Using Mathematical Programming to Solve Factored Markov Decision Processes with Imprecise Probabilities

Karina Valdivia Delgado[a], Leliane Nunes de Barros[b], Fabio Gagliardi Cozman[c], Scott Sanner[d]

[a]*Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, SP, Brazil*
[b]*Instituto de Matemática e Estatística, Universidade de São Paulo, SP, Brazil*
[c]*Escola Politécnica, Universidade de São Paulo, SP, Brazil*
[d]*NICTA and the Australian National University, Canberra, ACT 2601, Australia*

Abstract

This paper investigates Factored Markov Decision Processes with Imprecise Probabilities (MDPIPs); that is, Factored Markov Decision Processes (MDPs) where transition probabilities are imprecisely specified. We derive efficient approximate solutions for Factored MDPIPs based on mathematical programming. To do this, we extend previous linear programming approaches for linear approximations in Factored MDPs, resulting in a multilinear formulation for robust "maximin" linear approximations in Factored MDPIPs. By exploiting the factored structure in MDPIPs we are able to demonstrate orders of magnitude reduction in solution time over standard exact non-factored approaches, in exchange for relatively low approximation errors, on a difficult class of benchmark problems with millions of states.

Key words: Imprecise Markov Decision Processes (MDPIPs), Probabilistic Planning, Multilinear programming

1. Introduction

Sequential decision making is an essential activity in many domains, ranging from operations research [1] and planning [2] to robotics [3]. Markov Decision Processes (MDPs) provide an elegant mathematical framework for representing and solving sequential decision problems under uncertainty in completely observable environments. An MDP encodes the interaction between an agent and its environment: at every stage, the agent decides to execute an action (with probabilistic effects) that takes it to a next state and yields a reward. The agent's goal is to maximize the expected reward over a sequence of actions.

Traditionally, MDPs assume a "flat" (enumerated) representation of states. A more compact representation for MDPs uses a state representation factored into multiple state variables — the so-called Factored MDPs. Such representations suffer from Bellman's curse of dimensionality [4]: the size of the state

space grows exponentially in the number of state variables. Recent approximate solutions for Factored MDPs exploit the factored representation [5, 6, 7] so as to solve problems that are orders of magnitude larger than those solvable using classical flat dynamic programming approaches.

Despite their elegance and generality, Markov Decision Processes are often inadequate to represent real-world problems. In many problems, it is simply impossible to obtain precise values for all transition probabilities. This may occur for many reasons, including: (a) imprecise or conflicting transition probabilities elicited from experts, (b) insufficient data to estimate precise transition models, or (c) abstraction of parts of the model that are too complex to detail, for instance by omitting variables that cannot be properly measured.

The seminal work by Satia and Lave Jr. [8] studied several optimality criteria for Markov Decision Processes with Imprecise Probabilities (MDPIPs); that is, MDPs where transition probabilities may be imprecisely specified. Satia and Lave Jr. paid significant attention to a maximin criterion: an agent must choose an action that maximizes the lowest possible future expected reward (that is, "Nature" is assumed to select transition probabilities in an adversarial manner). While there have been proposals for exact and approximate solutions to MDPIPs using this maximin criterion [8, 9, 10], these approaches can only solve relatively small problems. These algorithms face the difficulty that, to deal with imprecisely specified probabilities, they have to solve *at least* one nonlinear programming problem *per state*.

The present article makes a number of fundamental contributions to state-of-the-art solution methods for large MDPIPs:

- We introduce a definition of Factored MDPIPs and a factored specification of imprecision in transition probabilities (based on dynamic credal networks (DCNs) [11]). To the best of the authors' knowledge, no previous work has investigated Factored MDPIPs.

- We provide algorithms for approximate solution of Factored MDPIPs based on mathematical programming, by extending previous work [10, 12]. Specifically, we first give a bilevel programming formulation for the approximate solution of Factored MDPIPs, and then transform it into multilinear programming, a well known formalism for which many practical solvers with strong convergence guarantees exist.

- We extend previous work [12] so as to obtain efficient approximate linear programming solutions for Factored MDPs. We exploit the Factored MDPIP structure to reduce the number of constraints generated and to compactly encode the remaining constraints that empirically leads to an exponential reduction in the number of constraints for some problems.

Section 2 reviews basic concepts on MDPs and Factored MDPs, and the basic theory of "flat" MDPIPs. Section 3 defines Factored MDPIPs and gives bilevel and multilinear programming formulations. Section 4 presents an algorithm, called FactoredMPA (Factored Multilinear programming-based approximation), that produces maximin policies by resorting to approximate nonlinear

programming. We demonstrate orders of magnitude reduction in solution time over standard exact non-factored approaches to MDPIPs in exchange for relatively low approximation errors on a difficult class of benchmark problems with millions of states.

2. Background

In this section we review basic concepts on MDPs and Factored MDPs. We also define MDPIPs given two flat (i.e., non factored) formulations: i) based on bilevel programming and ii) based on multilinear programming.

2.1. Flat Markov Decision Processes

In this paper a *Markov Decision Process* is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, P, \gamma \rangle$, where [1, 13]:

- $\mathcal{S}$ is a finite set of fully observable states;
- $\mathcal{A}$ is a finite set of actions;
- T is a countable set of stages starting at stage 0;
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a fixed reward function associated with every state and action;
- $P(s'|s, a)$ is the conditional probability of reaching state s' at stage $t + 1$ when in state $s \in \mathcal{S}$ at stage t, given action $a \in \mathcal{A}$ is taken at t;
- $\gamma \in (0, 1)$ is a *discount factor* (the reward obtained t stages into the future is discounted in the sense that it is multiplied by γ^t).

A policy returns an action in each state, at any stage. A stationary policy $\pi : S \to \mathcal{A}$ returns an action $\pi(s)$ in state s (regardless of stage). The value of a stationary policy π, starting in state s_0 at stage 0, and progressing with an infinite horizon ($|T = \infty|$), is here taken to be the following expected value, known as the *value function*:

$$V_\pi(s) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s \right], \tag{1}$$

where R_t is the reward obtained at stage $t \in T$. Equation (1) can be decomposed and rewritten recursively based on the values of the possible successor states $s' \in \mathcal{S}$ as follows:

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V_\pi(s').$$

The goal is to find a policy π^* that yields the maximal value in each state: $\forall s, \pi' \ : \ V_{\pi^*}(s) \geq V_{\pi'}(s)$. For infinite horizon with discounted cost there always

exists such an *optimal stationary* policy. The *optimal value function* associated with an optimal policy, represented by V^*, can be computed by the Bellman equation [14]:

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right\}. \quad (2)$$

The Bellman equation can be solved through a linear program [15]:

$$\begin{aligned} \min_{V^*} \quad & : \quad \sum_s V^*(s) \quad (3) \\ s.t. \quad & : \quad V^*(s) \geq R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s'), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned}$$

To see that linear program (3) produces the unique fixed-point solution of Equation (2), note that constraints force $V^*(s)$ to be greater than or equal to $\max_a \{R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s')\}$, considering all $a \in \mathcal{A}$, and then minimizing $\sum_s V^*(s)$ to enforce that the minimal $V^*(s)$ is obtained.

2.2. Factored Markov Decision Processes

In a *Factored MDP*, states $s \in \mathcal{S}$ are represented by a set $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ of *state variables*: a state s is represented as a tuple $\mathbf{x} = (x_1, x_2, ..., x_n)$ where x_i is the value of the state variable X_i. The cardinality of $\mathcal{S}$ is exponential in the number n of state variables.[1] In a Factored MDP, the reward function $R(\mathbf{x}, a)$ is usually defined by the sum of ψ local-reward functions $R_j(\mathbf{x}, a)$:

$$R(\mathbf{x}, a) = \sum_{j=1}^{\psi} R_j(\mathbf{x}, a). \quad (4)$$

The scope of each local-reward function $R_j(\cdot, a)$ is typically restricted to a subset of the variables $\mathbf{X} = \{X_1, ..., X_n\}$.

The next step is to encode transition probabilities using *Dynamic Bayesian Networks* (DBNs) [18]. That is, we employ a directed acyclic graph with two layers for each action: one layer represents the variables in the current state and the other layer represents the next state (Figure 1.a). Nodes X_i and X_i' refer to the current and next state respectively. Directed edges are allowed *from* nodes in the first layer *into* the second layer, and also between nodes in the second layer. We denote by $\text{pa}(X_i')$ the parents of X_i' in the graph. The graph is assumed endowed with the following Markov condition: a variable X_i' is conditionally independent of its nondescendants given its parents. This leads to the following factorization of transition probabilities:

$$P(\mathbf{x}'|\mathbf{x}, a) = \prod_{i=1}^{n} P(x_i'|\text{pa}(X_i'), a), \quad (5)$$

[1] Finding an optimal policy is a P-Complete problem for flat MDPs [16, 17]; factored MDPs introduce an exponential increase in the size of the state space.

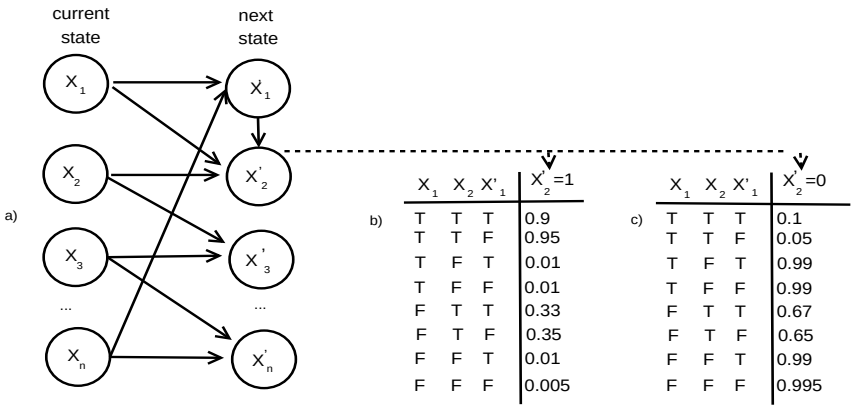

Figure 1: a) A Dynamic Bayesian Network (DBN) for an action a; b) conditional probability table for $X_2' = 1$; c) conditional probability table for $X_2' = 0$.

where $\text{pa}(X_i')$ may be fixed by $\mathbf{x}$. That is, the probability of moving to $\mathbf{x}' \in \mathcal{S}$, given the agent is in state $\mathbf{x} \in \mathcal{S}$ and executes the action $a \in \mathcal{A}$, is the product of conditional probabilities for $\{X_i' = x_i'\}$ given the configuration of parents of X_i' and the action $a \in \mathcal{A}$.

Figure 1.b shows the conditional probability table (CPT) for $X_2' = 1$; Figure 1.c shows the CPT for $X_2' = 0$. The tables include all combinations of variables values only for the parents of X_2', i.e., $\text{pa}(X_2')$ and the sum of each line in Figure 1.b and Figure 1.c is 1.

Recent results have shown that it is possible to solve a Factored MDP with billions of states [5, 17]. The technique of Approximate Linear Programming (ALP) [19] has emerged as one of the most promising methods for solving complex Factored MDPs. The basic idea is to solve an MDP, formulated as Problem (3), by approximating the optimal value function through basis functions that are provided by domain experts or automatically generated [17, 20, 21]. The quality of the approximation depends on the set of basis functions.

The approximate value function is denoted by $\widehat{V}(\mathbf{x})$. Given $\mathbf{x} \in \mathcal{S}$ and a set of basis functions $H = \{h_1, ..., h_k\}$, $V^*(\mathbf{x})$ is approximated using a linear combination:

$$\widehat{V}(\mathbf{x}) = \sum_{j=1}^{k} w_j h_j(\mathbf{x}). \tag{6}$$

The ALP formulation of an MDP, given Expressions (3), (4) and (6), is given

by the linear program:

$$\begin{aligned} \min_{w} \quad &: \quad \sum_{\mathbf{x}} \sum_{i=0}^{k} w_i h_i(\mathbf{x}) \qquad (7) \\ \text{s.t.} \quad &: \quad \sum_{i=0}^{k} w_i h_i(\mathbf{x}) \geq \sum_{j=1}^{\psi} R_j(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}' \in \mathcal{S}} P(\mathbf{x}'|\mathbf{x}, a) \sum_{i=0}^{k} w_i h_i(\mathbf{x}'), \\ & \qquad \forall \mathbf{x} \in \mathcal{S}, a \in \mathcal{A}. \end{aligned}$$

In order to guarantee that this linear program is feasible, a constant basis function must be included. We denote the constant function by h_0, that has the same constant value for all states. Hence, the sum $\sum_i$ starts from 0 instead of 1.

The number of variables in Expression (7) should be made smaller than $|\mathcal{S}|$ by selecting a relatively small number of basis functions. Note however that the number of constraints is the same as in Expression (3). In Section 4 we discuss how to obtain computational gains in ALP by exploiting the factored structure of the problem.

2.3. Markov Decision Processes with Imprecise Probabilities

An MDPIP is a sequential decision process endowed with state space, actions, rewards and discount factor as any MDP, but where transition probabilities can be imprecisely specified. For instance, perhaps the tightest bounds on the probability $P(s_2|s_1, a_1)$ are just $P(s_2|s_1, a_1) \in [1/3, 1/2]$. That is, instead of a probability measure $P(\cdot|s, a)$ over the state space $\mathcal{S}$, we have a *set* of probability measures for a fixed pair state/action. We use the term *credal set* to refer to a set of probability measures (or a set of distributions for a random variable) [22]. A set of distributions for a variable X is denoted by $K(X)$. We adopt elementwise conditioning: $K(X|B)$ is obtained from $K(X)$ by conditioning every distribution in the credal set $K(X)$ on the event B (using Bayes rule). A credal set containing conditional distributions over the state space, given a state s and an action a, is referred to as a *transition credal set* [22] and denoted by $K(s'|s, a)$.

We assume that all credal sets are closed and convex, an assumption that is often used in the literature, and that encompasses most of the practical applications [23, 24, 25]. We further assume stationarity for the transition credal sets $K(s'|s, a)$; that is, they do not depend on the stage t. Note that a probability distribution for a complete history of the process (that is, a sequence of states) may be non-stationary: distributions $P(s'|s, a)$ may be selected from the corresponding credal sets in a time dependent manner, from stage to stage [26].

There are several criteria of choice regarding policies in MDPIPs. The maximin criterion selects the policy that maximizes reward gained under the assumption that Nature minimizes reward; that is, the policy that yields the supremum of lower expected reward. The maximin criterion is sometimes referred to Γ-maximin [23], to differentiate it from the maximin criterion used in

frequentist decision making [27]. Several other criteria of choice can be found in the literature. For instance, the "maximax" criterion [8] selects a policy that yields the supremum of upper expected reward [28], while the "maximix" criterion selects a policy that yields the maximum of $\alpha(\max_P V_\pi)+(1-\alpha)(\min_P V_\pi)$, for some $\alpha \in (0,1)$. In this paper we adopt the maximin criterion throughout, as it is the most prevalent criterion and it offers a reasonable approach when robust policies are sought for. For a critical appraisal of the maximin criterion, including an analysis of incoherence in sequential decision making, the reader may consult the analysis by Seidenfeld [29].

There is always a deterministic stationary policy that is maximin for expected reward in a discounted infinite horizon [8]; moreover, this policy induces a value function that is the unique solution of the equation

$$V^*(s) = \max_a \min_P \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^*(s') \right). \tag{8}$$

There exist algorithms for solving this equation based on dynamic programming [8, 9]. In particular, value iteration is a straightforward implementation of Equation (8) using dynamic programming techniques. Although value iteration is a general solution, it is a very inefficient solution that can solve only small problems. Only a few special cases of MDPIPs do admit efficient solution schemes [30, 31].

It does not seem possible to reduce the solution of Equation (8) to linear programming (similar for instance to Problem 3). In our previous work [10] we have shown that it is possible to solve Equation (8) by resorting to *bilevel* and *multilinear* programming. First, Equation (8) can be reduced to bilevel programming:

$$\begin{aligned}
\min_{V^*} \quad &: \quad \sum_s V^*(s) \qquad (9)\\
\text{subject to} \quad &: \quad V^*(s) \geq R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s'), \forall s \in \mathcal{S}, a \in \mathcal{A}\\
&\quad\; P(s'|s,a) = \arg\min_Q \sum_{s' \in \mathcal{S}} Q(s'|s,a) V^*(s')\\
&\quad\; \text{subject to :} \quad Q(s'|s,a) \in K_a(s'|s).
\end{aligned}$$

This bilevel program can be transformed to an equivalent multilinear program [10]:

$$\begin{aligned}
\min_{V^*,P} \quad &: \quad \sum_s V^*(s) \qquad (10)\\
\text{s.t.} \quad &: \quad V^*(s) \geq R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s'), \forall s \in \mathcal{S}, a \in \mathcal{A}\\
&\quad\; P(s'|s,a) \in K_a(s'|s).
\end{aligned}$$

Lemma 1. *Problem (9) and Problem (10) produce the optimal value function,* $V^*(s)$.

Proof: To verify that Problem (9) produces $V^*(s)$, i.e, that Problem (9) finds the unique fixed-point solution of Equation (8), $V^*(s)$, we use the constraints to force $V^*(s)$ to be greater than or equal to:

$$\max_a \left\{ R(s,a) + \gamma \min_P \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right\},$$

and then minimizing $\sum_s V^*(s)$ to enforce that the minimal $V^*(s)$ (at equality) is obtained. Because Problems (9) and (10) are equivalent from [10], we can find $V^*(s)$ by solving the multilinear program in Expression (10). ■

Note that the solution of multilinear programs is far from trivial; only relatively small flat MDPIPs can be tackled directly through Expression (10) [10].

3. Defining and Representing Factored MDPIPs

A *Factored MDPIP* is essentially a Factored MDP where transition probabilities are not unique. We propose *Dynamic Credal Networks* (DCNs) as the adequate language to express factored transition credal sets. A DCN has the same structure as a DBN (Figure 1); however, each variable X_i is associated with credal sets $K_a(X_i|\text{pa}(X_i) = \pi_k)$ for each value π_k of $\text{pa}(X_i)$. We assume that a DCN represents a joint credal set over all of its variables consisting of all distributions that satisfy the factorization

$$P(\mathbf{x}'|\mathbf{x}, a) = \prod_{i=1}^{n} P(x'_i|\text{pa}(X'_i), a), \tag{11}$$

where each $P(x'_i|\text{pa}(X'_i), a)$ comes from an appropriate credal set associated with the DCN. This sort of joint credal set is called the *strong extension* of the credal network in the literature [11, 32]. The entries in the CPT are specified by parameters p_{ijk}, for $\{X'_i = x'_{ij}\}$ given $\{\text{pa}(X'_i) = \pi_k\}$; note that each p_{ijk} may be free rather than a fixed number. Figure 2 shows a Dynamic Credal Network for action $a_1 \in \mathcal{A}$, the CPT for state variable X'_1 that depends on X_1 and the constraints over the parameters p_{ijk}, where $i = 1$ is the index of X'_1, $j \in \{1, 2\}$ indicates the possible values of X'_1 and k the possible values of $\text{pa}(X'_1)$. Note that $K_a(X'_i|\text{pa}(X'_i))$ are defined, indirectly, by a set of constraints $\mathfrak{C}$ over the probability values p_{ijk} (inequations in Figure 2).

3.1. Bilevel and Multilinear Approximate Formulations

To derive maximin policies for Factored MDPIPs, we begin by the bilevel formulation (Problem (9)). The *factored value function* of a Factored MDPIP is given by taking Equation (6) and restricting the scope of each basis function to some subset of state variables $\aleph_i \subset \mathbf{X} = \{X_1, ..., X_n\}$. We can then insert this new factored value function plus the reward function (4) and the transition probabilities (11) into Problem (9) to obtain:

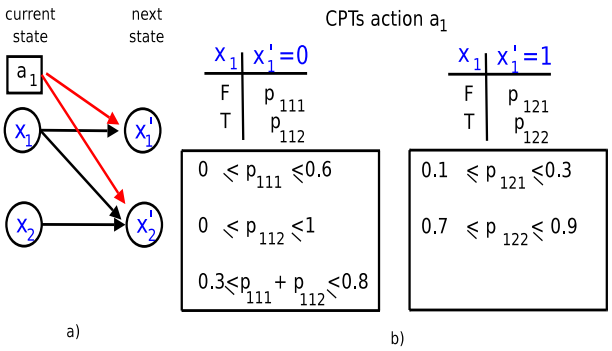

Figure 2: a) Dynamic Credal Network for action $a_1 \in \mathcal{A}$. b) The (symbolic) conditional probability table for the state variable X_1' and the constraints on probability values.

$$
\begin{aligned}
\min_{w} \quad & \sum_{\mathbf{x}} \sum_{i=0}^{k} w_i h_i(\mathbf{x}) \qquad (12)\\
\text{subject to} \quad : \quad & \sum_{i=0}^{k} w_i h_i(\mathbf{x}) \geq R(\mathbf{x}, a) + \\
& \gamma \sum_{\mathbf{x}' \in \mathcal{S}} P(\mathbf{x}'|\mathbf{x}, a) \sum_{i=0}^{k} w_i h_i(\mathbf{x}'), \forall \mathbf{x} \in \mathcal{S}, a \in \mathcal{A} \\
& P(\mathbf{x}'|\mathbf{x}, a) = \arg\min_{Q} \sum_{\mathbf{x}' \in \mathcal{S}} Q(\mathbf{x}'|\mathbf{x}, a) \sum_{i=0}^{k} w_i h_i(\mathbf{x}') \\
& \text{where } Q(\mathbf{x}'|\mathbf{x}, a) = \prod_{i} Q(x_i'|pa(X_i'), a) \\
& \text{subject to:} \\
& \quad Q(x_i'|pa(X_i'), a) \in K_a(X_i'|pa(X_i')).
\end{aligned}
$$

which is the bilevel formulation of a Factored MDPIP. In this formulation the first level minimizes w and the second level minimizes Q. Some characteristics of Problem (12) are worth mentioning:

- there are $|\mathcal{S}| * |\mathcal{A}|$ constraints at the first level;
- the constraints in the first level are non-linear, since the weights w_i are multiplied by $P(\mathbf{x}'|\mathbf{x}, a)$;
- the objective in the second level is non-linear whenever the basis functions are based on more than one variable;
- the first level and the second level share the same free optimization variables (i.e., the probability values).

Thus, Problem (12) is not a trivial bilevel problem, and most existing methods to solve bilevel problems do not apply; for instance, there are obstacles to applying the following methods: *K-th Best* [33], *Branch-and-bound* [34], *Trust-Region* [35], *Inexact Restoration* [36] and *Steepest Descent Direction* [37]. (For a more detailed discussion on the difficulties faced by these methods, see the paper by Delgado et al [38].)

We can also use the factored value function and replace it in the multilinear formulation (Problem (10)) of an MDPIP so as to obtain the factored multilinear programming problem:

$$\begin{aligned}
\min_{w,P} \quad & \sum_{\mathbf{x}} \sum_{i=0}^{k} w_i h_i(\mathbf{x}) \qquad (13)\\
\text{subject to} \quad : \quad & \sum_{i=0}^{k} w_i h_i(\mathbf{x}) \geq R(\mathbf{x}, a) + \\
& \gamma \sum_{\mathbf{x}' \in \mathcal{S}} P(\mathbf{x}'|\mathbf{x}, a) \sum_{i=0}^{k} w_i h_i(\mathbf{x}'), \forall \mathbf{x} \in \mathcal{S}, a \in \mathcal{A} \\
& \text{where } P(\mathbf{x}'|\mathbf{x}, a) = \prod_i P(x'_i|pa(X'_i), a) \\
& P(x'_i|pa(X'_i), a) \in K_a(X'_i|pa(X'_i)).
\end{aligned}$$

In Problem (13), if the set of basis functions is such that $V^*(\mathbf{x}) = \sum_{j=1}^{k} w_j h_j(\mathbf{x})$, we say the basis functions attain the exact solution.

Theorem 1. *If the set of chosen basis functions attains the exact solution, the exact solution is obtained by solving Problem (13).*

Proof: By replacing V^* in Problem (13), we obtain Problem (10) and by Lemma 1 we guarantee that solving problem (13) we will find the exact solution: $V^*(s)$. ■

Notice that in Problem (13), even though we can efficiently compute the coefficients of the objective function and the constraints, we are still working with the complete set of constraints ($|\mathcal{S}| * |\mathcal{A}| + m_2$), where m_2 is the number of constraints in $\mathfrak{C}$ related to the probabilities p_{ijk} (used to define the credal sets K_a). The direct use of general non-linear solvers [39, 40], geometric solvers [41] or multilinear solvers [42] for Factored-MDPIPs, can only solve problems of type (13) with small state space.

Local Optimization Solution. We can use local optimization algorithms [43] to solve the Multilinear Problem (13). Such an algorithm divides the variables in groups and, at each iteration, fixes the values of each group to obtain a linear problem. A Factored-MDPIP formulated as a multilinear programming problem can be solved by this local optimization algorithm if and only if the basis functions have scope restricted to one state variable, because we can then divide the variables in two groups (for instance we can define one group with

the w_i variables for $i = 1$ to k and the other group with w_0) and the variables related with the probabilities. Whenever basis functions have more than one variable, it is not possible to divide the variables in order to obtain a linear problem at each iteration, so the algorithm [43] cannot be applied.

Reducing the Number of Constraints. We can also try to reduce the number of constraints in the problem and call a nonlinear solver only once. This idea is the same one that has been used to efficiently solve Factored MDPs [5].

Given the mentioned difficulties in solving an MDPIP formulated as a bilevel problem (Problem (12)), in this paper we explore solutions to the multilinear Problem (13) by applying techniques to reduce the number of constraints. In the next section we present the main contribution of this paper: an algorithm for the generation of maximin policies in Factored MDPIPs that solves Problem (13) by reducing the number of constraints, so as to tackle large state spaces.

4. An Efficient Solution for Factored MDPIPs: FactoredMPA

We wish to solve Factored MDPIPs by exploiting ideas that have been successfully applied to MDPs; namely, by pursuing analogues of Approximate Linear Programming (ALP). There has been significant evidence [44] that if one is interested in Factored MDPs, and thus interested in solving Problem (7), two conditions must be fulfilled so as to apply ALP successfully. First, it is necessary to restrict the scope of each basis function to some small subset of state variables. Second, it is necessary to assume a relatively sparse set of dependencies in the DBNs that encode the factorization of probabilities. Guestrin [5] has demonstrated that these conditions are fulfilled in a variety of applications, and has exploited these conditions to develop efficient algorithms for Factored MDPs.

The success of Guestrin's FACTOREDLPA algorithm to solve MDPs is related with the set of constraints of the approximate linear program (Problem (7)): (i) the exploitation of factored structure to avoid the generation of unnecessarily complex constraints and (ii) the generation of compact sets of constraints.[2] Our goal is to solve Factored MDPIPs (Problem (13)) by developing some form of approximate multilinear programming. Even though the same techniques used in connection with Guestrin's algorithm [17, 45, 46] cannot be applied directly to the multilinear problem, the FACTOREDLPA algorithm can be generalized to a FACTOREDMPA algorithm to solve factored MDPIPs, as we show next.

To solve a Factored MDPIP we have to simplify the set of constraints in Problem (13) by applying the same ideas used by Guestrin in his *FactoredLPA* algorithm: it creates a new and smaller equivalent set of constraints for each action in the Linear Programming Problem (7) before calling a linear solver with the new minimization problem. This is done in two steps: (i)*a simplification step* and (ii) *a contraction step*.

[2]There are variants of these techniques in the literature, exploiting various schemes from linear programming [17, 45, 46].

4.1. Simplification Step: Exploiting the Factored Structure of an MDPIP

Let an MDPIP be defined by $\langle \mathcal{S}, \mathcal{A}, T, R, K, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, T is a countable set of stages, R is a reward function associated with every state and action and is defined by local-reward functions R_j, $K(s'|s,a)$ is the transition credal set defined by DCNs and probability constraints $\mathfrak{C}$ and $\gamma \in (0,1)$ is a *discount factor*. The computing of constraints in Problem (13) must benefit from the fact that our MDPIP is factored and that the basis functions H have a restricted scope. So, the purpose of this step is to precompute some values to simplify the constraints. Take Problem (13); we have the following set of constraints:

$$\sum_i w_i h_i(\mathbf{x}) \geq \sum_{j=1}^{\psi} R_j(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}' \in \mathcal{S}} P(\mathbf{x}'|\mathbf{x}, a) \sum_i w_i h_i(\mathbf{x}'), \forall \mathbf{x} \in \mathcal{S}, a \in \mathcal{A}. \quad (14)$$

We can rearrange terms as follows:

$$\sum_i w_i h_i(\mathbf{x}) \geq \sum_{j=1}^{\psi} R_j(\mathbf{x}, a) + \gamma \sum_i w_i g_i^a(\mathbf{x}, \mathbf{p}), \quad (15)$$

where

$$g_i^a(\mathbf{x}, \mathbf{p}) = \sum_{\mathbf{x}' \in \mathcal{S}} P(\mathbf{x}'|\mathbf{x}, a) h_i(\mathbf{x}'); \quad (16)$$

we use $\mathbf{p}$ to denote a vector containing all probability values that are free to vary within the given credal sets (i.e., that satisfy the probability constraints $\mathfrak{C}$ in the DCN). That is, $\mathbf{p}$ contains all probability values that define the distributions we seek.

Koller and Parr [44] showed that if h_i has scope restricted to a subset of state variables $\aleph_i \subset \mathbf{X} = \{X_1, ..., X_n\}$, then g_i^a has scope restricted to the parents of $\aleph_i$ in the DBN of action a (these parents are denoted by Γ).

Intuitively, in $\sum_{\mathbf{x}' \in \mathcal{S}} P(\mathbf{x}'|\mathbf{x}, a) h_i(\mathbf{x}')$ we can push the sum over variables $x'_j \notin \Gamma$ inwards to obtain:

$$\sum_{x'_i \in \Gamma} P(x'_i|\mathbf{x}, a) h_i(\mathbf{x}') \sum_{x'_j \notin \Gamma} P(x'_j|\mathbf{x}, a).$$

As the inwards sum adds up to 1, we have:

$$\sum_{x'_i \in \Gamma} P(x'_i|\mathbf{x}, a) h_i(\mathbf{x}').$$

For MDPIPs, $g_i^a(\mathbf{x}, \mathbf{p})$ is a polynomial expression and has scope restricted to the parents of $\aleph_i$ in the DCN. That is, it is described in terms of probability values and has the canonical form $d_0 + \sum_i d_i \prod_j p_{ijk}$, where d_0 and d_i are constants and p_{ijk} are parameters. Thus, to avoid repeated calculations, it is only necessary to

calculate g_i^a for each assignment z of Γ. For further computational improvement, the set of constraints can be rewritten as:

$$0 \geq \sum_{j=1}^{\psi} R_j(\mathbf{x}, a) + \sum_i w_i c_i^a(\mathbf{x}, \mathbf{p}), \tag{17}$$

where:

$$c_i^a(\mathbf{x}, \mathbf{p}) = \gamma g_i^a(\mathbf{x}, \mathbf{p}) - h_i(\mathbf{x}). \tag{18}$$

This latter term can be precomputed resulting also in a polynomial form and has scope restricted to $\Theta = \aleph_i \cup \Gamma$. Finally, we obtain:

$$0 \geq \sum_{j=1}^{\psi} R_j(\mathbf{x}, a) + \sum_i w_i c_i^a(\mathbf{x}, \mathbf{p}), \forall \mathbf{x} \in \mathcal{S}, a \in \mathcal{A}. \tag{19}$$

Algorithm 1 named COMPUTEc_i^a has as input an MDPIP given by $\langle \mathcal{S}, \mathcal{A}, T, R, K, \gamma \rangle$ and a set of basis functions H and returns the set of functions $\mathcal{C}^a$. COMPUTEc_i^a computes g_i^a and c_i^a for each basis function h_i as in Expressions (16) and (18) respectively, note that the term g_i^a has scope restricted to the parents of $\aleph_i$ in the DCN (denoted by Γ) and c_i^a has scope restricted to $\Theta = \aleph_i \cup \Gamma$.

Although precomputing the expression c_i^a helps to simplify the constraints, note that we still have the complete set of constraints (that is, $|\mathcal{S}| * |\mathcal{A}| + m_2$). Because general non-linear solvers applied to Problem (13) can only solve problems with small state space, we must further reduce the number of constraints (Section 4.2).

4.2. Contraction Step: Generating a Compact Set of Constraints

Guestrin [5] generates a compact set of *linear* constraints to efficiently solve Factored MDPs; we can also generate a compact set of *multilinear* constraints to solve Factored MDPIPs (Problem (13)). The basic idea is first to replace the set of constraints in Expression (19) by an equivalent set of non-linear constraints (maximizing over $\mathbf{x}$), as follows:

$$0 \geq \max_{\mathbf{x}} \left\{ \sum_{j=1}^{\psi} R_j(\mathbf{x}, a) + \sum_i w_i c_i^a(\mathbf{x}, \mathbf{p}) \right\}, \forall a \in \mathcal{A}. \tag{20}$$

So, for each action a, we have to satisfy

$$0 \geq \max_{\mathbf{x}} \left\{ \sum_{j=1}^{\psi} R_j^a(\mathbf{x}) + \sum_i w_i c_i^a(\mathbf{x}, \mathbf{p}) \right\}. \tag{21}$$

where $R_j^a(\mathbf{x})$ stands for $R_j(\mathbf{x}, a)$. Now we show how to transform the constraint given by the Inequation (21) into a set of simpler constraints. Since in Expression (21) we have to solve the maximization over the complete state space, its computation is too expensive. Instead of adding up all terms and performing

Algorithm 1: COMPUTE$c_i^a(MDPIP, H, a)$

input : $MDPIP$ (given by $\langle \mathcal{S}, \mathcal{A}, T, R, K, \gamma \rangle$),
H (set of basis functions h_i),
a (action)
output: $\mathcal{C}^a$ (set of functions c_i^a, with $1 \leq i \leq |H|$)

begin
  //k is the number of basis functions;
  $k = |H|$;
  for $i = 1$ *to* k **do**
    //compute g_i^a
    $\aleph_i$ = scope of h_i;
    Γ=parents of $\aleph_i$ in the DCN;
    foreach *assignment* z *of* Γ **do**
      calculate $g_i^a(z, \mathbf{p})$ //Equation (16)
    //compute c_i^a using g_i^a
    $\Theta = \aleph_i \cup \Gamma$;
    foreach *assignment* θ *of* Θ **do**
      $c_i^a(\theta, \mathbf{p}) = \gamma g_i^a(\theta, \mathbf{p}) - h_i(\theta)$;
  return $\mathcal{C}^a$;
end

this maximization over all states in $\mathcal{S}$, we can maximize over state variables one at a time. To do so, we modify the version of the variable elimination algorithm proposed by Guestrin [5].

For instance, suppose that we want to perform the maximization over variable $X_1 \in \mathbf{x}$. In order to eliminate this variable, we do as follows: if R_1^a is the only local-reward function for action a that depends on X_1; c_1^a is a function that depends, for instance, on X_1 and X_4 and there is no other function c_i^a that depends on X_1 in the Inequation (21); then we can push the maximization over X_1 inwards the sum to obtain:

$$0 \geq \max_{X_2 \ldots X_n} \left\{ \sum_{j=2}^{\psi} R_j^a(\mathbf{x}) + w_0 c_0^a + \sum_{i=2} w_i c_i^a(\mathbf{x}, \mathbf{p}) + \boxed{\max_{X_1} \left\{ R_1^a(X_1) + w_1 c_1^a(X_1, X_4, \mathbf{p}) \right\}} \right\}. \tag{22}$$

Let $R_1^a(X_1)$ and $w_1 c_1^a(X_1, X_4, \mathbf{p})$ in Expression (22), in our hypothetical example, be called *relevant function* to the variable X_1 (since in the example R_1^a is the only local-reward function that depends on X_1 and the same for the function c_1^a). In general, let be a relevant function, a function whose scope contains a variable X_l that we want to eliminate. In the above example, the relevant functions, renamed as $u_{Z_1}^{f_1}, \cdots, u_{Z_L}^{f_L}$, are $u_{X1}^{f_1} = R_1^a(X_1)$ and $u_{X1,X4}^{f_2} = w_1 c_1^a(X_1, X_4, \mathbf{p})$ (we call these expressions as *equality constraints*). Note that Z_i is the scope of each relevant function.

So, for each variable X_i we want to eliminate (under some order criterion O), *FactoredMP* selects L relevant functions. Then we can replace the maximization over these relevant functions by:

$$u_Z^{e_r} = \max_{X_l} \sum_{j=1}^{L} u_{Z_i}^{f_j}, \tag{23}$$

where Z is the union of all variables in functions $u_{Z_1}^{f_1}, \cdots, u_{Z_L}^{f_L}$ minus X_l, since after performing max, the new function is independent of X_l.

Back to our example (Expression (22)), the term $u_Z^{e_r}$ given by the maximization in the box where $Z = \{X_1\} \cup \{X_1, X_4\} \backslash \{X_1\}$, is renamed with relevant functions, as follows:

$$u_{X_4}^{e_r} = \max_{X_1} \left\{ u_{X_1}^{f_1} + u_{X_1,X_4}^{f_2} \right\}, \tag{24}$$

resulting in the following expression:

$$0 \geq \max_{X_2 \ldots X_n} \left\{ \sum_{j=2}^{\psi} R_j^a(\mathbf{x}) + w_0 c_0^a + \sum_{i=2} w_i c_i^a(\mathbf{x}, \mathbf{p}) + u_{X_4}^{e_r} \right\}.$$

Note that, in our example, to eliminate X_1 means to simplify the computation of Expression (22) once there are few relevant functions for X_1. But to enforce the definition of $u_{x_4}^{e_r}$ as in Expression (24) we must introduce four new inequality constraints, one for each combination of the configuration of X_4 and X_1, i.e.:

$$\begin{aligned}
u_{x_4}^{e_r} &\geq u_{x_1}^{f_1} + u_{x_1,x_4}^{f_2}. \\
u_{\overline{x_4}}^{e_r} &\geq u_{x_1}^{f_1} + u_{x_1,\overline{x_4}}^{f_2}. \\
u_{x_4}^{e_r} &\geq u_{\overline{x_1}}^{f_1} + u_{\overline{x_1},x_4}^{f_2}. \\
u_{\overline{x_4}}^{e_r} &\geq u_{\overline{x_1}}^{f_1} + u_{\overline{x_1},\overline{x_4}}^{f_2}.
\end{aligned}$$

In the general case, to enforce the definition of $u_Z^{e_r}$ as the maximum over X_l (Expression (23)), the *FactoredMP* algorithm introduces a new constraint for each assignment z of Z; i.e.:

$$u_z^{e_r} \geq \sum_{j=1}^{L} u_{z,x_l}^{f_j} \ \forall x_l. \tag{25}$$

This procedure is repeated until all variables have been eliminated. At the end, all the remaining functions u^{e_i} have an empty scope and the following inequality constraint must be added:

$$0 \geq \sum_i u^{e_i}. \tag{26}$$

that is, at the end of this variable elimination process, we transform the original set of inequality constraints (Expression (21)) to a new set of constraints

(Expression (25) and (26)) plus the equality constraints for renamed functions, which corresponds to a smaller set of simpler constraints (i.e., without maximization).

FACTOREDMP (Algorithm 2) implements the contraction step by creating a smaller equivalent set of constraints $\mathbb{J}$ for **one action** $a \in \mathcal{A}$, as we just described. It has as input the set of functions $\mathcal{C}^a$, the set of local-reward R^a, an order criterion O and the MDPIP. The algorithm starts with $\mathbb{F} = \{\}$ and $\mathbb{J} = \{\}$, where $\mathbb{F}$ is the set of new functions that will be created during the process (such as the renamed relevant functions and the new functions created in Expression (25)) and $\mathbb{J} = \{\}$ is the set of new constraints, initially empty, that will be of two types: equality and inequality constraints. FACTOREDMP calls the following functions in order to generate the set of *equality constraints*:

- GENERATEEQUALITYCONSFORREWARD (Algorithm 3) generates a set of equality constraints, $u_Z^{e_r} = R_j^a(Z)$, for each assignment z of Z(scope) of each local-reward R_j^a and
- GENERATEEQUALITYCONSFORc_i (Algorithm 4) generates an equality constraint $u^{e_r} = c_0^a * w_0$ and a set of multilinear equality constraints, $u_Z^{e_r} = c_i^a(Z, \mathbf{p}) * w_i$, for each assignment z of Z(scope) of each c_i^a function.

Finally, it generates a set of inequality constraints calling the function GENERATEINEQUALITYCONS (Algorithm 5), which eliminates the state variables one by one. These variables must be ordered by some criterion O (for instance, by removing first the variables that produce the smallest functions). For each eliminated variable X_l, the relevant functions $u_{Z_1}^{f_1}, \cdots, u_{Z_L}^{f_L}$ are selected and for each assignment z of Z a new inequality constraint, $u_z^{e_r} \geq \sum_{j=1}^{L} u_{z,x_l}^{f_j} \; \forall x_l$, is added to $\mathbb{J}$. Finally an inequality constraint with the sum over of all empty scope functions, $0 \geq \sum_i u^{e_i}$ (Expression (26)), is added and the new set of constraints $\mathbb{J}$ is returned.

4.3. The FactoredMPA *Algorithm*

FactoredMP must be applied for all actions $a \in \mathcal{A}$. This is done by the main algorithm FACTOREDMPA (Algorithm 6) that solves Problem (13) by generating a new (smaller) set of constraints **for all action** $a \in \mathcal{A}$. By doing so, the *FactoredMPA* algorithm reduces the structured multilinear programming Problem (13) with exponentially many constraints into a new smaller equivalent set of constraints. (This property is in fact inherited from the *FactoredLP* algorithm [5].)

Algorithm 6 has as input an MDPIP, the set of basis functions H and the order criterion O; and returns w and $\mathbf{p}$. Note that with these values we can compute the value function for each state.

FACTOREDMPA first calls the function CALCULATEOBJECTIVE that constructs the objective function of Problem (13). This is done by creating an expression that is the sum of the linear combination of the basis function $(\sum_{\mathbf{x}} \sum_{i=0}^{k} w_i h_i(\mathbf{x}))$ as it was done in [17]. Then, for each action a and each

Algorithm 2: FACTOREDMP($\mathcal{C}^a, \mathcal{R}^a, O$,MDPIP)

input : $\mathcal{C}^a$ (set of functions c_i^a),
$\mathcal{R}^a$ (set of local-reward R_j^a),
O (order criterion),
MDPIP (given by $\langle \mathcal{S}, \mathcal{A}, T, R, K, \gamma \rangle$)
output: $\mathbb{J}$ (set of constraints)

begin
  $\mathbb{J} = \{\}$;
  $\mathbb{F} = \{\}$; //set of new functions
  $\mathbb{J}, \mathbb{F} \leftarrow$ GENERATEEQUALITYCONSFORREWARD ($\mathbb{J}$,$\mathcal{R}^a, \mathbb{F}$);
  //Algorithm 3
  $\mathbb{J}, \mathbb{F} \leftarrow$ GENERATEEQUALITYCONSFORc_i ($\mathbb{J}$,$\mathcal{C}^a, \mathbb{F}$,MDPIP);
  //Algorithm 4
  n =number of state variables in the MDPIP;
  $\mathbb{J} \leftarrow$ GENERATEINEQUALITYCONS ($\mathbb{J}$,$\mathbb{F}, O, n$); //Algorithm 5
  return $\mathbb{J}$;
end

Algorithm 3: GENERATEEQUALITYCONSFORREWARD($\mathbb{J}$,$\mathcal{R}^a, \mathbb{F}$)

input : $\mathbb{J}$ (set of constraints),
$\mathcal{R}^a$ (set of local-reward R_j^a, with $1 \leq j \leq \psi$),
$\mathbb{F}$ (set of new functions)
output: $\mathbb{J}$, $\mathbb{F}$

begin
  $r = |\mathbb{F}| + 1$;
  for $j = 1$ *to* ψ **do**
    //create a constraint for each assigment to the scope of R_j^a
    Z=scope of R_j^a;
    foreach *assignment* z *of* Z **do**
      create a new variable $u_z^{e_r}$;
      add an equality constraint $u_z^{e_r} = R_j^a(z)$ to $\mathbb{J}$;
      $\mathbb{F} = \mathbb{F} \cup \{u_z^{e_r}\}$;
      r=r+1;
  return $\mathbb{J}, \mathbb{F}$;
end

Algorithm 4: GENERATEEQUALITYCONSFOR$c_i(\mathbb{J},\mathcal{C}^a,\mathbb{F})$

input : $\mathbb{J}$ (set of constraints),
$\mathcal{C}^a$ (set of functions c_i^a, with $1 \leq j \leq k$),
$\mathbb{F}$ (set of new functions),
MDPIP (given by $\langle \mathcal{S}, \mathcal{A}, T, R, K, \gamma \rangle$)
output: $\mathbb{J}$, $\mathbb{F}$

begin
  $r = |\mathbb{F}| + 1$;
  //create a new constraint for the constant basis function h_0
  calculate c_0^a for h_0; //Eq.(16) and (18)
  add an equality constraint $u^{e_r} = c_0^a * w_0$ to $\mathbb{J}$;
  $\mathbb{F} = \mathbb{F} \cup \{u^{e_r}\}$;
  r=r+1;
  //create a set of constraints for each c_i^a
  for $i = 1$ *to* k **do**
    //create a new constraint for each assigment to the scope of c_i^a
    Z=scope of c_i^a;
    foreach *assignment* z *of* Z **do**
      create a new variable $u_z^{e_r}$;
      add a multilinear equality constraint $u_z^{e_r} = c_i^a(z, \mathbf{p}) * w_i$ to $\mathbb{J}$;
      $\mathbb{F} = \mathbb{F} \cup \{u_z^{e_r}\}$;
      r=r+1;
  return $\mathbb{J}, \mathbb{F}$;
end

Algorithm 5: generateInequalityCons($\mathbb{J}$,$\mathbb{F}, O, n$)

input : $\mathbb{J}$ (set of constraints),
$\mathbb{F}$ (set of new functions),
O (order criterion),
n (number of state variables)
output: $\mathbb{J}$

begin
  $r = |\mathbb{F}| + 1$;
  for $i = 1$ *to* n **do**
    //select the variable to be eliminated
    $X_l = O(i)$;
    //select the relevant functions
    select $u^{f_1}_{Z_1}, \cdots, u^{f_L}_{Z_L}$ from $\mathbb{F}$ whose scope contains X_l;
    $Z = \cup_{j=1}^{L} Z_j \backslash \{X_l\}$;
    r=r+1;
    //define a new function $u^{e_r}_Z$
    foreach *assignment* z *of* Z **do**
      create a new variable $u^{e_r}_z$;
      add a constraint $u^{e_r}_z \geq \sum_{j=1}^{L} u^{f_j}_{\{z,xl\}}$ $\forall x_l$ to $\mathbb{J}$;
      //add the new function and remove the relevant functions
      $\mathbb{F} = \mathbb{F} \cup \{u^{e_r}_z\} \backslash \{u^{f_1}_{Z_1}, \cdots, u^{f_L}_{Z_L}\}$;
  //at the end all variables have been eliminated
  //all remaining functions u^{e_i} have empty scope
  add a constraint $0 \geq \sum_{e_i \in \mathbb{F}} u^{e_i}$ to $\mathbb{J}$;
  return $\mathbb{J}$;
end

Algorithm 6: FACTOREDMPA($MDPIP, H, O$)

input : $MDPIP$ (given by $\langle \mathcal{S}, \mathcal{A}, T, R, K, \gamma \rangle$),
H (set of basis functions),
O (order criterion)
output: $\{w, \mathbf{p}\}$

begin
  obj = CALCULATEOBJECTIVE($MDPIP, H$);
  foreach *action* $a \in \mathcal{A}$ **do**
    $\mathcal{C}^a$=COMPUTEc_i^a ($MDPIP, H, a$);//where $\mathcal{C}^a = \{c_1^a, \cdots, c_k^a\}$
  //initialize the set of constraints $\mathbb{J}$ with the probability constraints $\mathfrak{C}$
  $\mathbb{J} = \mathfrak{C}$;
  //the set of constraints $\mathbb{J}$ is expanded for each action $a \in \mathcal{A}$
  foreach *action* $a \in \mathcal{A}$ **do**
    $\mathbb{J} = \mathbb{J} \cup$ FACTOREDMP($\{c_1^a, \cdots, c_k^a\}, \{R_1^a, \cdots, R_\psi^a\}, O$,$MDPIP$);
  //the nonlinear solver is called with the new set of constrains $\mathbb{J}$
  $\{w, \mathbf{p}\}$=CALLNONLINEARSOLVER($obj, \mathbb{J}$);
  return $\{w, \mathbf{p}\}$;
end

basis function $h_i \in H$, FACTOREDMPA calculates c_i^a calling COMPUTEc_i^a (Algorithm 1). The set of constraints $\mathbb{J}$ is initialized with the probability constraints $\mathfrak{C}$. Next, for each action a the FACTOREDMP algorithm is called to compute a new smaller set of constraints that are added to $\mathbb{J}$. Finally, a nonlinear solver is called (algorithm CALLNONLINEARSOLVER) with the objective function and the new set of constraints $\mathbb{J}$, to solve a simpler and smaller multilinear problem, returning w and $\mathbf{p}$.

5. Experimental Results

To run our experiments we used the well-known System Administrator Problem [5], named SYSADMIN, where we have n computers $c_1, \ldots, c_n$ connected via two different directed graph topologies: *unidirectional-ring* and *star* (Figure 3). The administrator can execute $n+1$ actions: $reboot(c_1), \ldots, reboot(c_n)$ and $notreboot()$, which means not reboot any machine.

Let variable X_i denote whether computer c_i is up and running ($X_i = 1$) or not ($X_i = 0$). Let $Conn(c_j, c_i)$ denote a connection from c_j to c_i. Formally, the CPTs in the transition DCN for this domain have the following form:

$$P(X_i' = 1 | \mathbf{x}, a) = \begin{cases} if\ a = reboot(c_i): & then\ 1 \\ if\ a \neq reboot(c_i) \wedge X_i = 1: & then \\ & p_i \cdot \frac{|\{x_j | j \neq i \wedge X_j = 1 \wedge Conn(c_j, c_i)\}| + 1}{|\{x_j | j \neq i \wedge Conn(c_j, c_i)\}| + 1} \\ if\ a \neq reboot(c_i) \wedge X_i = 0: & then \\ & p_i' \cdot \frac{|\{x_j | j \neq i \wedge X_j = 1 \wedge Conn(c_j, c_i)\}| + 1}{|\{x_j | j \neq i \wedge Conn(c_j, c_i)\}| + 1} \end{cases}, \tag{27}$$

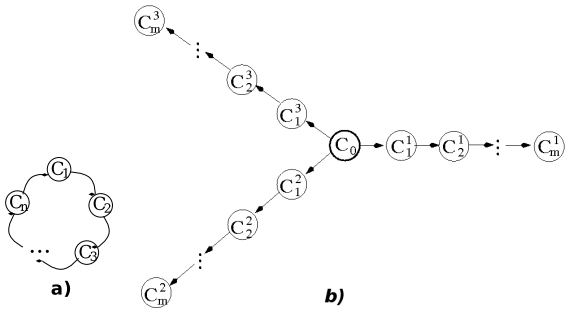

Figure 3: a)Unidirectional-ring and b)star connection topologies for the SysAdmin [5] example used in this paper.

and the constraints over the probability variables are:

$$0.85 + p'_i \leq p_i \leq 0.95, 1 < i < n.$$

The transition model tells that if a computer is rebooted then its probability of running in the next time step is 1, else the probability depends on its current status and the number of running computers with incoming connections. The probability variables p_i, p'_i and the constraint over them define the credal sets. Additionally, the reward is the number of computers that are running at the current time step: $R(\mathbf{x}, a) = \sum_{i=1}^{n} x_i$. An optimal policy in this problem will reboot the computer that has the most impact on the expected future discounted reward, given the network configuration. For example, in the star configuration on Figure 3, if the computers c_0 and c_2^2 are not running, the administrator should reboot c_0 since it will have the most impact on the whole network, that is, there are more machines depending on c_0 than c_2^2.

The instances of the above SysAdmin problem can be considered complex given the transition model defined by Expression (27), which has many transitions with imprecise probabilities. We have solved problems using two types of basis functions: (1) basis functions over single variables $h_i(X_i = 1) = 1$ and $h_i(X_i = 0) = 0$ and (2) basis functions over pairwise variables, that contain indicators for each neighboring pair of machines (x_i, x_{i+1}) as follows:

$$h_1(x_i, x_{i+1}) = x_i \wedge x_{i+1}$$
$$h_2(x_i, x_{i+1}) = \neg x_i \wedge x_{i+1}$$
$$h_3(x_i, x_{i+1}) = x_i \wedge \neg x_{i+1}$$
$$h_4(x_i, x_{i+1}) = \neg x_i \wedge \neg x_{i+1}$$

in both cases we have the constant basis function $h_0 = 1$.

We have implemented the FactoredMPA algorithm in Java calling MINOS [39] as the nonlinear solver (to solve the reduced multilinear program). In order to analyze the scalability of the proposed algorithm, we have calculated

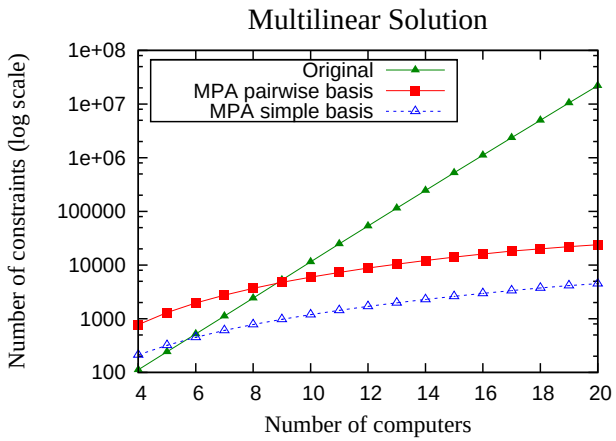

Figure 4: The number of constraints for the System Administrator domain with unidirectional-ring topology: (1) the original number of constraints; (2) the constraints after applying the FACTOREDMPA algorithm with simple basis functions; and (3) with pairwise basis functions.

the number of constraints **before** (i.e, the original number of constraints) for the SYSADMIN problems and **after** applying the algorithm FACTOREDMPA. Figure 4 shows the number of constraints for unidirectional-ring problems involving different numbers of computers. As we can notice the number of original constraints grows exponentially with the number of computers, while the constraints generated after applying the FACTOREDMPA algorithm grows quadratically. We can also notice that the number of constraints generated by FACTOREDMPA with simple basis functions (i.e., single variable functions) and pairwise basis functions have the same growing rate. Even though, the number of constraints for each problem in these two cases has a constant factor difference (≈ 4), the problems approximated by pairwise functions involve more complex constraints. We can see this by the time results obtained by applying our algorithms to solve those problems (Figure 5).

In Figures 5 and 6 we show the running time for problems with unidirectional-ring and star topology using the FACTOREDMPA algorithm for simple set and pairwise set of basis functions. We compare those results with the exact solutions given by our implementation of Value Iteration algorithm [8]. The results show that the exact solution is very time consuming when compared with the solution given by the FACTOREDMPA algorithm, that is able to solve large and complex problems (up to 2^{20} states). Thus, FACTOREDMPA algorithm is many orders of magnitude faster than the exact solution, due to its capability to generate a compact set of constraints and therefore, solving a less complex problem.

For unidirectional-ring configuration (Figure 5), the running time to solve problems with 16 computers (which would involve originally 2^{16} times 17 constraints) was less than 17 minutes and for problems with more than 16 com-

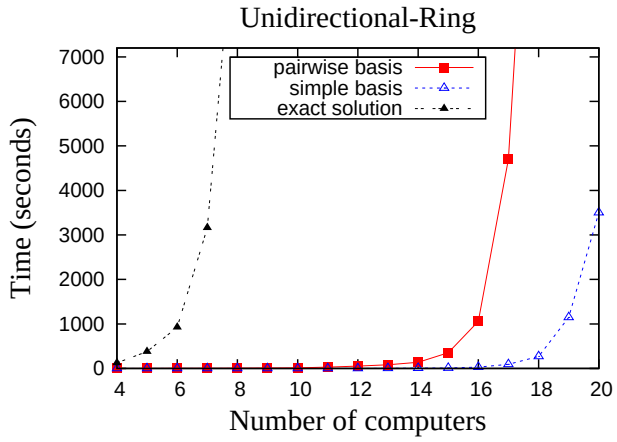

Figure 5: Running time of FACTOREDMPA algorithm using simple and pairwise basis functions, and an exact solution for the **Unidirectional-Ring configuration** of the System Administrator Domain.

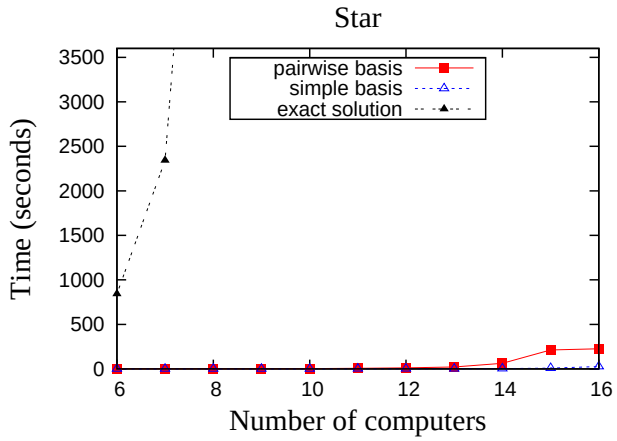

Figure 6: Running time of FACTOREDMPA algorithm using simple and pairwise basis functions, and an exact solution for the **Star configuration** of the System Administrator Domain.

puters the cpu-time grows quickly. A possible reason for this sudden change of behavior is that we have achieved the limit of complex constraints that the multilinear solver can handle. For star configuration (Figure 6), the running time to solve problems with 16 computers (with 2^{16} times 17 constraints) was less than 5 minutes.

In Figure 7 we show the percentage true approximation error, that is given by $(max_{\mathbf{x}}|V^*(\mathbf{x}) - V_{approx}(\mathbf{x})|)$ divided by the maximum discount reward sum $(R_{MAX}/1-\gamma)$ using pairwise basis functions and simple basis functions. To calculate $V^*(\mathbf{x})$ we used our implementation of Value Iteration algorithm. Since

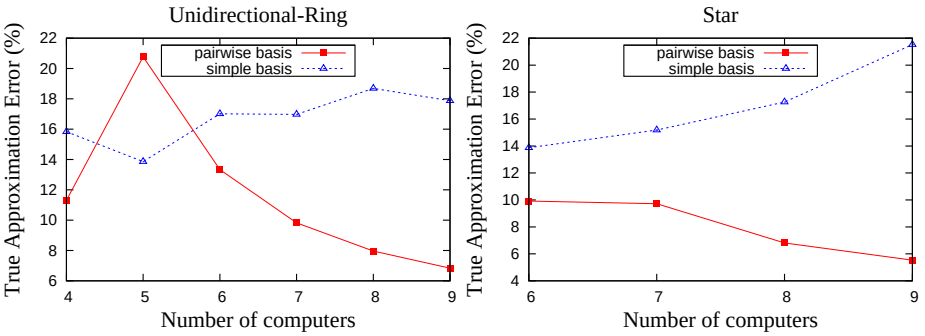

Figure 7: Percentage true Approximation Error ($max_{\mathbf{x}}|V^*(\mathbf{x}) - V_{approx}(\mathbf{x})|$ divided by the maximum discount reward sum) of FACTOREDMPA for the **Unidirectional-Ring configuration** and the **Star configuration** of the System Administrator Domain using pairwise basis functions and simple basis functions.

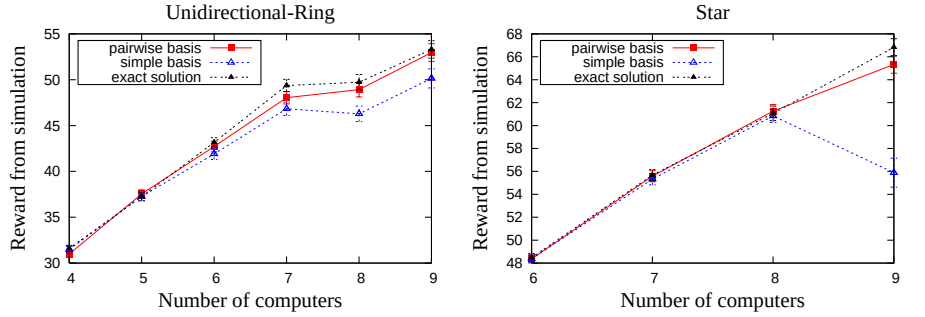

Figure 8: Value of policy estimated from simulation (average over 50 trials of 100 steps for 4 initial states) of an exact solution and FACTOREDMPA for the **Unidirectional-Ring configuration** and the **Star configuration** of the System Administrator Domain using pairwise basis functions and simple basis functions.

it can only solve small size problems, we show results only for problems where the exact solution [8] takes up to 10 hours. Note that the percentage in value loss incurred by using an approximation is up to 18% and, as expected, the use of pairwise basis functions resulted in better approximations (except for the problem with 5 computers with unidirectional configuration).

In Figure 8 we show the value of the policy estimated from simulation of an exact solution and FACTOREDMPA using pairwise basis functions and simple basis functions. We observe that the exact solution is a little better in some cases and the use of pairwise basis functions resulted in equal or better approximations.

If we additionally observe the time (Figures 5 and 6) and the percentage in value loss (Figures 7 and 8), we can conclude that this percentage incurred by using an approximation is up to 18% in exchange for a 1000 speedup.

6. Related Work

The *Bounded-parameter Markov Decision Process (BMDP)* [30] is a special case of an MDPIP, where the probabilities and rewards are specified by constant intervals. Exploiting the specific structure available in a BMDP given by the intervals, the algorithm in [30] can directly derive the solution without requiring expensive optimization techniques. Recent solutions to BMDPs include extensions of real-time dynamic programming (RTDP) [47] and LAO* [48, 49] that search for the best policy under the worst model. The *Markov Decision Process with Set-valued Transitions (MDPSTs)* [50] is another subclass of MDPIPs where probability distributions are given over finite sets of states. Since BMDP and MDPST are special cases of MDPIPs, we can represent both by "flat" MDPIPs. Then the algorithms defined in this paper clearly apply to both BMDPs and MDPSTs, however their solutions do not generalize to the factored MDPIPs we examined in this paper, which allow for general linear constraints between probabilities, which are prohibited in interval bounded probability settings like BMDPs. This use of general linear constraints is particularly useful when we do not know the probabilities or the interval they belong to, but only relative constraints between them.

Models involving imprecision have also been applied in the related field of Markov Chains with the work of Damjan Skulj [51].

7. Conclusion

In this work we have investigated Markov Decision Processes with Imprecise Probabilities (MDPIPs), a class of models that adds considerable flexibility and realism to probabilistic planning allowing the representation of imprecise transition probabilities. Inspired by the ideas of Guestrin's work on Factored MDPs, we first propose a compact Factored MDPIP model, which represents states throughout state variables and uses Dynamic Credal Networks to specify the imprecise transition probabilities. A Factored MDPIP is a more natural and compact representation of an MDPIP; it can reveal the structure of an application domain and allows for the construction of efficient and approximate solutions.

Second, based on mathematical programming formulation for MDPIPs [10] and Factored MDPs [5], we have proposed an approximate solution to Factored MDPIPs formulated as a multilinear program and we implement a new algorithm, named FactoredMPA, as an extension of the FactoredLPA (Factored Linear Programming-based Approximation) algorithm used to solve efficiently MDPs [5, 17]. The proposed algorithm finds maximin policies for Factored MDPIPs by resorting to approximate nonlinear programming and exploiting the domains structure in order to reduce the number of constraints of the multilinear program. We evaluated the FactoredMPA algorithm solving a difficult class of benchmark problems with up to 2^{20} states. Our experiments

show that by exploiting the factored representation, we gain orders of magnitude reduction in solution time over exact non-factored approach in exchange for less than 18% of approximation error.

To the best of our knowledge, this is the first work in the literature on MDPIPs that shows experimental results for problems with large state space sizes. MDPIPs were proposed in 1970, but have lacked general application for many decades, largely due to their computational complexity. Thus, we believe this is a significant contribution to both, the planning and operations research communities as well as to specific application domains where robust policies are important. A preliminary version of this paper was published in [52]. We are currently working on an extended version of our work on approximate solutions to solve Factored MDPIPs based on dynamic programming [53], in order to compare it with the multilinear approach. Our preliminary results show that, although with the dynamic programming approach we can give some error guarantees, with our FACTOREDMPA algorithm and the right choice of basis functions, we can solve larger problems.

Acknowledgements

This work has been supported by FAPESP grant 2008/03995-5; the first author was supported by CAPES, the third author is partially supported by CNPq and the fourth author is supported by NICTA. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] M. L. Puterman, Markov Decision Processes, John Wiley and Sons, New York, 1994.

[2] C. Boutilier, S. Hanks, T. Dean, Decision-theoretic planning: Structural assumptions and computational leverage, Journal of Artificial Intelligence Research 11 (1999) 1–94.

[3] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, The MIT Press, 2005.

[4] R. E. Bellman, Dynamic Programming, Princeton University Press, USA, 1957.

[5] C. Guestrin, Planning under uncertainty in complex structured environments, Ph.D. thesis, Stanford University, adviser-Daphne Koller (2003).

[6] J. Hoey, R. St-Aubin, A. Hu, C. Boutilier, SPUDD: Stochastic planning using decision diagrams, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann, 1999, pp. 279–288.

[7] R. St-Aubin, J. Hoey, C. Boutilier, APRICODD: Approximate policy construction using decision diagrams, in: Advances in Neural Information Processing Systems (NIPS), MIT Press, 2000, pp. 1089–1095.

[8] J. K. Satia, R. E. Lave Jr., Markovian decision processes with uncertain transition probabilities, Operations Research 21 (1973) 728–740.

[9] C. C. White III, H. K. El-Deib, Markov decision processes with imprecise transition probabilities, Operations Research 42 (4) (1994) 739–749.

[10] R. Shirota, F. G. Cozman, F. W. Trevizan, C. P. de Campos, L. N. de Barros, Multilinear and integer programming for Markov decision processes with imprecise probabilities, in: Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), Prague,Czech Republic, 2007, pp. 395–404.

[11] F. G. Cozman, Graphical models for imprecise probabilities, International Journal of Approximate Reasoning 39 (2-3) (2005) 167–184.

[12] C. Guestrin, D. Koller, R. Parr, S. Venkataraman, Efficient solution algorithms for factored MDPs, Journal of Artificial Intelligence Research 19 (2003) 399–468.

[13] D. P. Bertsekas, J. N. Tsitsiklis, An analysis of stochastic shortest path problems, Mathematics of Operations Research 16 (3) (1991) 580–595.

[14] R. A. Howard, Dynamic Programming and Markov Process, The MIT Press, 1960.

[15] A. S. Manne, Linear programming and sequential decision models, in: Management Science, Vol. 6(3), 1960, pp. 259–267.

[16] C. Papadimitriou, J. N. Tsitsiklis, The complexity of Markov decision processes, Mathematics of Operations Research 12 (3) (1987) 441–450.

[17] R. Patruscu, Linear approximations for factored Markov decision processes, Ph.D. thesis, University of Waterloo (2004).

[18] T. Dean, K. Kanazawa, A model for reasoning about persistence and causation, Computational Intelligence 5 (3) (1990) 142–150.

[19] P. J. Schweitzer, A. Seidmann, Generalized polynomial approximations in Markovian decision processes, Journal of Mathematical Analysis and Applications 110 (1985) 568–582.

[20] S. Mahadevan, Samuel meets Amarel: Automating value function approximation using global state space analysis, in: Proceedings of the 20th National Conference on Artificial Intelligence, 2005, pp. 1000–1005.

[21] A. Kolobov, Mausan, , D. S. Weld, Hidden structure of factored MDPs, in: Doctoral Consortium in International Conference on Automated Planning and Scheduling (ICAPS), Canada, 2010.

[22] I. Levi, The Enterprise of Knowledge, MIT Press, Cambridge, Massachusetts, 1980.

[23] J. Berger, Statistical Decision Theory and Bayesian Analysis, Springer-Verlag, 1985.

[24] P. J. Huber, Robust Statistics, Wiley, New York, 1980.

[25] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.

[26] A. Nilim, L. El Ghaoui, Robust control of Markov decision processes with uncertain transition matrices, Oper. Res. 53 (5) (2005) 780–798. doi:http://dx.doi.org/10.1287/opre.1050.0216.

[27] A. Wald, Statistical Decision Functions, Wiley, New York, 1950.

[28] L. V. Utkin, T. Augustin, Powerful algorithms for decision making under partial prior information and general ambiguity attitudes, in: Proceedings of the 3th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), Prague,Czech Republic, 2005, pp. 349–358.

[29] T. Seidenfeld, A contrast between two decision rules for use with convex sets of probabilities: Γ-maximin versus E-admissibility, Synthese 140 (1-2) (2004) 69–88.

[30] R. Givan, S. Leach, T. Dean, Bounded-parameter Markov decision processes, Artificial Intelligence 122 (1-2) (2000) 71–109.

[31] F. W. Trevizan, F. G. Cozman, L. N. de Barros, Planning under risk and Knightian uncertainty, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India, 2007, pp. 2023–2028.

[32] F. G. Cozman, Credal networks, Artificial Intelligence 120 (2) (2000) 199–233.

[33] W. Bialas, M. Karwan, Multilevel linear programming, Tech. Rep. 78-1, Operations Research Program, Department of Industrial Engineering, State University of New York at Buffalo (1978).

[34] J. F. Bard, Convex two-level optimization, Mathematical Programming 40 (1) (1988) 15–27.

[35] B. Colson, P. Marcotte, G. Savard, A trust-region method for nonlinear bilevel programming: Algorithm and computational experience, Computational Optimization and Applications 30 (3) (2005) 211–227.

[36] R. Andreani, S. L. C. Castro, J. L. Chela, A. Friedlander, S. A. Santos, An inexact-restoration method for nonlinear bilevel programming problems, Computational Optimization and Applications.

[37] G. Savard, J. Gauvin, The steepest descent direction for the nonlinear bilevel programming problem, Operations Research Letters 15 (5) (1994) 265–272.

[38] K. V. Delgado, L. N. Barros, Usando programação em dois níveis para resolver processos de decisão markovianos com probabilidades imprecisas, in: Congresso Brasileiro de Automática (CBA), Brazil, 2010, pp. 5284–5291.

[39] B. A. Murtagh, M. A. Saunders, MINOS 5.5 user's guide, Tech. Rep. SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, California (1998).

[40] R. Andreani, E. G. Birgin, J. M. Martinez, M. L. Schuverdt, Augmented Lagrangian methods under the constant positive linear dependence constraint qualification, Mathematical Programming 111 (1) (2007) 5–32.

[41] S. Boyd, S.-J. Kim, L. Vandenberghe, A. Hassibi, A tutorial on geometric programming, http://www.stanford.edu/∼boyd/papers/gp_tutorial.html (December 2009).

[42] H. D. Sherali, C. H. Tuncbilek, A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique, Global Optimization 2 (1992) 101–112.

[43] A. M. Lukatskii, D. V. Shapot, Problems in multilinear programming, Computational Mathematics and Mathematical Physics 41 (5) (2000) 638–648.

[44] D. Koller, R. Parr, Computing factored value functions for policies in structured MDPs, in: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), 1999, pp. 1332–1339.

[45] D. P. de Farias, B. V. Roy, On constraint sampling in the linear programming approach to approximate dynamic programming, Mathematics of Operations Research 29 (3) (2004) 462–478.

[46] D. A. Dolgov, E. H. Durfee, Symmetric approximate linear programming for factored MDPs with application to constrained problems, Annals of Mathematics and Artificial Intelligence 47 (3-4) (2006) 273–293.

[47] O. Buffet, D. Aberdeen, Robust planning with LRTDP, in: International Joint Conference on Artificial Intelligence (IJCAI), 2005, pp. 1214–1219.

[48] S. Cui, J. Sun, M. Yin, S. Lu, Solving uncertain Markov decision problems: An interval-based method, in: Second International Conference Advances in Natural Computatation (ICNC), 2006, pp. 948–957.

[49] M. Yin, J. Wang, W. Gu, Solving planning under uncertainty: quantitative and qualitative approach, in: IFSA (2), 2007, pp. 612–620.

[50] F. W. Trevizan, F. G. Cozman, L. N. de Barros, Planning under risk and Knightian uncertainty., in: International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 2023–2028.

[51] D. Skulj, Discrete time Markov chains with interval probabilities, International Journal of Approximate Reasoning 50 (8) (2009) 1314 – 1329.

[52] K. V. Delgado, L. N. de Barros, F. G. Cozman, R. Shirota, Representing and solving factored Markov decision processes with imprecise probabilities, in: Proceedings of the 6th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), Durham, United Kingdom, 2009, pp. 169–178.

[53] K. V. Delgado, S. Sanner, L. N. de Barros, F. G. Cozman, Efficient solutions to factored MDPs with imprecise transition probabilities, in: Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS), Thessaloniki, Greece, 2009, pp. 98–105.