Representing and Solving Factored Markov Decision Processes with Imprecise Probabilities

Karina Valdivia Delgado
Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo – Brazil
kvd@ime.usp.br

Leliane Nunes de Barros
Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo – Brazil
leliane@ime.usp.br

Fabio Gagliardi Cozman
Escola Politécnica
Universidade de São Paulo
São Paulo – Brazil
fgcozman@usp.br

Ricardo Shirota
Escola Politécnica
Universidade de São Paulo
São Paulo – Brazil
ricardo.shirota@poli.usp.br

Abstract

This paper investigates Factored Markov Decision Processes with Imprecise Probabilities; that is, Markov Decision Processes where transition probabilities are imprecisely specified, and where their specification does not deal directly with states, but rather with factored representations of states. We first define a Factored MDPIP, based on a multilinear formulation for MDPIPs; then we propose a novel algorithm for generation of Γ-maximin policies for Factored MDPIPs. We also developed a representation language for Factored MDPIPs (based on the standard PPDDL language); finally, we describe experiments with a problem of practical significance, the well-known System Administrator Planning problem.

Keywords. Imprecise Markov Decision Processes (MDPIPs), Probabilistic Planning and PPDDL, Knowledge Representation Languages, Multilinear programming.

1 Introduction

Sequential decision making is an essential activity in many domains, ranging from operations research [22] to robotics [29]. The last forty years have seen steady interest in Markov Decision Processes with Imprecise Probabilities (MDPIPs), since the seminal work by Satia and Lave Jr. [24]. Several algorithms have been developed for "flat" representations of MDPIPs, that is, representations that explicitly deal with individual states and transitions between states [13, 28, 34].

In this paper we focus on *factored* representations for MDPIPs. A factored representation deals with state variables that compactly encode a possibly large set of states. Factored versions of Markov Decision Processes (MDPs), where all probabilities are precisely specified, have received considerable attention [3], particularly in connection with large planning problems that arise in artificial intelligence. In fact, the leading representation language for probabilistic planning, PPDDL, is essentially a fragment of first-order logic that can specify Factored MDPs by using predicates to encode states [35]. In our previous work [28], we have briefly discuss Factored MDPIPs as we examined algorithms for "flat" MDPIPs. In the present paper we aim to: (1) give a definition of a factored MDPIP; (2) present an algorithm for policy generation; (3) propose a language for compact specification of factored MDPIP and (4) show some experiments with a well-known practical problem.

In Section 2 we review basic concepts on Factored MDPs. In Section 3, we describe the theory of "flat" MDPIPs and the relevant literature. In Section 4 we define factored representations and the PDL_1 language, a variant on PPDDL. In Section 5 we present an algorithm, which we call FACTOREDMPA (Factored Multilinear programming-based approximation), that produces Γ-minimax policies by resorting to Approximate Nonlinear Programming, and we show the performance of this algorithm in a well-known sequential decision problem described in PDL_1, the System Administrator Planning problem.

2 Markov Decision Processes and their Factored Representations

In this section we review basic facts about MDPs and the high-level representation language PPDDL.

Markov Decision Processes (MDPs) encode possibly infinite sequences of decisions under uncertainty [1, 22]. We are interested in MDPs that consist of (i) a countable set $\mathcal{T}$ of *stages*, such that a decision is made at each stage; (ii) a finite set $\mathcal{S}$ of *states*; (iii) a finite set of *actions* $\mathcal{A}(s)$ for each state s; (iv) a conditional probability distribution P_t that specifies the probability of transition from state s to state s' given action a at stage t, such that probabilities are stationary (do no depend on t) and written $P(s'|s,a)$; (v) a *reward* function R_t that indicates how much is gained (or lost, by using a negative value) when action a is selected in state s at stage t, such that the reward function is stationary and written $R(s,a)$.

The state obtained at stage t is denoted s_t; the action selected at stage t is denoted a_t. The history h_t of an MDP at stage t is the sequence of states and actions visited by the process, $[s_1, a_1, \ldots, a_{t-1}, s_t]$. The *Markov assumption* for MDPs adopts $P(s_t|h_{t-1}, a_t) = P(s_t|s_{t-1}, a_t)$. The main consequence of the Markov condition is that $P(h_t|s_1)$ factorizes as $P(s_t|s_{t-1}, a_{t-1})\, P(s_{t-1}|s_{t-2}, a_{t-2}) \ldots \times P(s_3|s_2, a_2)\, P(s_2|s_1, a_1)$. A *decision rule* $d_t(s,t)$ indicates the action that is to be taken in state s at stage t. A *policy* π is a sequence of decision rules, one for each stage. A policy may be *deterministic* or *randomized*; that is, it may prescribe actions with certainty, or rather it may just prescribe a probability distribution over the actions. A policy may be *history-dependent* or not; that is, it may depend on all states and actions visited in previous stages, or just on the current state. A policy that is not history-dependent is called *Markovian*. A Markovian policy induces a unique probability distribution over histories. Moreover, a Markovian policy needs only specify the prescribed action for each state: $\pi : S \to \mathcal{A}(s)$, where $\pi(s)$ is the action recommended by the policy π for the state s.

To compare different policies we adopt the discounted expected reward with infinite horizon [22]; in this case the solution is given by the *Bellman equation* as follows. First, introduce the concept of *value function* $V_\pi : S \to \mathbb{R}$, that defines the value of state s based on the values of the possible successor states $s' \in S$:

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

The factor γ in this expression is called the *discount* factor of the MDP [22, p. 125].

For MDPs the *optimal value function*, represented by V^*, is the value function associated with any optimal policy. Then, the Bellman equation is [14]:

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

The Bellman equation can be also formulated as a linear program [18]:

$$\begin{aligned} \min_{V^*} \quad & : \quad \sum_s V^*(s) \qquad (1) \\ s.t. \quad & : \quad V^*(s) \geq R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s'), \\ & \qquad \forall s \in S, a \in \mathcal{A}(s). \end{aligned}$$

Basically, we force $V^*(s)$ to be greater than or equal to $\max_a \{R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s')\}$; by minimizing $\sum_s V^*(s)$, we obtain the maximum value of the righthand side.

We now consider factored representations of MDPs; that is, MDPs where states are compactly specified using variables/predicates. In a *Factored MDP*, states $\vec{x}$ are represented by a set $\Lambda = \{X_1, X_2, ..., X_n\}$ of variables. Thus, a state $\vec{x} \in S$ is represented as a tuple $\{x_1, x_2, ..., x_n\}$ where x_i is the value of the state variable X_i. Note that the size of S is exponential in the number n of variables.[1]. Recent results have shown that it is possible to solve a Factored MDP with billions of states [12, 3].

In a Factored MDP, the reward function $R(\vec{x}, a)$ can be defined by the sum of local-rewards $R_i(\vec{x}, a)$.

$$R(\vec{x}, a) = \sum_{j=1}^{k_R} R_j(\vec{x}, a). \qquad (2)$$

The scope of each local-reward function R_j is typically restricted to some subset of variables $D_j \subset \Lambda = \{X_1, ..., X_n\}$, defined for each pair $\vec{x} \in S$ and $a \in \mathcal{A}(\vec{x})$.

The next step is to encode the transition probabilities. For each action a we define probabilities using a Dynamic Bayesian Network (DBN); that is, a directed acyclic graph with two layers: one representing the actual state and other representing the next state (Figure 1a). The nodes are denoted by X_i and X_i' for variables in the actual state and next state, respectively. 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 implies the following factorization of transition probabilities:

$$P(\vec{x}'|\vec{x}, a) = \prod_{i=1}^{n} P(x_i'|\text{pa}(X_i'), a); \qquad (3)$$

[1] Although the complexity of an MDP is P-Complete, i.e, an MDP problem is solved in a polynomial time in the size of the state space, it is exponential in the number of variables [20, 21].

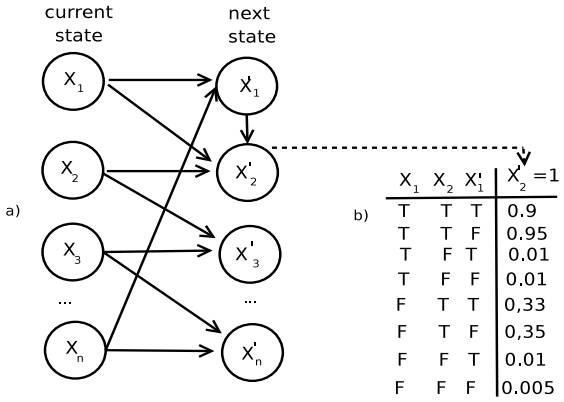

Figure 1: a) A DBN for an action a; b) a conditional probability table for the state variable X'_2.

that is, the probability to go to $\vec{x}' \in S$, given the agent is in state $\vec{x} \in S$ and executes the action $a \in \mathcal{A}(\vec{x})$, is the product of the conditional probability of the agent being in a state where $X'_i = x'_i$ given the parents of X'_i and the action $a \in \mathcal{A}(\vec{x})$ (Figure 1 b). We call P_a the set of conditional probability tables of a DBN for action a.

There are many methods to generate exact and approximate optimal policies for MDPs and Factored MDPs, including value and policy iteration. The technique of Approximate Linear Programming (ALP) [25] has recently been revisited as one of the most promising methods for solving complex Factored MDPs. Refinements for the ALP approach, geared towards Factored MDPs, have been developed over the past few years. The basic idea is to solve an MDP, formulated as Problem (1), by defining a set of basis functions and by using them to construct an approximation of the optimal value function, denoted by $\widehat{V}(\vec{x})$. Basis functions are provided by domain experts or automatically generated [21, 17]. Given $\vec{x} \in S$ and a set of of basis functions $H = \{h_1, ..., h_k\}$, $V^*(\vec{x})$ can be approximated using a linear combination of H:

$$\widehat{V}(\vec{x}) = \sum_{j=1}^{k} w_j h_j(\vec{x}). \tag{4}$$

The quality of the approximation depends on the algorithm used to find $\mathbf{w} = (w_1, ..., w_k)$, such that Equation (4) is a good approximation for $V^*(\vec{x})$. Thus, the ALP formulation of an MDP, given (1), (2) and (4), is the linear program:

$$\min_{w} \quad : \quad \sum_{\vec{x}} \sum_{i=1}^{k} w_i h_i(\vec{x}) \tag{5}$$

$$\text{s.t.} \quad : \quad \sum_{i=1}^{k} w_i h_i(\vec{x}) \geq \sum_{j=1}^{k_R} R_j(\vec{x}, a) + \gamma \sum_{\vec{x}' \in S} P(\vec{x}'|\vec{x}, a) \sum_{i=1}^{k} w_i h_i(\vec{x}'),$$
$$\forall \vec{x} \in S, a \in \mathcal{A}(\vec{x}).$$

The number of variables in the linear program (5) can be smaller than $|S|$, depending on the number of basis functions we have. However, the number of constraints does not change. ALP does not provide computational gains if we do not exploit the factored structure. In Section 5 we will discuss this fact in more detail.

In the last few years, many knowledge representation languages have been proposed for specifying factored MDPs. The most popular such language is Probabilistic Planning Domain Description Language (PPDDL)[35], a language based on first-order logic that has been applied to practical planning problems. In Section 4 we extend PPDDL to factored MDPIPs, and there we present the language in more detail.

3 Markov Decision Processes with Imprecise Probabilities

An MDPIP is simply an MDP where transition probabilities may be imprecisely specified. Note that the term MDPIP was proposed by White III and Eldeib [34], while Satia and Lave Jr. [24] adopt instead the term *MDP with Uncertain Transition Probabilities*.

To specify an MDPIP, one must specify all elements of an MDP except the transition probabilities; now one must specify a *set* of probabilities for each transition between states. We refer to these sets as *transition credal sets*. We assume stationarity for the transition credal sets $K(s'|s, a)$. We also assume that each history h_t is associated with stationary probability distributions $P(s_t|s_{t-1}, a_{t-1})$ that themselves satisfy the Markov condition (and of course $P(s_t|s_{t-1}, a_{t-1}) \in K(s_t|s_{t-1}, a_{t-1})$). That is, our MDPIPs are *elementwise-stationary* [28].

A few definitions are needed. We adopt elementwise conditioning: $K(X|A)$ is obtained from $K(X)$ by conditioning every distribution in the credal set $K(X)$ on the event A. The notation $K(X|Y)$ represents a *set* of credal sets: there is a credal set $K(X|Y = y)$ for each nonempty event $\{Y = y\}$. Given a credal set $K(X)$, we can compute *lower* and *upper* probabilities respectively as $\underline{P}(A) = \inf_{P \in K} P(A)$ and $\overline{P}(A) = \sup_{P \in K} P(A)$. We can also compute *lower* and *upper* expectations for any bounded function $f(X)$ as $\underline{E}[f] = \inf_{P \in K} E[f]$ and $\overline{E}[f] = \sup_{P \in K} E[f]$,

and likewise for conditional lower/upper probabilities/expectations. We assume all credal sets to be closed, so infima and suprema can be replaced by minima and maxima.

There are several criteria of choice for selecting policies in a given MDPIP, even if we fix a single utility and focus on discounted infinite horizon. The Γ-maximin criterion selects a policy that yields the supremum of lower expected reward. In the context of discounted infinite horizon, there is always a deterministic stationary policy that is Γ-maximin [24]; moreover, this policy induces a value function that is the unique solution of

$$V^*(s) = \sup_a \inf_P \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^*(s') \right), \tag{6}$$

subject to the fact that probabilities must belong to given transition credal sets. Given our assumption that sets of actions are finite and credal sets are closed, we can replace sup and inf respectively by max and min in this expression.

A few other criteria of choice are worth mentioning. The Γ-maximax criterion [24] selects a policy that yields the supremum of upper expected reward [33], while the Γ-maximix criterion selects a policy that yields the maximum of $\alpha(\max_P V_\pi) + (1-\alpha)(\min_P V_\pi)$, for some $\alpha \in (0,1)$. Other criteria seek *sets* of admissible policies, such as the Interval Dominance, Maximality and E-admissible criteria [15]. There are strong foundational reasons to side with the most restrictive of the last three criteria; that is, to adopt E-admissibility [26]. However, in this paper we adopt the Γ-maximin criterion due to its popularity in the existing literature on MDPIPs. We certainly hope to examine other criteria in our future work on Factored MDPIPs.

There are algorithms for solving flat MDPIPs based on dynamic programming [24, 34]. Additional algorithms have been proposed to solve special cases of MDPIPs [10, 31].

4 Defining and representing Factored MDPIPs

We define a *Factored MDPIP*, intuitively enough, as an MDPIP where states are compactly specified using variables/predicates. Thus we have a Factored MDP where transition probabilities are not unique, but rather given by transition credal sets. The challenge then is to specify such transition credal sets in a manner that is itself compact. We suggest that *Dynamic Credal Networks* (DCNs) offer the adequate language to express transition credal sets.

A DCN has the same structure as a DBN (Figure 1), but now each variable X is associated with a set of conditional credal sets; that is a credal set $K(X|\text{pa}(X) = k)$ for each value k of $\text{pa}(X)$. In this paper we assume that every DCN represents a joint credal set over all of its variables, and this joint credal set is exactly the *strong extension* of the credal network [5, 6]. That is, the DCN represents a joint credal set where each distribution satisfies the following expression:

$$P(\vec{x}'|\vec{x},\vec{p},a) = \prod_{i=1}^{n} P(x'_i|\text{pa}(X'_i),\vec{p},a); \tag{7}$$

where each $P(x'_i|\text{pa}(X'_i),\vec{p},a)$ comes from an appropriate credal set associated with the DCN.

Consider the generation of Γ-maximin policies; that is, solution of Equation (6). It does not seem possible to produce a linear programming solution like the linear programming for MDP (Problem 1). However in our previous work [28] we have shown that it is possible to generate solutions using well know programming problems. First, the Equation (6) can be reduced to a *bilevel programming problem*:

$$\begin{aligned} \min_{V^*} \quad & : \quad \sum_s V^*(s) \qquad (8)\\ \text{s.t.} \quad & : \quad V^*(s) \geq R(s,a) + \\ & \qquad \gamma \sum_{s' \in S} P(s'|s,a) V^*(s'), \forall s \in S, a \in A; \\ & \qquad P \in \arg\min \sum_{s' \in S} P(s'|s,a) V^*(s'), \\ & \qquad \text{s.t.} : P(s'|s,a) \in K_a(s'|s) \end{aligned}$$

Then, the bilevel problem (8) can be transformed in an equivalent *multilinear programming problem*:

$$\begin{aligned} \min_{V^*,P} \quad & : \quad \sum_s V^*(s) \qquad (9)\\ \text{s.t.} \quad & : \quad V^*(s) \geq R(s,a) + \\ & \qquad \gamma \sum_{s' \in S} P(s'|s,a) V^*(s'), \\ & \qquad \forall s \in S, a \in \mathcal{A}(s), P(s'|s,a) \in K_a(s'|s,a). \end{aligned}$$

Note that the solution of multilinear programs is far from trivial, thus our previous solution can only deal with relatively small flat MDPIPs.

We now specialize Problem (9) for Factored MDPIPs. The *factored value function* of a Factored MDPIP is given by Equation (4) restricting the scope of each basis function to some small subset of state variables $C_i \subset \Lambda = \{X_1, ..., X_n\}$. We can use the factored value function (4), the reward function (2), the transition probabilities (7) and replace them in Problem (9) in

order to obtain the factored multilinear programming problem:

$$\begin{aligned} \min_{w,\vec{p}} \quad & : \quad \sum_s \sum_i w_i h_i(\vec{\mathbf{x}}) \qquad (10) \\ s.t. \quad & : \quad \sum_i w_i h_i(\vec{x}) \geq \sum_{j=1}^{kR} R_j(\vec{x}, a) + \\ & \qquad \gamma \sum_{\vec{x}' \in S} P(\vec{x}'|\vec{x}, \vec{p}, a) \sum_i w_i h_i(\vec{x}'), \\ & \quad \forall \vec{x} \in S, a \in \mathcal{A}(\vec{x}). \\ & \quad P(\vec{x}'|\vec{x}, \vec{p}, a) \in K_a(\vec{x}'|\vec{x}) \end{aligned}$$

where:

$$P(\vec{x}'|\vec{x}, \vec{p}, a) = \prod_{i=1}^{n} P(x_i'|\mathrm{pa}(X_i'), \vec{p}, a)$$

This particular nonlinear program will be studied in the next section; the main contribution of this paper is an algorithm for the generation of Γ-maximin policies in Factored MDPIPs that solves Problem (10). Before we plunge into that, we spend the remainder of this section discussing the representation of Factored MDPIPs.

As we have mentioned before, the *Probabilistic Planning Domain Description Language* (PPDDL) [2] is a high-level language for the specification of Factored MDPs, with a relatively simple syntax. Every planning problem is expressed in two parts: the `domain` contains directives, constants, and descriptions of actions; the `problem` basically contains a description of the initial state and the desired goal. We wish to focus on the syntax and semantics of `domain`s, so we present the relevant pieces of the syntax here. The basic BNF for `domain`s is:

```
<domain> ::= (define (domain <NAME>)
    (:requirements :adl)
      [<types>][<constants>][<predicates>]
        <action>*)
<action> ::= (:action <NAME>
       [<param>] [<prec>] [<effect>])
<prec> ::= (:precondition <p-formula>)
<effect> ::= (:effect {<nd-eff>|<det-eff>})
<nd-eff> ::= <prob>|<one-of>
<prob> ::= (probabilistic <p-eff>+)
<p-eff> ::= <RATIONAL> <det-eff>
<one-of> ::= (oneof <det-eff>+),
```

where: `<types>`, `<constants>`, `<predicates>` and `<param>` refer to lists of names or logical variables (possibly typed); `<RATIONAL>` denotes a rational number; `<p-formula>` is a formula containing either atoms, or conjunction of `p-formula`s, or universal quantification over `p-formula`s, or inequality of two given names as `(not (= <NAME> <NAME>))`; and a `<det-eff>` is a formula containing either atoms, or negation of atoms, or conjunction of `det-eff`s, or universal quantification over `det-eff`s, or the *conditional* operator `when`. This conditional operator has syntax `(when <p-formula> <simple-eff>)`, where `simple-eff` is a formula containing either atoms, or negations of atoms, or conjunction of `det-eff`s, or universal quantification over `simple-eff`s.

In PPDDL, a probabilistic action is understood as a probabilistic transition given by a Dynamic Bayesian Network [35]. PPDDL also allows an action to contain `oneof` elements, where a nondeterministic choice is made and one of the effects listed in the scope of the `oneof` element is selected and pursued. There are no probabilities attached to such nondeterministic choices. We call these conventions the *standard semantics* of PPDDL. Note that the standard semantics of PPDDL takes us beyond Markov Decision Processes (MDPs) given the presence of nondeterminism; however the expressivity of PPDDL is still far from general MDPIPs, because in PPDDL each action may contain *either* a probabilistic effect *or* a nondeterministic effect. For instance, a domain may contain two actions, one with probabilistic effects, and the other with nondeterministic effects. What is not allowed in PPDDL is the mixture of probabilistic and nondeterministic effects *in the same action.*

In a previous publication we have explored the facilities of PPDDL to express planning problems where probabilistic and nondeterministic choices (in the PPDDL sense) are mixed, but only allowing that all probabilistic choices precede all nondeterministic choices in an action [30]. The reason for this restriction is that the ensuing planning problems are instances of MDPIPs were all transition credal sets are given by infinitely monotone Choquet capacities. We refer to PPDDL with this added flexibility as PDL_2, and we refer to PPDDL with no restrictions on the combination of probabilistic and nondeterministic choices as PDL_1. We note that PDL_1 can specify Factored MDPIPs with clear syntax; to illustrate this fact, we consider the well-known System Administrator Problem [12].[2]

Example 1 *Consider the problem of optimizing the behavior of a system administrator that works with a network of computers. There are many possible configurations; for example, the cycle network where com-*

[2]This example is actually a variant on the original Factored MDP for the System Administrator problem, because some additive aspects cannot be encoded in PPDDL [23]. The way we solve this limitation was to start with a high reward value and decrease every time the action `reboot` was executed (effect `(decrease (reward) 1)` from Figure 2).

puter i is connected to computer $i+1$. One of these computers is designated as server, while the rest are clients. Each computer is associated to a binary variable X_i, the value of a variable indicates whether the respective machine is up (1) or down (0). At each time step the administrator receives a payment (reward) for each machine that is working. Since the server is the most important computer in the network it is given a greater reward if it is working. The job of the system administrator is deciding which of the machines should reboot. So, there are n+1 possible actions at each step: reboot one of the n machines or not reboot any machine. After executing the action reboot the machine i, the probability of machine i to be working on the next step is high. At each step each computer has a low probability to stop working, which grows dramatically if their neighbors are not working. The machines can begin working spontaneously with a small probability.

In the original PPDDL for the System Administrator Domain [23], the probability of a computer i start working in the next state, given that the action `reboot(i)` was executed, is 0.9; and with the probability 0.1 the state remains unchanged. Also, with probability 0.6 the state variable x_i (computer i) becomes false in the next state, when it is connected with other computer that it is not working (and the computer i has not been rebooted); and with probability 0.4 the state remains unchanged.

Figure 2 presents a PDL_1 specification of a Factored MDPIP that represents the System Administrator Problem, where experts disagree on the probability distributions. With probability between 0.6 and 0.8 the state variable x_i (computer i) becomes false in the next state, if it is connected with other computer that it is not working (and the computer i has not been rebooted). Considering an instance of the domain described in Figure 2 with 3 state variables, which implies 8 states and 3 actions: a_1 for *reboot computer 1*, a_2 for *reboot computer 2* and a_3 for *reboot computer 3*, the corresponding factored MDPIP have the following set of constraints:

$$P(X_i' = 1 | X_1 = 0, a_i) = 0.9$$
$$P(X_i' = 1 | X_1 = 1, a_i) = 1$$

And for $i \neq j$ we have:

$$P(X_i' = 0 | X_{i-1} = 0, X_i = 0, a_j) = 1$$
$$0.6 <= P(X_i' = 0 | X_{i-1} = 0, X_i = 1, a_j) <= 0.8$$
$$P(X_i' = 0 | X_{i-1} = 1, X_i = 0, a_j) = 1$$
$$P(X_i' = 0 | X_{i-1} = 1, X_i = 1, a_j) = 0$$

Instances such these are solved by the algorithm presented in the next section.

```
(define (domain sysadmin)
(:requirements :adl)
(:types comp)
(:predicates (up ?c)(conn ?c ?d))
(:action reboot
  :parameters (?x - comp)
  :effect
     (and (decrease (reward) 1)
          (probabilistic 0.9 (up ?x))
          (oneof
          (forall (?d - comp)
            (probabilistic
              0.6 (when (exists (?c - comp)
                      (and (conn ?c ?d)
                         (not (up ?c))
                         (not (= ?x ?d))))
                  (not (up ?d))
          )))
          (forall (?d - comp)
            (probabilistic
              0.8 (when (exists (?c - comp)
                      (and (conn ?c ?d)
                          (not (up ?c))
                          (not (= ?x ?d))))
                  (not (up ?d))
     )))))

)
(define
 (problem sysadmin-3)
   (:domain sysadmin)
   (:objects x1 - comp x2
                - comp x3
                - comp)
(:init (conn x1 x2)
       (conn x2 x3)
       (conn x3 x1))
(:goal (forall (?c - comp)
               (up ?c)))
(:goal-reward 500)
)
```

Figure 2: The System Administrator domain in PDL_1, with action *reboot* (probabilistic and nondeterministic). This domain defines a Factored MDPIP (adapted from [23]). One limitation of this language is do not allow to express local-reward and basis functions for approximated solutions of MDPs and MDPIPs.

5 FACTOREDMPA: Solving a Factored MDPIP

Koller and Parr [16] show that if we are working with a Factored MDP (Problem 5), a necessary condition to efficiently apply the ALP technique is to restrict the scope of each basis function to some small subset of state variables $C_i \subset \Lambda = \{X_1, ..., X_n\}$ and also to assume small dependency in the DBN[3]. Guestrin et al. [12] then exploited these conditions and developed an efficient algorithm for Factored MDPs. The success of their FACTOREDLPA algorithm is due to: (i) the use of a method to simplify the computation of each constraint of the ALP problem, named *Backprojection* algorithm [16]; and (ii) the *FactoredLP* algorithm that creates a new and smaller set of equivalent constraints for the linear programming problem (5). There are other efficient algorithms that use general techniques to solve linear problems with large number of constraints [21, 8, 9] (e.g., constraints generation), and that somehow, have improved the approach proposed by Guestrin [12].

Based on those ideas, we want to solve a Factored MDPIP formulated as an Approximated Multilinear Programming (Problem 10). First, the same efficient and general techniques that solve linear problems with large number of constraints [21, 8, 9] cannot be applied directly on the multilinear problem. However, the FACTOREDLPA algorithm can be adapted to solve a factored MDPIP as we show in this section. The new algorithm we will name as FACTOREDMPA.

Shortly, FACTOREDMPA first simplifies the computation of each constraint applying the same *Backprojection* algorithm used by Guestrin for factored MDP, then it calls the *FactoredMP* algorithm to create a new and smaller equivalent set of constraints for the Multilinear Programming (Problem 10). Finally, in order to obtain w_i and $\vec{p}$, it calls a nonlinear solver with the new equivalent problem.

5.1 Simplifying the computation of each constraint

We can also take advantage of the fact that the transition model for MDPIPs is factored and the basis functions have scope restricted to a small set of variables in order to efficiently compute the constraints.

From problem (10), given $\vec{x} \in S$ and $a \in \mathcal{A}(\vec{x})$, we have the following constraint:

$$\sum_i w_i h_i(\vec{x}) \geq \sum_{j=1}^{kR} R_j(\vec{x}, a) + \gamma \sum_{\vec{x}' \in S} P(\vec{x}'|\vec{x}, \vec{p}, a) \sum_i w_i h_i(\vec{x}')$$

[3]Although this assumption seems too restrictive, there is a large set of applications that it can be done [11].

Now, we can reorder the sum and obtain:

$$\sum_i w_i h_i(\vec{x}) \geq \sum_{j=1}^{kR} R_j(\vec{x}, a) + \gamma \sum_i w_i \underbrace{\sum_{\vec{x}' \in S} P(\vec{x}'|\vec{x}, \vec{p}, a) h_i(\vec{x}')}$$

Let the underbrace term be renamed as $g_i^a(\vec{x}, \vec{p})$. Note that, for MDPIPs, $g_i^a(\vec{x}, \vec{p})$ is a polynomial expression, i.e. it is described in terms of probability variables and has the following canonical form (with $d_0 = 0$ and d_i a constant):

$$d_0 + \sum_i d_i \prod p_{ij} \tag{11}$$

This term can be precomputed in a efficient way using the *Backprojection* algorithm [16]. For a further computation improvement, the set of constraints can be rewritten as:

$$0 \geq \sum_{j=1}^{kR} R_j(\vec{x}, a) + \sum_i w_i \left(\underbrace{\gamma g_i^a(\vec{x}, \vec{p}) - h_i(\vec{x})} \right)$$

Again, let the underbrace term be renamed as $c_i^a(\vec{x}, \vec{p})$. This term can be precomputed resulting also in the polynomial form (11). Finally $\forall \vec{x} \in S, a \in \mathcal{A}(\vec{x})$ we obtain:

$$0 \geq \sum_{j=1}^{kR} R_j(\vec{x}, a) + \sum_i w_i c_i^a(\vec{x}, \vec{p}) \tag{12}$$

Even with this simplified form to rewrite constraints for the Approximate Multilinear Programming, we are still working with the complete set of constraints ($|S| * |A| + m_2$), where m_2 is the number of constraints related to the probabilities p_{ij}. Since the direct use of general non-linear solvers [19], geometric solvers [4] or multilinear solvers [27] for Problem (10), can only solve problems with small state space, we have to find a way to reduce the number of constraints.

5.2 The *FactoredMP* algorithm

We extend the *FactoredLP* technique proposed by Guestrin [12] in order to obtain a new and smaller equivalent multilinear program for Problem (10). We call this new algorithm *FactoredMP*.

The basic idea is to replace the set of constraints in (12) by an equivalent set of non-linear constraints $\forall a \in \mathcal{A}(\vec{x})$, given by:

$$0 \geq \max_{\vec{x}} \left\{ \sum_{j=1}^{kR} R_j(\vec{x}, a) + \sum_i w_i c_i^a(\vec{x}, \vec{p}) \right\}$$

So, for an action a, we have to compute the following maximization:

$$0 \geq \max_{\vec{x}} \left\{ \sum_{j=1}^{kR} R_j(\vec{x}) + \sum_i w_i c_i(\vec{x}, \vec{p}) \right\} \tag{13}$$

Note that $R_j(\vec{x})$ and $c_i(\vec{x},\vec{p})$ are functions of $\vec{x}$ and we want to do max over $\vec{x}$. Now we can, instead of adding all terms and do the maximization, do the maximization over state variables one by one. To do so we use a modification of the general variable elimination algorithm proposed by Guestrin [12].

For example, if we want to eliminate variable X_1 we do as following. If R_1 is the only local-reward function that depends on X_1 and c_1 is a function that depends on $(X_1,\ X_4)$ and there is no other function c_i that depends on X_1, we can push the maximization over X_1 inwards to obtain:

$$0 \geq \max_{X_2 \ldots X_n} \left\{ \sum_{j=2}^{kR} R_j(\vec{x}) + \sum_{i=2} w_i c_i(\vec{x},\vec{p}) + \max_{X_1} \{R_1(X_1) + w_1 c_1(X_1, X_4, \vec{p})\} \right\}$$

For each variable X_l we want to eliminate, *FactoredMP* selects L relevant functions, renamed as $u^{e_1} \ldots u^{e_L}$. A relevant function is the one whose scope contains X_l. We can now replace the maximization over the relevant functions for X_l by the following new function:

$$u_Z^{e_{new}} = \max_{X_l} \sum_{j=1}^{L} u^{e_j} \qquad (14)$$

Where Z is the union of all variables in functions $u^{e_1} \ldots u^{e_L}$ minus X_l. In the above example, the relevant functions are $u_{X_1}^{e_1} = R_1(X_1)$ and $u_{X_1,X_4}^{e_2} = w_1 c_1(X_1, X_4, \vec{p})$. The term $u_Z^{e_{new}}$ is $u_{X_4}^{e_{new}} = \max_{X_1} \left\{ u_{X_1}^{e_1} + u_{X_1,X_4}^{e_2} \right\}$, resulting in the following constraint:

$$0 \geq \max_{X_2 \ldots X_n} \left\{ \sum_{j=2}^{kR} R_j(\vec{x}) + \sum_{i=2} w_i c_i(\vec{x},\vec{p}) + u_{X_4}^{e_{new}} \right\}$$

In order to enforce the definition of $u_Z^{e_{new}}$ as the maximum over X_l (Eq. 14), *FactoredMP* introduces the following set of constraints for any assignment z to Z:

$$u_Z^{e_{new}} \geq \sum_{j=1}^{L} u^{e_j} \forall x_l$$

In the example we need to introduce four constraints: one constraint for each configuration of X_4 and for each configuration of X_1.

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

$$0 \geq \sum_{j=i} u^{e_i}$$

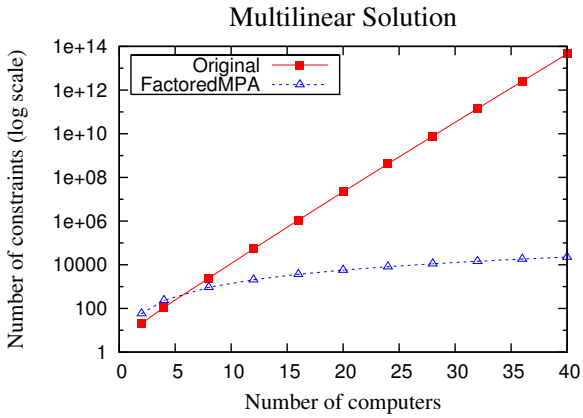

Figure 3: The number of constraints for the System Administrator domain, with imprecise probabilities, for problems with n computers; there are 2^n number of states and $n+1$ actions in this problem.

Notice that, the same process must be applied for all actions $a \in A$. The *FactoredMP* algorithm reduces a structured multilinear programming problem (10) with exponentially many constraints to a new smaller equivalent set of constraints. This property is inherited from the *FactoredLP* procedure.

5.3 Experimental Results

In order to analyze the scalability of the proposed algorithm, we have calculated the original number of constraints and the number of constraints after applying the algorithm FACTOREDMPA for the problem (10). In order to do this, we consider the System Administrator domain (described in the previous section). Figure 3 shows the result for problems varying the number of computers from 2 to 40. The graph shows the original number of constraints grows exponentially while the constraints generated by the FACTOREDMPA algorithm grows quadratically with the number of computers.

We have implemented the FACTOREDMPA algorithm using Matlab as frontend, and MINOS as the nonlinear solver (to handle the reduced multilinear programs). In Figure 4 we show the running times for the System Administrator domain described as in Figure 2 using a simple set of basis functions: the constant function $h_0 = 1$ and $h_i(X_i = 1) = 1$ and $h_i(X_i = 0) = 0$. These results show that with the FACTOREDMPA algorithm it is possible to solve large problems, e.g. we solve in 300 seconds a problem with 40 computers which in the original AMP formulation would have more than $2^{40} * 41$ constraints.

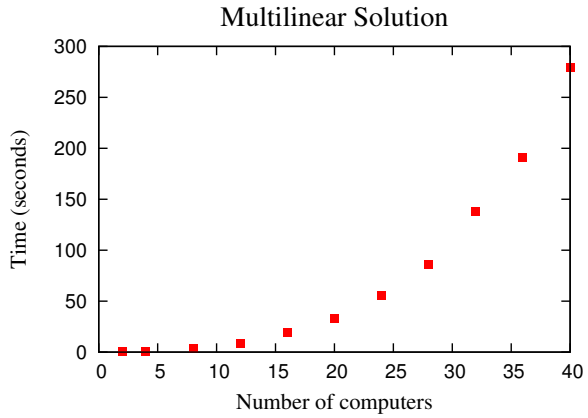

Figure 4: Running time of FACTOREDMPA for the System Administrator Domain Example 1

6 Conclusion

In this paper we have investigated Markov Decision Processes with Imprecise Probabilities, a class of models that adds considerable flexibility and realism to the popular Markov Decision Processes. We have defined a **Factored MDPIP** problem based on a multilinear formulation for MDPIPs [28] and Factored MDPs [12]. We also developed a representation language to specify Factored MDPIPs, named PDL_1, which extends PPDDL by allowing free mixtures of probabilistic and nondeterministic operators. Although PDL_1 does not allow to specify basis and local-reward functions, it is an original and practical high-level language to express factored MDPIPs. Further, we can take advantage of the fact the PPDDL language has largely been used as a benchmark language to solve probabilistic planning problems and, with a simple modification on those problems to obtain a PDL_1 specification, we can have a variety of MDPIP problems.

Our main contribution is a new algorithm, named FACTOREDMPA, to find Γ-maximin policies for Factored MDPIPs. The algorithm is an adaptation of the FACTOREDLPA (Factored Linear Programming-based Approximation) algorithm used to solve Factored MDPs [12, 21]. To evaluate the FACTOREDMPA algorithm, we have modified the System Administrator problem by introducing imprecision in probability values. We thus obtain Factored MDPIPs with varying sizes. Our experiments show that by exploiting the factored representation of a sequential decision problem, and by making the assumption of a restrict scope for variable dependences, relatively large problems can be solved (note that the number of constraints and cpu-time grows quadratically with the number of variables).

Acknowledgements

This work has been supported by FAPESP grant 2008/03995-5; the first author is supported by CAPES; the third author is partially supported by CNPq; and the fourth author is supported by FAPESP. Tests were run in MATLAB and AMPL that calls a multilinear programming solver MINOS.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Math. Oper. Res.*, 16(3):580–595, 1991.

[2] B. Bonet and R. Givan. International Planning Competition: Non-deterministic track — Call for Participation, December 2005.

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

[4] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming, 2004.

[5] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.

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

[7] G. de Cooman and M. C. M. Troffaes. Dynamic programming for deterministic discrete-time systems with uncertain gain. *International Journal Approximate Reasoning*, 3(2-3):257–278, 2005.

[8] D. P. de Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.*, 29(3):462–478, 2004.

[9] D. A. Dolgov and E. H. Durfee. Symmetric approximate linear programming for factored MDPs with application to constrained problems. *Ann. Math. Artif. Intell.*, 47(3-4):273–293, 2006.

[10] R. Givan, S. Leach, and T. Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122:71–109(39), 2000.

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

[12] C. E. Guestrin. *Planning under Uncertainty in Complex Structured Environments*. PhD thesis, Stanford University, 2003. Adviser-Daphne Koller.

[13] D. Harmanec. Generalizing Markov decision processes to imprecise probabilities. *Journal of Statistical Planning and Inference*, 105:199–213, 2002.

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

[15] D. Kikuti, F. G. Cozman, and C. P. de Campos. Partially ordered preferences in decision trees: computing strategies with imprecision in probabilities. In *IJCAI Workshop on Advances in Preference Handling*, pages 118–123, Edinburgh, United Kingdom, 2005.

[16] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *IJCAI*, pages 1332–1339, 1999.

[17] S. Mahadevan. Samuel meets Amarel: Automating value function approximation using global state space analysis. In *AAAI*, pages 1000–1005, 2005.

[18] A. S. Manne. Linear programming and sequential decision models. In *Management Science*, volume 6(3), pages 259–267, 1960.

[19] B. A. Murtagh, M. A. Saunders, W. Murray, P. E.Gill, R. Raman, and E. Kalvelagen. Minos: A solver for large-scale.

[20] C. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

[21] R.-E. Patruscu. *Linear Approximations for Factored Markov Decision Processes*. PhD thesis, University of Waterloo, 2004.

[22] M. L. Puterman. *Markov Decision Processes*. Wiley series in probability and mathematical statistics. John Wiley and Sons, New York, 1994.

[23] S. Sanner. How to spice up your planning under uncertainty research life, 2008. In Workshop on A Reality Check for Planning and Scheduling Under Uncertainty at ICAPS, 2008.

[24] J. K. Satia and R. E. Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21:728–740, 1970.

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

[26] 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.

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

[28] R. Shirota, F. G. Cozman, F. W. Trevizan, C. P. de Campos, and L. N. de Barros. Multilinear and integer programming for markov decision processes with imprecise probabilities. In *5th ISIPTA*, pages 395–404, Prague,Czech Republic, 2007.

[29] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[30] F. W. Trevizan, F. G. Cozman, and L. N. de Barros. Mixed probabilistic and nondeterministic factored planning through Markov decision processes with set-valued transitions. In Workshop on A Reality Check for Planning and Scheduling Under Uncertainty at ICAPS, 2008.

[31] F. W. Trevizan, F. G. Cozman, and L. N. de Barros. Planning under Risk and Knightian Uncertainty. In *Proc. of IJCAI*, pages 2023 – 2028, Hyderabad, India, 01 2007. AAAI.

[32] M. C. M. Troffaes. Learning and optimal control of imprecise Markov decision processes by dynamic programming using the imprecise Dirichlet model. pages 141–148, Berlin, 2004. Springer.

[33] L. V. Utkin and T. Augustin. Powerful algorithms for decision making under partial prior information and general ambiguity attitudes. *ISIPTA*, pp. 349–358, 2005.

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

[35] H. L. S. Younes, M. L. Littman, D. Weissman, and J. Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24:851–887, 2005.