t, a_t) + r^m_{T+1}(s_{T+1}) \right] \in \mathbb{R}.
\]
We associate any policy, \( \pi \in \Pi \), for the MMDP with its weighted value:

\[
W(\pi) := \sum_{m \in M} \lambda_m v^m(\pi) = \sum_{m \in M} \lambda_m E^{\pi,P^m,\mu^m} \left[ \sum_{t=1}^{T} r_t^m(s_t,a_t) + r_{T+1}^m(s_{T+1}) \right] \in \mathbb{R}.
\]  

(3)

Thus, we consider the WVP in which the goal of the DM is to find the policy \( \pi \in \Pi \) that maximizes the weighted value defined in (3). Given an MMDP \((T,S,A,M,\Lambda)\), the WVP is defined as the problem of finding a solution to:

\[
W^* := \max_{\pi \in \Pi} W(\pi) = \max_{\pi \in \Pi} \left\{ \sum_{m \in M} \lambda_m E^{\pi,P^m,\mu^m} \left[ \sum_{t=1}^{T} r_t^m(s_t,a_t) + r_{T+1}^m(s_{T+1}) \right] \right\} \in \mathbb{R},
\]  

(4)

and a set of policies \( \Pi^* := \{ \pi : W(\pi) = W^* \} \subseteq \Pi \) that achieve the maximum in (4).

Steimle et al. (2019) showed that the WVP for the Markov deterministic policy class, \( \Pi^{MD} \), for an MMDP is a hard problem:

**PROPOSITION 1 (Steimle et al. (2019)).** Solving the weighted value problem with \( \Pi = \Pi^{MD} \) is NP-hard.

Steimle et al. (2019) propose the MIP formulation in (5) as an exact solution method for the MMDP:

\[
\max_{\pi,v} \sum_{m \in M} \sum_{s \in S} \lambda_m \mu^m_1(s) v^m_1(s) \quad \text{(5a)}
\]

s.t.

\[
\sum_{a \in A} \pi_t(a|s) = 1, \quad \forall s \in S, t \in T, \quad \text{(5b)}
\]

\[
M \pi_t(a|s) + v^m_t(s) - \sum_{s' \in S} p^m_t(s'|s,a) v^m_{t+1}(s') \leq r^m_t(s,a) + M, \forall m \in M, s \in S, a \in A, t \in T, \quad \text{(5c)}
\]

\[
v^m_{T+1}(s) \leq r^m_{T+1}(s), \quad \forall m \in M, s \in S, \quad \text{(5d)}
\]

\[
\pi_t(a|s) \in \{0,1\}, \quad \forall a \in A, s \in S, t \in T. \quad \text{(5e)}
\]

The MIP formulation extends the standard linear programming formulation of an MDP (Puterman 1994, §6.9) to include continuous variables representing the value function for each model of the
MMDPs and modifies the epigraph constraints to enforce that each model is evaluated according to the same policy. We define model-specific value function continuous variables such that \( v_t^m(s) \in \mathbb{R} \) represents the value-to-go from state \( s \) at decision epoch \( t \) in model \( m \). We define the policy decision variables:

\[
\pi_t(a | s) := \begin{cases} 
1 & \text{if the policy states to take action } a \text{ in state } s \text{ at decision epoch } t, \\
0 & \text{otherwise}, 
\end{cases}
\]

\[ \forall a \in A, s \in S, t \in T. \]

Throughout this article, we will refer to (5) as the extensive form (EF) of the MMDP. Constraints (5b) and (5e) are used to encode a valid Markov deterministic policy. Constraint (5c) is a logic-based constraint which uses “big-M”s to enforce the relationship between the value functions in each model and the policy, so long as \( M \in \mathbb{R} \) is selected sufficiently large enough. Together, constraints (5c) and (5d) ensure that the value functions in each model correspond to the policy encoded via the binary \( \pi \) variables. As mentioned previously, the MMDP can be viewed as a two-stage stochastic integer program with binary first-stage decision variables, continuous second-stage decision variables, and relatively complete random recourse. Given a fixed policy, the MMDP reduces to the evaluation of \( |\mathcal{M}| \) Markov chains. However, policy optimization for an MMDP is challenging due to the coupling constraints that enforce the same policy is used in each model in the MMDP.

The view of the WVP as a two-stage stochastic program lends itself to measures of the impact of uncertainty in the MDP parameters. The VSS is the value added by solving the WVP given in (4) rather than solving a simpler version of the MMDP, called the mean value problem (MVP), wherein the DM solves a single MDP with parameters that are obtained by taking the weighted average of the parameters from each model of the MMDP. Another measure of uncertainty is the expected value of perfect information (EVPI) which is the expected amount that the DM would pay to know with which model of the MMDP they are interacting. These metrics allow for a better understanding of the impact of uncertainty on the performance of the DM’s decisions and how valuable it would be to obtain better information about the MDP parameters.
3.3. Other multi-model formulations

The WVP is just one formulation that considers multiple models of the MDP’s parameters. The WVP may be appropriate for a DM who is risk-neutral to model ambiguity. However, there is considerable evidence that some DMs may be risk-averse to the ambiguity which can affect decision-making (Ellsberg 1961). Therefore, it could be desirable to find a policy that offers more protection against the possible outcomes of the ambiguity and helps guide DMs who exhibit some degree of ambiguity aversion. To do so, we present other multi-model formulations of the MMDP which reflect preferences beyond neutrality towards parameter ambiguity. Some alternative approaches to addressing risk include maximizing the worst-case (max-min), minimizing the maximum regret (min-max-regret), and percentile optimization (PercOpt). In the max-min approach, the DM seeks to find a policy that will maximize the expected rewards in the worst-case realization of the ambiguity, i.e.,

\[
\max_{\pi \in \Pi} \min_{m \in M} v^m(\pi) \tag{6}
\]

The min-max-regret criterion considers the performance relative to the best possible performance in that model. The regret for policy \( \pi \) in model \( m \), \( \ell(\pi, m) \), is the difference between the optimal value in model \( m \) and the value achieved by policy \( \pi \) in model \( m \):

\[
\ell(\pi, m) = \max_{\bar{\pi} \in \Pi} v^m(\bar{\pi}) - v^m(\pi) \tag{7}
\]

The min-max-regret criterion then seeks to find the policy \( \pi \in \Pi \) that minimizes the maximum regret:

\[
\min_{\pi \in \Pi} \max_{m \in M} \ell(\pi, m) \tag{8}
\]

In the PercOpt approach, the DM selects a level of confidence \( \epsilon \in [0, 1] \), and wants to maximize the \( \epsilon \)-percentile of the value in the models, \( z \):

\[
\max_{\pi \in \Pi} \max_{z \in \mathbb{R}} z \text{ s.t. } \mathbb{P}(v^m(\pi) \geq z) \geq 1 - \epsilon. \tag{9}
\]

Each of the multi-model objectives listed above can be formulated as MIPs as shown in the appendix.
4. Methods that leverage problem structure

In this section, we describe two methods that leverage the special structure of the MMDP for the WVP. The first is a B&C approach applied to the proposed MIP formulation. The second is a custom PB-B&B approach which considers the problem as $|\mathcal{M}|$ independent MDPs with coupling requirements on the policy. Later, in Section 5, we compare these two methods to a generic branch-and-bound implementation using a commercial MIP solver for test cases based on an example problem in the context of machine maintenance. Although we develop these approaches in the context of the WVP, the custom PB-B&B easily extends to the other multi-model formulations presented in Section 3.3.

4.1. Branch-and-cut for the MMDP

In this section, we present a B&C scheme for solving the MMDP which is in the same vein as Benders decomposition for stochastic programming. Benders decomposition breaks the extensive form of a stochastic program into a master problem and subproblems. The master problem typically only considers “complicating variables” while the subproblems will consider the other variables assuming fixed values of the complicating variables. In the context of stochastic programming, typically one solves a “relaxed master problem” which involves only the first-stage decisions and a subset of the constraints required to specify the complete optimization problem. Then, one uses duality for the subproblems to subsequently add constraints, or cuts, to the relaxed master problem to enforce the feasibility and optimality of the proposed first-stage solutions from the relaxed master problem.

Our B&C approach uses a master problem that includes the binary policy variables and enforces constraints (5c) and (5d) via cutting planes that are generated from each model’s subproblem. We begin by describing the decomposition of (5) into the master problem and subproblems, and then we describe a B&C algorithm that uses this decomposition approach.
First, we describe the decomposition of (5) into a master problem and model-specific subproblems. The value of a fixed policy, $\pi$, in a model $m$ can be determined by solving a linear program, which we refer to as Subproblem$^m(\pi)$:

$$\max \sum_{s \in S} \mu^m_1(s) v_1^m(s)$$

subject to

$$v_1^m(s) - \sum_{s' \in S} p_{1}^m(s|s,a) v_{1+1}^m(s') \leq r_1^m(s,a) + M - M \pi_1(a|s), \quad \forall s \in S, a \in A, t \in T,$$

$$v_{T+1}^m(s) \leq r_{T+1}^m(s), \quad \forall s \in S.$$  \hspace{1cm} (10a)

For constraints of the form (10b), we assign dual variables $x_{1}^m(s,a) \in \mathbb{R}$ and for constraints of the form (10c) we assign dual variables $x_{T+1}^m(s) \in \mathbb{R}$. Then, the dual of Subproblem$^m(\pi)$ is:

$$\min \sum_{s \in S} \sum_{a \in A} \sum_{t \in T} (r_{t}^m(s,a) + M(1 - \pi_{t}(a|s))) x_{t}^m(s,a) + \sum_{s \in S} r_{T+1}^m(s) x_{T+1}^m(s)$$

subject to

$$\sum_{a \in A} x_{t}^m(s,a) = \mu_{1}^m(s), \quad \forall s \in S,$$

$$\sum_{a \in A} x_{t}^m(s,a) - \sum_{s' \in S} \sum_{a' \in A} p_{t}^m(s'|s,a) x_{t-1}^m(s',a') = 0, \quad \forall s \in S, t \in T,$$

$$x_{T+1}^m(s) = 0, \quad \forall s \in S.$$

Given the policy, $\pi$, (11) is easy to solve because the constraint in (10b) corresponding to the tuple $(s,a,t)$ is binding if and only if $\pi_{t}(a|s) = 1$ so long as $M$ is selected sufficiently large enough to enforce the logical relationship between the value functions and the policy. Therefore, given a policy $\pi_{t}(a|s)$, we have already identified an optimal basis for the subproblems. Because the primal constraint in (10b) corresponding to $(s,a,t)$ is non-binding if $\pi_{t}(a|s) = 0$, it follows from complementary slackness that $\pi_{t}(a|s) = 0 \Rightarrow x_{t}^m(s,a) = 0$.

If we were to use a standard commercial solver to implement this branch-and-cut scheme, we could use the dual of formulation in (11) to generate optimality cuts. However, as we will show in...
Propositions 2 and 3, we can leverage a policy evaluation approach to quickly generate optimality cuts without the need of solving a linear program directly.

**Proposition 2.** The following forward substitution gives an optimal solution to (11):

\[
x_m(s,a) = \begin{cases} 
\mu_m(s) & \text{if } \pi_1(a|s) = 1, \\
0 & \text{otherwise},
\end{cases} \quad \forall s \in S, a \in A,
\]

(12)

\[
x_m(s,a) = \begin{cases} 
\sum_{s' \in S} \sum_{a' \in A} p_{m-1}(s'|s', a') x_{m-1}(s', a'), & \text{if } \pi_t(a|s) = 1, \\
0, & \text{otherwise},
\end{cases} \quad \forall s \in S, a \in A, t \in T \setminus \{1\},
\]

(13)

\[
x_{T+1}(s) = \sum_{s' \in S} \sum_{a' \in A} p_T(s'|s', a') x_T(s', a'), \quad \forall s \in S.
\]

(14)

**Proof.** First, we show that \( x \) as defined in (12)-(14) is feasible for (11). The solution generated by equation set (12)-(14) satisfies (11b) because

\[
\sum_{a \in A} x_1(s,a) = \sum_{a \in A} \mu_1(s) \pi_1(a|s) = \mu_1(s) \sum_{a \in A} \pi_1(a|s) = \mu_1(s), \quad \forall s \in S.
\]

The solution satisfies (11c) because

\[
\sum_{a \in A} x_t(s,a) = \sum_{a \in A} \sum_{s' \in S} \sum_{a' \in A} p_{m-1}(s'|s', a') x_{m-1}(s', a') \pi_t(a|s) \\
= \sum_{a \in A} \pi_t(a|s) \sum_{s' \in S} \sum_{a' \in A} p_{m-1}(s'|s', a') x_{m-1}(s', a') \\
= \sum_{s' \in S} \sum_{a' \in A} p_{m-1}(s'|s', a') x_{m-1}(s', a'), \forall s \in S, t \in T \setminus \{1\},
\]

and (11d) is satisfied by definition. The non-negativity constraints are also satisfied.

Next, we show that \( x \) as defined in (12)-(14) is optimal. Consider the following feasible solution to Subproblem\((m, \pi)\):

\[
u_{T+1}^m(s) = r_T^m(s), \quad \forall s \in S,
\]
\[ v_t^m(s) = \begin{cases} r_t^m(s,a) + \sum_{s' \in S} p_t^m(s'|s,a)v_{t+1}^m(s') & \text{if } \pi_t(a|s) = 1, \forall s \in S, a \in A, t \in T, \\ 0 & \text{otherwise}. \end{cases} \]

The solutions \( v \) and \( x \) are feasible for Subproblem\(^m(\pi) \) and its dual. For all \((s,a,t) \in S \times A \times T \) such that \( \pi_t(a|s) = 0 \), \( x_t^m(s,a) = 0 \), and for all \((s,a,t) \in S \times A \times T \) such that \( \pi_t(a|s) = 1 \), \( v_t^m(s) - \sum_{s' \in S} v_{t+1}^m(s') = r_t^m(s,a) + M(1 - \pi_t(a|s)) \) which implies

\[
x_t^m(s,a) \left( v_t^m(s) - r_t^m(s,a) - M + M\pi_t(a|s) - \sum_{s' \in S} p_t(s'|s,a)v_{t+1}^m(s') \right) = 0, \quad \forall s \in S, a \in A, t \in T,
\]

\[
x_{t+1}^m(s) \left( v_{t+1}^m(s) - r_{t+1}^m(s) \right) = 0, \quad \forall s \in S,
\]

\[
v_t^m(s) \left( \sum_{a \in A} x_t^m(s,a) - \mu_t^m(s) \right) = 0, \quad \forall s \in S,
\]

\[
x_t^m(s,a) - \sum_{s' \in S} \sum_{a' \in A} p_{t-1}(s'|s,a)x_{t-1}^m(s',a') = 0, \quad \forall s \in S, t \in T \setminus \{1\},
\]

\[
v_{T+1}^m(s) \left( x_{T+1}^m(s) - \sum_{s' \in S} \sum_{a' \in A} p_{T}(s'|s,a')x_T^m(s',a') \right) = 0, \quad \forall s \in S.
\]

Thus, by complementary slackness, \( v \) and \( x \) are optimal solutions for their respective problems.

\( \square \)

Proposition 2 provides a means to compute the solution to subproblems without the need to solve a linear program. The importance of this from a computational perspective is motivated by the following proposition.

**PROPOSITION 3.** (11) can be solved in \( O(T|S|^2) \) time.

**Proof.** Computing the values in (12) can be completed in \( O(|S|) \) time. For a given state, (13) can be completed in \( O(|S|) \) time because exactly one action is taken in each state. Therefore, for a given decision epoch, the solution of (13) requires \( O(|S|^2) \) time and thus, the total time required for substitutions in (13) can be completed in \( O(T|S|^2) \) time. Equation set (14) also requires \( O(|S|^2) \) time.

For an optimal policy \( \pi^* \) for a given subproblem, the dual solution has a corresponding objective value of

\[
z_m := \sum_{s \in S} \sum_{a \in A} \sum_{t \in T} \left( r_t^m(s,a) + M(1 - \pi_t^*(a|s)) \right) x_t^m(s,a) + \sum_{s \in S} r_{T+1}^m(s)x_{T+1}^m(s). \quad (15)
\]
Therefore, $z_m$ is a tight upper bound on the value that can be obtained by using the fixed policy $\pi^*$ in model $m$. Further, the hyperplane in (15) can be used to bound the value of other policies in model $m$.

To this point, we have described how the subproblem can be solved quickly for a fixed value of the policy $\pi$. We now describe the master problem (16) which is a MIP that uses binary variables, $\pi \in \{0,1\}^{|S \times A \times T|}$, to encode the policy and continuous variables, $\theta \in \mathbb{R}^m$, to be used as surrogate variables for the value functions in each model. The master problem incorporates optimality cuts generated from the subproblems corresponding to previously investigated policies, $\pi^k, k = 1, \ldots, K$.

\[
\max_{\pi, \theta} \sum_{m \in M} \lambda_m \theta_m
\]

s.t.

\[
\sum_{a \in A} \pi_t(a|s) = 1, \quad \forall s \in S, t \in T,
\]

\[
\theta_m \leq \sum_{(s,a,t) \in S \times A \times T} x^m_k(s,a,t)(r^m_t(s,a) + M - M\pi_t(a|s)) + \sum_{s \in S} r^m_{T+1}(s)x^m_k(s,T+1), \quad \forall s \in S, a \in A, t \in T, k = 1, \ldots, K,
\]

\[
v^m_T(s) \leq v^m_{T+1}(s), \quad \forall s \in S,
\]

\[
\pi_t(a|s) \in \{0,1\}, \quad \forall a \in A, s \in S, t \in T
\]

Algorithm 1 is a B&C algorithm for solving the MMDP that uses the decomposition described above. The algorithm we present largely follows from the Integer L-Shaped Method (Laporte and Louveaux 1993), but leverages the special structure of the subproblems to quickly generate optimality cuts.

### 4.2. Custom policy-based branch-and-bound for the MMDP

In this section, we present a customized PB-B&B framework that can be used to solve MMDPs without relying on a MIP formulation. We begin by introducing the general framework of the
Algorithm 1 Branch-and-cut for the MMDP

1: Set \( k \leftarrow 0, \nu \leftarrow 0 \). Let \( \bar{\pi} \) be any feasible policy and \( \bar{z} \) be the corresponding weighted value.

2: Select a pending node from the list; if none exists, stop.

3: Set \( \nu \leftarrow \nu + 1 \); Solve the current instance of problem (16).

4: \textbf{if} Current problem has no feasible solution \textbf{then}
5: Fathom the current node; Return to Step 2.
6: \textbf{else} Let \((\pi^\nu, \theta^\nu)\) be an optimal solution to the current problem.
7: \textbf{end if}

8: \textbf{if} \( \theta^\nu < \bar{z} \) then \hspace{1cm} \(\triangleright\) The policy described by \( \pi^\nu \) can be pruned by bound
9: Fathom the current node. Return to Step 2.
10: \textbf{end if}

11: \textbf{if} The current solution \( \pi^\nu \) violates an integer constraint \textbf{then}
12: Create two new pendant nodes; Return to Step 2.
13: \textbf{end if}

14: Use (12)-(14) to obtain the dual solution \( x^\nu_m(s,a,t) \leftarrow x^m_t(s,a), x^\nu_m(s,T+1) \leftarrow x^m_{T+1}(s), \forall m \in \mathcal{M} \).

15: Let \( z^\nu_m \) be the corresponding objective function value as in (15) and \( z^\nu \leftarrow \sum_{m \in \mathcal{M}} \lambda_m z^\nu_m \).

16: \textbf{if} \( z^\nu > \bar{z} \) then \hspace{1cm} \(\triangleright\) The policy described by \( \pi^\nu \) is better than the incumbent
17: Update the incumbent to be \((\pi^\nu, z^\nu)\).
18: \textbf{end if}

19: \textbf{if} \( \theta^\nu \leq z^\nu \) then \hspace{1cm} \(\triangleright\) The policy described by \( \pi^\nu \) is suboptimal
20: Fathom the current node. Return to Step 2.

21: \textbf{else} For each model such that \((\theta^\nu > z^\nu)\), add an optimality cut to (16):
\[
\theta_m \leq \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sum_{t \in T} (r^m_t(s,a) + M(1 - \pi_t(a|s))) x^\nu_m(s,a,t) + \sum_{s \in \mathcal{S}} r^m_{T+1}(s)x^\nu_m(s,T+1)
\]
22: Set \( k \leftarrow k + 1 \). Return to Step 3.
23: \textbf{end if}
MMDP PB-B&B procedure, and then subsequently describe search strategies including node selection, branching, and pruning within this framework.

4.2.1. General MMDP policy-based branch-and-bound framework The custom PB-B&B procedure is grounded in the idea that, if we relax the requirement that each of the $|M|$ MDPs must have the same policy, the problem becomes easy to solve as it involves solving $|M|$ independent MDPs. Thus, at the beginning of the custom PB-B&B procedure, we completely relax the requirement that each of the $|M|$ MDPs must have the same policy, and solve each individual model via backward induction to obtain an optimal policy for each model. Then, we sequentially add restrictions that the policies must prescribe the same action for certain $(s, t)$-pairs for each of the $|M|$ MDPs (for $(s, t) \in S \times T$). We refer to these restrictions as partial policies because they specify the actions to be taken in some, but not necessarily all, of the $(s, t)$-pairs. These partial policies are stored as nodes in a B&B tree with the root node corresponding to the empty partial policy (see Figure 1 for a small example. In the figure, each partial policy is represented above by a matrix where the component in row $s$ and column $t$ indicates which action to take in state $s$ and decision epoch $t$ if specified by the partial policy. Only 3 levels of the tree are shown.)

Given a partial policy, at a given node, one solves each MDP independently, taking the best action for each $(s, t)$-pair in that given model unless that $(s, t)$-pair’s action has already been fixed in the partial policy.

The custom PB-B&B procedure begins with an empty partial policy, meaning that none of the $(s, t)$-pairs are fixed to a specific action, and thus, each of the $|M|$ MDPs can be solved independently. For a given partial policy, $\hat{\pi}$, the corresponding relaxation of the MMDP is solved. The solution of the relaxation will be described in more detail in Section 4.2.4. Solving the relaxation provides an optimistic completion of the partial policy of a given node and an upper bound on the weighted value objective that could be achieved by completing this partial policy. The upper bound is compared to the value associated with the best solution seen so far, called the incumbent. If the upper bound is worse than the lower bound associated with the incumbent, the node is
Figure 1  A small example of a PB-B&B tree of partial policies for an MMDP with $|S|=2$, $|T|=3$, and $|A|=3$. Each node encodes a partial policy, which is represented above by a matrix where the component in row $s$ and column $t$ indicates which action to take in state $s$ and decision epoch $t$ if specified by the partial policy. Only three levels of the tree are shown.

pruned, meaning that none of its descendants in the tree will be examined. If the model-specific optimal completions of the partial policy are the same in each of the $|M|$ MDPs then the policy is an implementable policy for the MMDP and, if the corresponding weighted value is better than the incumbent, then the incumbent solution is replaced with this policy and the node is pruned by optimality. If the partial policy is not pruned, the node is added to the list of pending nodes. At each iteration a pending node (i.e. partial policy) is selected for branching.

**Proposition 4** (Worst-case running time of the custom PB-B&B procedure). The worst-case running time of the PB-B&B procedure is $O(|M|TS^2A^{ST+1})$ where $T=|T|$, $S=|S|$, and $A=|A|$.

**Proof.** PB-B&B algorithms have a worst-case running time of $O(Ub^d)$ where $b$ is the branching factor, $d$ is the depth of the tree, and $U$ is an upper bound on time required to solve a subproblem (Morrison et al. 2016). The branching factor of the PB-B&B tree described above is $A$ and the depth of the tree is $ST$. The worst-case running time to solve a subproblem is the time required to solve $M$ MDPs in the worst-case. The worst-case time to solve a single MDP is $O(TS^2A)$ (Puterman 1994, p. 93).

Interestingly the worst-case running time grows linearly in the number of models considered and exponentially in the number of states and epochs. This suggests that identifying actions than
Algorithm 2 Solve Relaxation for the MMDP

1: procedure SolveRelaxation(\( \hat{\pi} \)) \Comment{Solve MMDP relaxation with partial policy \( \hat{\pi} \)}
2: \( v^m_T(s) \leftarrow r^m_T(s), \forall s \in S, m \in M \)
3: \( t \leftarrow T \)
4: while \( t > 0 \) do \Comment{Iterate backwards through the decision epochs}
5: for \( (s, m) \in S \times M \) do
6: if \( \hat{\pi}_t(s) \) fixed then \Comment{Action for \( (s, t) \) is fixed by partial policy.}
7: \( v^m_t(s) \leftarrow r^m_t(s, \hat{\pi}_t(s)) + \sum_{s' \in S} p^m_t(s'|s, \hat{\pi}_t(s)) v^m_{t+1}(s') \)
8: else \Comment{Action for \( (s, t) \) is not fixed. Select the best action for this model.}
9: \( v^m_t(s) \leftarrow \max_{a \in A} \{ r^m_t(s, a) + \sum_{s' \in S} p^m_t(s'|s, a) v^m_{t+1}(s') \} \)
10: \( \pi^m(s, t, \hat{\pi}) \leftarrow \arg \max_{a \in A} \{ r^m_t(s, a) + \sum_{s' \in S} p^m_t(s'|s, a) v^m_{t+1}(s') \} \)
11: end if
12: end for
13: end while
14: \( v^m(\hat{\pi}) \leftarrow \sum_{s \in S} \mu^m_1(s) v^m_1(s), \forall m \in M \)
15: \( \bar{z}^i \leftarrow \sum_{m \in M} \lambda_m v^m(\hat{\pi}) \)
16: end procedure

cannot be optimal via an action elimination procedure (Puterman 1994, §6.7.2) or the presence of a special structure such as a monotone optimal policy (Puterman 1994, §6.11.2) could be used to reduce computation time.

4.2.2. Node selection strategy Standard options for the search strategy include breadth-first search (BrFS), depth-first search (DFS), and best-first search (BeFS). In DFS, the unexplored partial policies are explored in last-in-first-out fashion, such that there is a priority placed on finding implementable policies. In this search strategy, the list of partial policies is maintained using a stack structure wherein the algorithm explores the partial policy of the most recent child node first before generating any of its siblings. One advantage of this approach is that often there are
some value function estimates corresponding to a partial policy that can be reused by the children of that partial policy. In other types of searches, the reusable value function estimates would need to be stored while awaiting selection of the node, however in DFS this information can be removed from memory as soon as the policy is optimally completed. There can be drawbacks to DFS. For example, DFS can lead to imbalanced search trees in which some partial policies remain pending at small depths in the tree while the algorithm is focused on completing other partial policies, which could lead to the PB-B&B procedure spending a lot of time completing poor policies before finding an optimal policy.

Another search strategy is BrFS which can be viewed as a first-in-first-out approach to exploring partial policies, meaning that all children of a node are examined before any grandchildren can be examined. In this case, the unexplored partial policies roughly have the same number of \((s,t)\)-pairs fixed at each stage in the PB-B&B procedure. However, there are usually substantial memory requirements associated with BrFS because most children nodes are not explored immediately after their parent nodes. Another search strategy is the BeFS which considers the partial policies that appear most promising. In the MMDP, a best-bound approach explores a partial policy \(\hat{\pi}\) next if its corresponding upper bound \(\hat{z}\) is higher than all the other unexplored policies.

### 4.2.3. Branching strategy

At each iteration in which at least one pending node exists, the PB-B&B algorithm selects an \((s,t)\)-pair to branch on to generate new subproblems. In our PB-B&B procedure, we focus on wide branching strategies which generate \(|\mathcal{A}(s,t)|\) new partial policies to be investigated upon branching on a pair \((s,t)\), where each new subproblem corresponds to fixing \(\pi_t(s)\) to be each of the possible actions in \(\mathcal{A}(s,t)\), the action set specific to this \((s,t)\)-pair. In this context, there are several strategies for branching on \((s,t)\)-pairs.

One such branching strategy is horizon-based branching. In this strategy, decisions for branching are made on the basis of the decision epoch. That is, there is some ordering of the \((s,t)\)-pairs such that \(t < t'\) implies something about the order in which \(\pi_t(s)\) and \(\pi_{t'}(s')\) are fixed for any states \(s\) and \(s'\). One such approach is early-horizon branching in which the decisions for epochs early in
the planning horizon are fixed first. Early-horizon branching may be desirable because this allows
the branch-and-bound procedure to reuse value function estimates from the wait-and-see problem
wherein each of the |M| MDPs is solved independently.

Another branching strategy is disagreement branching, which is in the vein of most fractional
branching in integer programming. The idea behind the disagreement branching strategy is to
select the parts of the optimal completion of the partial policy found in the relaxation, and select
the (s,t)-pair for which there is the largest disagreement among the models’ optimal completion
policies. For instance, one disagreement-based strategy is to select the (s,t)-pair for which there is
the largest number of actions suggested by the different models.

4.2.4. Pruning strategy A natural pruning procedure in this context is to eliminate partial
policies based on their corresponding upper bounds. If the upper bound on the weighted value
corresponding to the completion of partial policy \( \hat{\pi} \) is lower than the weighted value associated with
the incumbent solution, then the PB-B&B procedure no longer needs to consider partial policy \( \hat{\pi} \)
as there is no way to complete the policy such that it will be better than the incumbent. We restrict
our attention to this pruning strategy because the upper bound associated with a partial policy
is easily obtained by solving the relaxation via Algorithm 2. When the partial policy is empty,
this relaxation corresponds to solving the wait-and-see problem. For any other partial policy such
that some (s,t)-pairs have been fixed, this procedure uses backward induction to find the optimal
completion of the partial policy. In the appendix, we discuss how the pruning procedure is easily
modified to reflect other multi-model formulations of this problem.

5. Computational study

5.1. Test instances

To test the performance of the proposed algorithms, we present an MDP related to machine
maintenance adapted from Delage and Mannor (2009). The states of the MDP represent the quality
of the machine. Over the course of the planning horizon, the DM chooses between doing nothing
and different levels of repairs with the goal of minimizing expected total cost. We generated test
instances of this MDP of four different problem sizes in terms of the number of states, actions, and decision epochs, and used a Dirichlet distribution to sample the transition probabilities. For each problem size, we varied the number of models and the concentration parameter of the Dirichlet distribution which controls how concentrated the parameters from the different models are to their mean value.

5.2. Experiments

First, we consider the WVP. We solve the WVP for the machine maintenance MMDP instances using the MIP EF, the B&C, and the custom PB-B&B algorithms presented in Section 4. We implemented the MIP EF and the B&C procedure presented in Algorithm 1 using the commercial solver Gurobi Optimizer Version 7.5.1 in C++. We implemented the EF using default settings in Gurobi and used special ordered set (SOS) Type 1 constraints for both the EF and B&C implementations. In the B&C procedure, the optimality cuts were added as lazy constraints within user callbacks whenever an integer feasible solution was found. We implemented the custom PB-B&B procedure in C++ using a priority queue data structure to manage the search tree and a custom node class to manage the information stored at each node in the tree. We specified an optimality gap of 1% for each of the algorithms, specified a time limit of 300 seconds, and set the number of threads for Gurobi to be 1. For each solution method, we provided the mean value problem (MVP) solution as a warm-start in Gurobi for the EF and B&C, and as the incumbent for the PB-B&B procedure. We also solve each instance using the custom PB-B&B procedure for the various multi-model objectives. We report the solution time required to solve the MMDP formulations and compare the resulting solutions. All experiments were run on a Windows Server 2012 R2 Standard with a 3.4 GHz Intel processor and 128 GB of RAM.

5.3. Results

We now present the results of our experiments. First, we describe the performance of the solution methods on the machine maintenance test instances. Then, we discuss the properties of the solutions obtained by solving these MMDP instances.
5.3.1. **Comparison of the solution methods**  Our initial experiments with the branching and node selection strategies for the custom PB-B&B algorithm suggested that “early-horizon” branching and BeFS node selection are the most effective search strategies. Early-horizon outperformed disagreement-based branching in large part due to the ability to reuse value function information from the parent nodes when early-horizon branching is used. We tested the node selection strategy using random MMDP test instances and solved each test instance using each of the three node selection strategies with early-horizon branching. Our initial experimentation showed that BeFS search requires the least amount of computation time to solve these instances, followed by DFS, and finally BrFS. BeFS explored the fewest number of nodes of all node selection strategies followed by DFS; BrFS explored the most nodes.

We now compare the MIP-based solution methods and the custom PB-B&B approach using the search strategy described above. Tables 1 and 2 demonstrate the solution times and optimality gaps for the machine maintenance problem for each solution method. For the base case problem size, we observe that the PB-B&B procedure was able to solve each of the 120 instances within the time limit whereas the EF and B&C were only able to solve 9 of these problem instances. The instances that were solved by EF and B&C terminated immediately because the value of the MVP was within the tolerance of the upper bound at the root node. In contrast, the PB-B&B algorithm was able to solve each instance within the time limit. We observed that the PB-B&B outperformed the EF and B&C algorithms on a variety of problem sizes and characteristics. Out of the 48 problem types, the PB-B&B algorithm performed better than EF and B&C in terms of average optimality gap and worst-case optimality gap on all but one problem type (see \((|\mathcal{S}|, |\mathcal{A}|, |\mathcal{T}|, |\mathcal{M}|, \alpha) = (4, 4, 8, 20, 0.1)\)) where the EF had a better worst-case optimality gap. In total, the PB-B&B algorithm was able to solve 409 out of the 480 instances while the EF and B&C solution methods were only able to solve 50 of the instances within the time limit. These findings demonstrate that across many instances of MMDPs with different characteristics in terms of variation in parameters, the PB-B&B procedure outperforms the MIP-based procedures.
Table 1  Computational performance of the extensive form, branch-and-cut, and policy-based branch-and-bound procedures on the machine maintenance MMDP as the problem size, number of models, and concentration parameter in the Dirichlet distribution are varied. For each problem type, 10 test instances were generated and solved to within 1% of optimality. Computation time is reported in CPU seconds and a maximum time limit of 300 seconds was enforced.

| $(|S|, |A|, |T|)$ | $|M|$ | $\alpha$ | Solution Time (CPU Seconds) | Optimality Gap (%) |
|---|---|---|---|---|
| | | | EF | B&C | PB-B&B | EF | B&C | PB-B&B |
| | 5 | 0.1 | > 300 | > 300 | > 300 | > 300 | 0.1 | 0.3 | 17.5 | 24.7 | 23.4 | 30.2 | < 1 | < 1 |
| | | 1 | > 300 | > 300 | > 300 | > 300 | 0.2 | 0.4 | 17.4 | 25.1 | 20.1 | 28.2 | < 1 | < 1 |
| | | 10 | > 300 | > 300 | > 300 | > 300 | < 0.1 | 0.1 | 7.7 | 12.7 | 8.1 | 12.7 | < 1 | < 1 |
| | | 100 | 180.1 | > 300 | 180.1 | > 300 | < 0.1 | < 0.1 | 1.2 | 3.1 | 1.2 | 3.1 | < 1 | < 1 |
| (4,4,4) | 10 | 0.1 | > 300 | > 300 | > 300 | > 300 | 0.3 | 0.7 | 24.6 | 32.7 | 32.2 | 38.5 | < 1 | < 1 |
| | | 1 | > 300 | > 300 | > 300 | > 300 | 0.4 | 0.9 | 19.2 | 26.7 | 22.0 | 33.0 | < 1 | < 1 |
| | | 10 | > 300 | > 300 | > 300 | > 300 | 0.1 | 0.2 | 8.7 | 12.9 | 9.0 | 13.4 | < 1 | < 1 |
| | | 100 | 210.1 | > 300 | 210.1 | > 300 | < 0.1 | < 0.1 | 1.6 | 2.6 | 1.6 | 2.6 | < 1 | < 1 |
| | 20 | 0.1 | > 300 | > 300 | > 300 | > 300 | 0.5 | 0.7 | 26.0 | 31.8 | 34.5 | 40.5 | < 1 | < 1 |
| | | 1 | > 300 | > 300 | > 300 | > 300 | 1.0 | 1.6 | 23.9 | 28.6 | 27.1 | 32.1 | < 1 | < 1 |
| | | 10 | > 300 | > 300 | > 300 | > 300 | 0.4 | 0.9 | 10.6 | 12.8 | 10.9 | 13.2 | < 1 | < 1 |
| | | 100 | 240.1 | > 300 | 240.1 | > 300 | < 0.1 | < 0.1 | 1.4 | 2.8 | 1.4 | 2.8 | < 1 | < 1 |
| (5,5,5) | 5 | 0.1 | > 300 | > 300 | > 300 | > 300 | 89.8 | > 300 | 11.0 | 14.0 | 17.2 | 27.8 | 1.4 | 12.5 |
| | | 1 | > 300 | > 300 | > 300 | > 300 | 65.9 | > 300 | 9.0 | 11.9 | 11.4 | 17.8 | < 1 | 1.7 |
| | | 10 | > 300 | > 300 | > 300 | > 300 | 0.3 | 1.1 | 4.2 | 7.0 | 4.4 | 7.3 | < 1 | < 1 |
| | | 100 | 30.1 | > 300 | 30.1 | > 300 | < 0.1 | < 0.1 | < 1 | 1.2 | < 1 | 1.2 | < 1 | < 1 |
| | 10 | 0.1 | > 300 | > 300 | > 300 | > 300 | 169.0 | > 300 | 13.5 | 17.5 | 18.7 | 27.3 | 1.5 | 5.6 |
| | | 1 | > 300 | > 300 | > 300 | > 300 | 212.6 | > 300 | 11.9 | 14.6 | 14.5 | 17.8 | 3.3 | 7.8 |
| | | 10 | > 300 | > 300 | > 300 | > 300 | 0.8 | 6.0 | 4.8 | 6.1 | 5.1 | 6.3 | < 1 | < 1 |
| | | 100 | 30.2 | > 300 | 30.0 | > 300 | < 0.1 | < 0.1 | < 1 | 1.9 | < 1 | 2.0 | < 1 | < 1 |
| (8,4,4) | 10 | 0.1 | > 300 | > 300 | > 300 | > 300 | 254.9 | > 300 | 16.5 | 19.2 | 22.3 | 26.5 | 8.0 | 13.3 |
| | | 1 | > 300 | > 300 | > 300 | > 300 | 218.5 | > 300 | 14.1 | 16.7 | 17.7 | 21.2 | 3.9 | 9.1 |
| | | 10 | > 300 | > 300 | > 300 | > 300 | 0.8 | 3.4 | 4.8 | 6.1 | 5.0 | 6.6 | < 1 | < 1 |
| | | 100 | 60.4 | > 300 | 60.1 | < 300 | < 0.1 | < 0.1 | < 1 | 1.5 | < 1 | 1.5 | < 1 | < 1 |

Tables 1 and 2 also allow us to determine how problem characteristics such as problem size and variance affects computation time. We also observe that the time to solve the MMDP instances
Table 2  Computational performance of the extensive form, branch-and-cut, and policy-based branch-and-bound procedures on the machine maintenance MMDP as the problem size, number of models, and concentration parameter in the Dirichlet distribution are varied. For each problem type, 10 test instances were generated and solved to within 1% of optimality. Computation time is reported in CPU seconds and a maximum time limit of 300 seconds was enforced.

| ((|S|,|A|,|T|), |M|, α) | Solution Time (CPU Seconds) | Optimality Gap (%) |
|----------------|--------------------------|---------------------|
|                | EF | B&C | PB-B&B | EF | B&C | PB-B&B |
| (4,8,4) 5 | 0.1 | > 300 | > 300 | > 300 | > 300 | 1.6 | 9.4 | 14.9 | 26.8 | 21.1 | 31.1 | <1 | <1 |
|           | 1 | > 300 | > 300 | > 300 | > 300 | 3.8 | 12.4 | 17.1 | 25.7 | 19.5 | 28.9 | <1 | <1 |
|           | 10 | > 300 | > 300 | > 300 | > 300 | 0.3 | 1.8 | 7.1 | 10.4 | 7.3 | 11.4 | <1 | <1 |
|           | 100 | 60.1 | > 300 | 60.0 | > 300 | <0.1 | <0.1 | <1 | 1.9 | <1 | 1.9 | <1 | <1 |
| (4,8,4) 10 | 0.1 | > 300 | > 300 | > 300 | > 300 | 10.1 | 26.6 | 25.5 | 29.2 | 33.0 | 43.4 | <1 | <1 |
|           | 1 | > 300 | > 300 | > 300 | > 300 | 5.1 | 20.2 | 19.6 | 24.7 | 22.3 | 30.5 | <1 | <1 |
|           | 10 | > 300 | > 300 | > 300 | > 300 | 0.7 | 2.6 | 10.2 | 15.1 | 10.5 | 16.4 | <1 | <1 |
|           | 100 | 270.1 | > 300 | 270.1 | > 300 | <0.1 | <0.1 | 1.4 | 2.4 | 1.4 | 2.4 | <1 | <1 |
| (4,4,8) 20 | 0.1 | > 300 | > 300 | > 300 | > 300 | 23.6 | 42.1 | 27.8 | 31.3 | 33.0 | 37.1 | <1 | <1 |
|           | 1 | > 300 | > 300 | > 300 | > 300 | 47.8 | 104.7 | 24.5 | 27.2 | 27.9 | 31.2 | <1 | <1 |
|           | 10 | > 300 | > 300 | > 300 | > 300 | 6.1 | 18.7 | 11.6 | 13.3 | 12.0 | 13.5 | <1 | <1 |
|           | 100 | 240.1 | > 300 | 240.1 | > 300 | <0.1 | <0.1 | 1.5 | 2.5 | 1.5 | 2.5 | <1 | <1 |
| (4,4,8) 5 | 0.1 | > 300 | > 300 | > 300 | > 300 | 83.6 | 231.3 | 19.6 | 41.1 | 30.4 | 46.8 | 1.9 | 17.2 |
|           | 1 | > 300 | > 300 | > 300 | > 300 | 314.5 | >300 | 18.0 | 30.9 | 23.7 | 40.0 | <1 | 3.4 |
|           | 10 | > 300 | > 300 | > 300 | > 300 | 62.2 | >300 | 9.2 | 13.4 | 9.8 | 14.3 | <1 | 1.8 |
|           | 100 | 240.1 | > 300 | 240.1 | > 300 | <0.1 | <0.1 | 1.7 | 3.9 | 1.7 | 3.9 | <1 | <1 |
| (4,4,8) 10 | 0.1 | > 300 | > 300 | > 300 | > 300 | 213.1 | >300 | 24.1 | 31.5 | 33.2 | 42.4 | 6.5 | 19.3 |
|           | 1 | > 300 | > 300 | > 300 | > 300 | 271.2 | >300 | 23.0 | 31.9 | 27.5 | 34.1 | 10.2 | 19.6 |
|           | 10 | > 300 | > 300 | > 300 | > 300 | 69.7 | >300 | 10.8 | 17.3 | 11.4 | 17.3 | 1.1 | 2.7 |
|           | 100 | 270.1 | > 300 | 270.1 | > 300 | <0.1 | 0.1 | 1.9 | 3.3 | 1.9 | 3.3 | <1 | <1 |
| (4,4,8) 20 | 0.1 | > 300 | > 300 | > 300 | > 300 | 250.5 | >300 | 27.2 | 35.3 | 39.4 | 47.6 | 14.6 | 40.5 |
|           | 1 | > 300 | > 300 | > 300 | > 300 | 281.1 | >300 | 22.6 | 27.4 | 26.7 | 33.9 | 10.9 | 21.4 |
|           | 10 | > 300 | > 300 | > 300 | > 300 | 189.8 | >300 | 11.4 | 16.6 | 11.7 | 17.9 | 2.5 | 7.8 |
|           | 100 | 270.2 | > 300 | 270.2 | > 300 | <0.1 | 0.1 | 1.7 | 3.1 | 1.7 | 3.0 | <1 | <1 |

for each algorithm tends to be at its highest when the transition probabilities are sampled from a Dirichlet distribution with concentration parameters 0.1 and 1 and that instances with lower
Figure 2 demonstrates how the variance among the models’ parameters can play an important role in the value of solving the WVP relative to the simpler MVP and the value of perfect information. Figure 2(a) shows the VSS for four values of the concentration parameter used in the Dirichlet distribution. The problem size was 4 states, 4 actions, and 4 decision epochs and the number of models was varied to be 5, 10, or 20. Each line represents the average value across 10 instances of each size. All instances were solved to within 1% of optimality.
Figure 3 The objective function values for several values of the concentration parameter used in the sampling of the Dirichlet distribution. The problem size was 4 states, 4 actions, and 4 decision epochs with 20 models. Each line represents the average value across 30 instances of this size.

distribution. We observe that for higher values of concentration parameters (i.e., 10 and 100), the VSS is quite small in all cases, but the VSS can be quite high for cases where there is high variance among the models. This finding suggests that when the different models’ have parameters that are similar to each other the MVP is a suitable approximation and that the PB-B&B algorithm is spending its time proving the optimality of the MVP solution. However, as the variance among the models increases, the VSS tends to increase. Figure 2(b) demonstrates how variance among the model parameters can influence expected value of perfect information (EVPI). We observe that EVPI tends to be higher than VSS and that EVPI is also decreasing as the models’ parameters become more closely concentrated. We also observe that EVPI can still be considerable even for larger concentration parameters.

Figure 3 demonstrates how the objective function value changes with the concentration parameter used in the Dirichlet distribution. Each point corresponds to one of 30 instances with 4 states, 4 actions, and 4 decision epochs. The lines represents the average value across these 30 instances for each objective function. As we might expect, we find that the value of the optimal MVP policy
is quite robust to the concentration parameter. However, surprisingly, we found that the percentile optimization approach also was relatively robust to changes in the concentration parameter. The max-min, min-max-regret, and weighted value approaches all were sensitive to the concentration parameter to some extent. We observe that the optimal cost for the WVP is increasing with the concentration parameter while the max-min model is decreasing with the concentration parameter. It appears that the percentile optimization, weighted value, and max-min objective functions start to converge as the concentration parameter increases and that the maximum regret among the models decreases as the models’ parameter are more closely concentrated. These findings suggest that the DM’s preference towards ambiguity is less important when the models are quite similar, and more important in cases when the models’ parameters are quite different.

In summary, the findings from the case study are as follows. We tested the EF, B&C, and PB-B&B procedures on several instances of a machine maintenance problem of different sizes and characteristics and found that the PB-B&B approach outperforms the MIP-based approaches. We explored the impact of variance among model parameters on VSS and EVPI and found that these metrics tend to increase with variance among the multiple models’ parameters. When the variance is low among the models’ parameters, the MVP solution is near-optimal in many cases. These findings present general guidelines for practitioners to know if there will be a large value in solving the WVP rather than the MVP. For problems where the variance is low, the VSS tends to be low and the PB-B&B algorithm spends its computation time proving optimality of the MVP rather than generating new incumbent solutions. Third, the PB-B&B is easily modified to incorporate other objective functions for handling multiple models in MDPs. We demonstrate that the optimal value functions will depend on the DM’s objective and that, as the variance among the model parameters increases, the choice of the DM’s objective function becomes more important.

6. Conclusion

In this article, we addressed the problem of solving MMDPs exactly. The MMDP has been proposed as a method for designing policies that account for ambiguity in the input data for MDPs, but
the solution of MMDPs had been restricted to very small instances. Finding an optimal Markov
deterministic policy for an MMDP is an NP-hard problem, and the EF of the problem is a MIP
that includes “big-M”s which weaken the linear programming relaxation.

We proposed two decomposition methods that leverage the problem structure to solve the
MMDP. The first was a B&C algorithm in the vein of the Integer L-Shaped Method for two-stage
stochastic integer programming with binary first-stage variables. The B&C algorithm decomposed
the EF of the MIP into a master problem involving the binary variables used to encode the policy
and $|M|$ subproblems that evaluate a proposed policy in each model of the MDP. Unfortunately,
the EF relied on the notorious “big-M”s to enforce logical constraints; the big-Ms led to weak
optimality cuts to be added within the B&C procedure. We also proposed a custom PB-B&B pro-
cedure which does not require big-Ms in the formulation. The custom PB-B&B procedure begins
by viewing the MMDP as $|M|$ independent MDPs. The algorithm began by allowing each model
of the MDP to have its own policy and sequentially added requirements that the decision rules for
certain state-time pairs agree in each model.

We presented the first numerical comparison of MIP-based solution methods and other algo-
 rhythmic approaches for solving finite-horizon MDPs with multiple models. To do so, we generated
random MMDP instances of a machine maintenance problem to compare the computation time
required to solve these problems using the extensive form (extensive form), the B&C procedure,
and the custom PB-B&B procedure. Our computational experiments showed that the custom PB-
B&B solution method greatly outperforms the solution of the MIP using the EF directly and with
the B&C procedure across a variety of problem sizes and problem characteristics. We found that
the custom PB-B&B solution time increased significantly with the number of decision epochs and
the number of states. We also observed that the differences between the models plays an important
role in how long the required computation time for the solution methods where these methods
tended to perform better on test instances where the models were most closely concentrated.

Due to the ability to solve larger MMDPs than before, the PB-B&B solution method enabled
us to investigate trends between problem characteristics and the impact of ambiguity. Using the
machine maintenance case study, we considered the impact of the variance among the models used in the MMDP on the value of the MMDP approach in terms of value relative to the mean value problem and expected regret relative to each model’s optimal policy. We found that the MVP may be a suitable approximation to the WVP, especially when there is a large number of models and a high concentration of the models’ parameters around their mean. We found that the value in solving the weighted value problem tended to be higher when there were fewer models that were quite different. We also found that EVPI tends to be higher than VSS. We also used the machine maintenance instances to compare alternative ways to incorporate multiple models into the solution of the MDP. We found that the DM’s preference towards ambiguity is most important when there is more discrepancy between the models’ parameters. Thus, the DM’s attitude towards ambiguity may be important when solving MMDPs in these cases.

Our study has limitations. First, our B&C procedure does not include logic-based optimality cuts which could potentially improve its performance. However, our initial experiments with such cuts showed they do not provide significant enhancements to the B&C procedure. Second, we describe properties of the solution, such as the expected value of perfect information and the value of the MMDP, for a particular machine maintenance problem; these values and solution properties may depend on this problem’s particular transition probability and reward structure and not be representative of all MDPs with ambiguity in the transition probabilities. Third, we do not specifically tailor our custom PB-B&B procedure to the other multi-model objective. Our results suggest that developing better heuristics and further refining the custom PB-B&B for these other objectives could lead to computational gains.

Given that MMDPs are NP-hard, no exact algorithms will scale with respect to increases in model size. Even solution methods for single-model MDPs struggle as the state-space grows due to the well-known curse of dimensionality, and we expect that this problem would be exacerbated in the MMDP framework because it is NP-hard and incorporates multiple models. As we showed, there are some cases where the value of solving MMDPs is low. An important future direction
for research is to develop better lower and upper bounds on the VSS to provide the DM with information as to whether or not solving the WVP for a particular MMDP instance is worth the increase in computational complexity. There are also future opportunities to enhance the PB-B&B procedure to reduce the computation time for solving these multi-model formulations. For example, there may be opportunities to explore the use of action elimination procedures (see Puterman (1994, §6.7)) in the setting of multiple model MDPs to enhance the PB-B&B algorithm. The benefits of reducing the action space would be two-fold. First, we could potentially eliminate children nodes corresponding to non-optimal actions which could reduce the number of nodes considered. Second, the backward induction procedure at each node would not consider the non-optimal actions leading to a reduction in the computation time at each node. Recent research on approximate dynamic programming methods also raises interesting opportunities for how such methods might be integrated into the MMDP context. Methods such as using state aggregation to reduce the number of states could be easily used with our approach because such methods preserve the structure exploited by our PB-B&B algorithm. However, other popular methods such as finite basis function approximations would need special consideration for how to exploit these methods in the context of the PB-B&B algorithm wherein each model must operate under the same policy (see Chapter 6 of Powell (2007) for a discussion of such approaches). This would likely require a significantly different algorithmic approach than we propose, but it could be an important direction for future research. The work in this article provides the means to evaluate future approximate dynamic programming methods applied to MDPs with multiple models by comparing the results to optimal policies for moderate size MDP instances which are solvable using our method.

Our approach could be extended in several ways. First, we consider only finite-horizon MMDPs, but our decomposition algorithms might be extended to the infinite-horizon setting. For instance, the custom PB-B&B method might be extended to infinite-horizon MMDPs by using linear programming, value iteration, or policy iteration to solve the relaxations at each node in the PB-B&B tree. It would be interesting to compare this approach to those proposed in Buchholz and
Scheftelowitsch (2018). Second, we could further investigate other branching and node selection strategies to enhance the custom PB-B&B procedure. Third, we do not assume any structure on the original MDP or its optimal policy. However, if we were to assume structure, such as monotonicity of the optimal policy, our algorithms might be able to be modified to exploit this structure for computational gain. Finally, it would be interesting to compare these model-based approaches to more traditional robust MDP formulations in which an ambiguity set is constructed around the models although constructing an appropriate ambiguity set for fair comparisons would require special consideration. The work presented in this article provides a foundation for exploring these important future directions.

Acknowledgments

This work was supported by the National Science Foundation under grant numbers 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.

References


Appendix. Other multi-model formulations

In this appendix, we provide MIPs for the multi-model formulations discussed in Section 3.3.

A. Maximize worst-case

The max-min MMDP as presented in (6) can be formulated as the following MIP:

\[
\begin{align*}
\max_{\pi, v, w} & \quad w \\
\text{s.t.} & \quad \sum_{a \in A} \pi_t(a|s) = 1, \quad \forall s \in S, t \in T, \quad (17a) \\
& \quad M \pi_t(a|s) + v_t^m(s) - \sum_{s' \in S} p_{s|a}^m(s') v_{t+1}^m(s') \leq r_{t}^m(s,a) + M, \quad \forall m \in M, s \in S, a \in A, t \in T, \quad (17b) \\
& \quad v_{T+1}^m(s) \leq r_{T+1}^m(s), \forall m \in M, s \in S, \quad (17c) \\
& \quad w - \sum_{s \in S} \mu_t^m(s) v_t^m(s) \leq 0, \quad \forall m \in M, \quad (17d) \\
& \quad \pi_t(a|s) \in \{0, 1\}, \quad \forall a \in A, s \in S, t \in T, \quad (17e) \\
& \quad v_t^m(s) \in \mathbb{R}, \quad \forall m \in M, s \in S, t \in T, \quad (17f) \\
& \quad w \in \mathbb{R}. \quad (17g)
\end{align*}
\]

The formulation in (17) represents the policy through the binary variables \(\pi\) and the value functions for each model, state, and decision epoch through the continuous variables \(v\). The continuous variable \(w\) represents the worst-case model value and takes on the appropriate value due to constraint (17e).

However, rather than solving the MIP directly, one can solve the problem using a modification of the PB-B&B algorithm described. To do so, one needs to modify the bounding approach to reflect the best possible worst-case model value that comes from completing a partial policy. The upper bound on the worst-case value is determined by selecting the value for the model with the smallest upper bound found by solving the relaxation. That is, Step 15 in Algorithm 2 (presented in Section 4.2), should be modified to

\[
\bar{z} \leftarrow \min_{m \in M} v^m(\tilde{\pi}) \quad (18)
\]

to reflect the appropriate upper bound.

B. Minimize maximum regret

Ahmed et al. (2017) proposed a similar MIP-based solution method for solving the min-max-regret MMDP. In the MIP formulation below, we define \(\pi \in \{0, 1\}^{|S| \times |A| \times |T|}\) and \(v \in \mathbb{R}^{|M| \times |S| \times |A| \times |T|}\) as
above representing the policy and value function variables of the MMDP, respectively. In the
min-max-regret formulation, we introduce continuous variables $\bar{v}$ to represent the optimal value
functions for each model. Notice that these continuous variables are independent of the policy
variables $\pi$. The constraints in (19e) and (19f) ensure that the variables $\bar{v}^m$ take on the optimal value
functions corresponding to model $m \in \mathcal{M}$. We also introduce a continuous variable $w$ to represent
the worst-case regret among all of the models. Constraints (19g) serve as epigraph constraints
ensuring that $w$ will take on the value of the largest regret among all of the models. To minimize
the value of $w$, the variables $v^m$ will attempt to take on values as close to their $\bar{v}^m$ counterparts,
so long as the constraints in (19c) and (19d), which enforce that these value functions correspond
to policy $\pi$, are satisfied.

\[
\begin{align*}
\min_{\pi, v, w, \bar{v}} & \quad w \\
\text{s.t.} & \quad \sum_{a \in \mathcal{A}} \pi_t(a|s) = 1, \quad \forall s \in \mathcal{S}, t \in \mathcal{T}, \quad (19b) \\
& \quad M \pi_t(a|s) + v^m_t(s) - \sum_{s' \in \mathcal{S}} \mathbb{P}^m_t(s'|s, a)v^m_{t+1}(s') \leq r^m_t(s, a) + M, \quad \forall m \in \mathcal{M}, s \in \mathcal{S}, a \in \mathcal{A}, t \in \mathcal{T}, \quad (19c) \\
& \quad \bar{v}^m_t(s) - \sum_{s' \in \mathcal{S}} \mathbb{P}^m_t(s'|s, a)\bar{v}^m_{t+1}(s') \geq r^m_t(s, a), \quad \forall m \in \mathcal{M}, s \in \mathcal{S}, a \in \mathcal{A}, t \in \mathcal{T}, \quad (19e) \\
& \quad \sum_{s \in \mathcal{S}} \mu^m_t(s)\bar{v}^m_t(s) - \sum_{s \in \mathcal{S}} \mu^m_t(s)v^m_t(s) - w \leq 0, \quad \forall m \in \mathcal{M}, \quad (19g) \\
& \quad \pi_t(a|s) \in \{0, 1\}, \quad \forall a \in \mathcal{A}, s \in \mathcal{S}, t \in \mathcal{T}, \quad (19h) \\
& \quad v^m_t(s), \bar{v}^m_t(s) \in \mathbb{R}, \quad \forall m \in \mathcal{M}, s \in \mathcal{S}, t \in \mathcal{T}, \quad (19i) \\
& \quad w \in \mathbb{R}. \quad (19j)
\end{align*}
\]

Rather than solving the MIP (19) directly, the PB-B&B algorithm can be modified to apply to
this problem. To do so, the PB-B&B requires a modification to reflect that this is a minimization
problem. Further, one needs to modify the bounding approach to reflect the best possible worst-
case model regret that comes from completing a partial policy. The bound is easily found after
solving the relaxation, which is done in the same manner as before. However, at a given node, the lower bound needs to reflect the worst-case regret by determining which model’s value in the relaxation is furthest from the corresponding model’s optimal value. To reflect this modification to the lower bound, Step 15 in Algorithm 2, should be modified to

$$\bar{z} \leftarrow \min_{m \in \mathcal{M}} \left( \max_{\pi \in \Pi} \{ v^m(\pi) \} - v^m(\hat{\pi}) \right)$$

(20)

Then, one can remove nodes from consideration in the PB-B&B procedure by pruning any node whose lower bound is greater than the incumbent solution in the minimization problem.

C. Percentile optimization

Meraklı and Küçükyavuz (2019) proposed a similar MIP-based formulation for solving the PercOpt MMDP. In the following MIP formulation, we define \( \pi \) and \( v \) as above representing the policy and value function variables, respectively. In the PercOpt formulation, we 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. Constraint (21e) ensures that all models with values greater than or equal to \( z \) will have corresponding \( y_m \) variables equal to 1 and all models with a value less than \( z \) will have a \( y_m \) value of 0. Constraint (21f) ensures that there is no more than \( \epsilon \) chance that the models have values less than \( z \).

$$\max_{\pi, v, y, z} z$$

s.t.

$$\sum_{a \in \mathcal{A}} \pi_t(a|s) = 1, \quad \forall s \in \mathcal{S}, t \in \mathcal{T},$$

(21b)

$$M \pi_t(a|s) + v^m_t(s) - \sum_{s' \in \mathcal{S}} p^m_t(s'|s,a)v^m_{t+1}(s') \leq r^m_t(s,a) + M, \quad \forall m \in \mathcal{M}, s \in \mathcal{S}, a \in \mathcal{A}, t \in \mathcal{T},$$

(21c)

$$v^m_{T+1}(s) \leq r^m_{T+1}(s), \forall m \in \mathcal{M}, s \in \mathcal{S},$$

(21d)

$$z + My_m - \sum_{s \in \mathcal{S}} \mu^m_t(s)v^m_t(s) \leq M, \quad \forall m \in \mathcal{M}, s \in \mathcal{S},$$

(21e)

$$\sum_{m \in \mathcal{M}} \lambda_m y_m \geq 1 - \epsilon \quad \forall m \in \mathcal{M},$$

(21f)

$$\pi_t(a|s) \in \{0, 1\}, \quad \forall a \in \mathcal{A}, s \in \mathcal{S}, t \in \mathcal{T},$$

(21g)

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

(21h)

$$v^m_t(s) \in \mathbb{R}, \quad \forall m \in \mathcal{M}, s \in \mathcal{S}, t \in \mathcal{T},$$

(21i)

$$z \in \mathbb{R}.$$
Similar to the cases above, rather than solving the MIP directly, one can solve the problem using a modification of the PB-B&B algorithm. To do so, the upper bound on the $\epsilon$-percentile is determined by selecting the value for the model with the smallest upper bound found by solving the relaxation. Step 15 in Algorithm 2 should be modified to reflect the appropriate upper bound. The upper bound $\bar{z}$ is easily found by sorting the models in increasing order by the value functions $v^m(\hat{\pi})$ to obtain an order statistic $m_{(1)}, \ldots, m_{(|M|)}$. Then, one can select $\bar{z}$ to be $v^{m(x)}(\hat{\pi})$ where $x$ is the smallest value such that $\sum_{m=m_{(1)}}^{m_{(x-1)}} \lambda_m \leq \epsilon$. 