Factored Markov Decision Process with Imprecise Transition Probabilities

Karina V. Delgado
EACH-University of Sao Paulo
Sao Paulo - Brazil

Leliane N. de Barros
IME-University of Sao Paulo
Sao Paulo - Brazil

Scott Sanner
NICTA-ANU
Camberra - Australia

Fabio Cozman
POLI-University of Sao Paulo
Sao Paulo - Brazil

Abstract

This paper presents a short survey of the research we have carried out on planning under uncertainty where we consider different forms of imprecision on the probability transition functions. Our main results are on efficient solutions for Markov Decision Process with Imprecise Transition Probabilities (MDP-IPs), a generalization of a Markov Decision Process where the imprecise probabilities are given in terms of credal sets. Noting that the key computational bottleneck in the solution of MDP-IPs is the need to repeatedly solve an optimization problem. Our results show how to target approximation techniques to drastically reduce the computational overhead of the optimization solver while producing bounded, approximately optimal solutions.

Introduction

Markov Decision Processes (MDP) (Puterman 1994) have become the *de facto* standard model for decision-theoretic planning problems and a great deal of research in recent years has aimed to propose efficient solutions to tackle large and more realistic problems. Formally, an MDP is defined by the tuple $\mathcal{M} = \langle S, A, P, R, T, \gamma \rangle$, where S is a finite set of fully observable states; A is a finite set of actions; $P(s'|s,a)$ is the conditional probability of reaching state $s' \in S$ when action $a \in A$ is taken from state $s \in S$; $R : S \times A \rightarrow \mathbb{R}$ is a fixed reward function associated with every state and action; T is the time horizon (number of stages-to-go) for decision-making; and $\gamma = [0, 1)$ is a *discount factor*. An important research topic in this area is how to exploit structure in order to compactly represent and efficiently solve factored MDPs (Boutilier, Hanks, and Dean 1999; Hoey et al. 1999; St-Aubin, Hoey, and Boutilier 2000; Guestrin et al. 2003). In many MDPs, it is often natural to think of the state as an assignment to multiple state variables and a transition function that compactly specifies the probabilistic dependence of variables in the next state on a subset of variables in the current state. Such an approach naturally leads us to define a *Factored MDP* (Boutilier, Hanks, and Dean 1999), where $S = \{\vec{x}\}$. Here, $\vec{x} = (x_1, \ldots, x_n)$ where each state variable $x_i \in \{0, 1\}$.

The reward can simply be specified as $R(\vec{x}, a)$. The transition probabilities in a factored MDP are encoded using *Dynamic Bayesian Networks (DBNs)* (Dean and Kanazawa 1990). A DBN is a *directed acyclic graph (DAG)* with two layers: one layer represents the variables in the current state and the other layer represents the next state (Figure 3a). Nodes x_i and x'_i refer to the respective current and next state variables. The connection between these two layers defines the dependences between state variables w.r.t. the execution of an action $a \in A$. Directed edges are allowed *from* nodes in the first layer *into* the second layer, and also between nodes in the second layer (these latter edges are termed *synchronic arcs*). We denote by $pa_a(x'_i)$ the parents of x'_i in the graph for action a and $P(x'_i|pa_a(x'_i), a)$, the conditional probability table (CPT). The graph encodes the standard Bayes net conditional independence assumption that a variable x'_i is conditionally independent of its nondescendants given its parents, which incidentally for a DBN also encodes the Markov assumption (the current state is independent of the history given the previous state). The use of a DBN leads to the following factorization of transition probabilities:

$$P(\vec{x}'|\vec{x}, a) = \prod_{i=1}^{n} P(x'_i|pa_a(x'_i), a). \quad (1)$$

However, in many real-world problems, it is simply impossible to obtain a precise representation of the transition probabilities in an MDP. This may occur for many reasons, including (a) imprecise or conflicting elicitations from experts, (b) insufficient data from which to estimate reliable precise transition models, or (c) non-stationary transition probabilities due to insufficient state information.

For example, in an automated navigation system, it can be difficult to estimate the probability of reaching some location after a move. The probabilities may change throughout time due the environment conditions (such weather and road conditions) and can make the navigation more difficult and subject to failures. In general, it is hard to accurately model all these changes since they can have many external dependencies. In this case, it is better to have a policy optimized over a range of possible probabilities in order to be robust against transition uncertainty.

To accommodate optimal models of sequential decision-making in the presence of strict uncertainty over the tran-

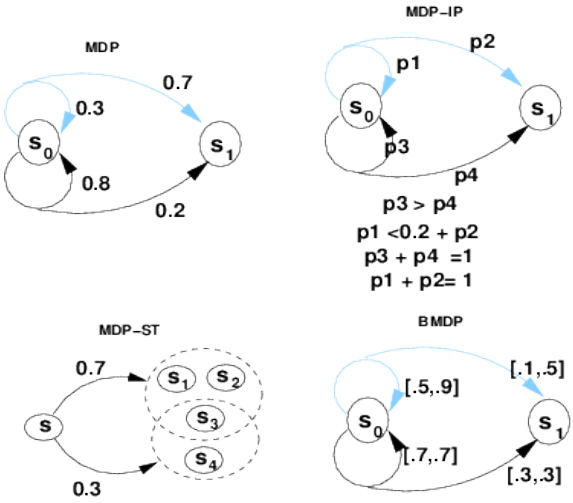


Figure 1: Examples of MDP, MDP-IP, MDP-ST and BMDP.

sition model, the MDP with imprecise transition probabilities (MDP-IP) was introduced (Satia and Lave Jr. 1970; White III and El-Deib 1994) where the imprecise probabilities are represented by probabilistic parameters p_i and a set of constraints over them which define a credal set. While the MDP-IP poses a robust framework for the real-world application of decision-theoretic planning, its general solution requires the use of computationally expensive optimization routines that are extremely time-consuming in practice.

A particular sub-classes of MDP-IP is the *Bounded-parameter Markov Decision Process (BMDP)* (Givan, Leach, and Dean 2000), where the probabilities are specified by intervals. Givan proposes the Interval Value Iteration algorithm (Givan, Leach, and Dean 2000) that can find an optimal policy without requiring expensive optimization techniques. Recent solutions to BMDPs include extensions of real-time dynamic programming (RTDP) (Buffet and Aberdeen 2005) and LAO* (Cui et al. 2006; Yin, Wang, and Gu 2007) that search for the best policy under the worst model. However, a problem with general linear constraints over the probability parameters, for example $\{0.3 \leq p_j \leq 0.5, p_j \geq p_i\}$ can not be solved by these solutions. Another sub-class of MDP-IPs is the *Markov Decision Process with Set-valued Transitions (MDP-ST)* (Trevizan, Cozman, and de Barros 2007), where probability distributions are given over finite sets of states. Examples of MDP-IP, BMDP and MDP-ST are given in Figure 1.

In Figure 2 we show the relationship among different types of planning under uncertainty which includes as special cases the deterministic and nondeterministic planning problems. Notice that BMDP and MDP-ST do not have the same representational power, i.e., some MDP-ST problems can not be reduced to BMDP, and vice versa. Note also that since BMDPs and MDP-STs are special cases of MDP-IPs, we can represent them as MDP-IPs, thus the algorithms for MDP-IPs clearly apply to both BMDPs and MDP-STs.

To address the computational deficiency of solutions for MDP-IPs, first we extended the factored MDP model by replacing the usual Dynamic Bayesian Network (DBN) (Dean and Kanazawa 1990) used in factored MDPs with Dynamic Credal Network (DCNs) (Delgado et al. 2009; Delgado, Sanner, and de Barros 2011) to support compact factored structure in the imprecise transition model of factored MDP-IPs. Second, we have proposed efficient, scalable algorithms for solving these factored MDP-IPs based on two different approaches: dynamic programming (Delgado, Sanner, and de Barros 2011) and multilinear programming (Delgado et al. 2011).

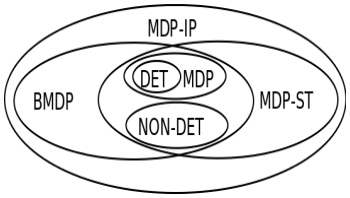


Figure 2: Relationship among MDP-IP and its sub-classes.

MDPs with Imprecise Transitions

An *MDP with imprecise transition probabilities (MDP-IP)* is simply an extension of the MDP where the transition probabilities can be imprecisely specified. That is, instead of a probability measure $P(\cdot|s,a)$ over the state space S, we have a *set* of probability measures. that is referred to as a *credal set* (Cozman 2000).

Definition 1. Transition credal set. *A credal set containing conditional distributions over the next state s', given a state s and an action a, is referred to as a* transition credal sets *(Cozman 2000) and denoted by $K(s'|s,a)$. Thus, we have $P(\cdot|s,a) \in K(\cdot|s,a)$ to define imprecisely specified transition probabilities.*

We assume that all credal sets are closed and convex, an assumption that is often used in the literature, and that encompasses most practical applications (Walley 1991). We further assume stationarity for the transition credal sets $K(s'|s,a)$; that is, they do not depend on the stage t. While $K(s'|s,a)$ is non-stationary, we note that this does not require $P(s'|s,a)$ to be stationary in an MDP-IP: distributions $P(s'|s,a)$ may be selected from the corresponding credal sets in a time-dependent manner (Nilim and El Ghaoui 2005).

Formally, an MDP-IP is defined by $\mathcal{M}_{IP} = (S, A, K, R, T, \gamma)$. This definition is identical to the MDP $\mathcal{M}$, except that the transition distribution P is replaced with a transition credal set K. We will represent K implicitly as the set of transition probabilities consistent with a set of side linear inequality constraints C over the probability parameters.

There are several optimization criteria that can be used to define the value of a policy in an MDP-IP. In the context of the discounted infinite horizon setting focused on this work, there is always a deterministic stationary policy that is *maximin* optimal (Satia and Lave Jr. 1970) (i.e., no other policy could achieve greater value under the assumption that Nature's selects $P(s'|s,a)$ adversarially to mini-

mize value); moreover, given the assumption that A is finite and the credal set K is closed, this policy induces an optimal value function that is the unique fixed-point solution of

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

There are various algorithms for solving enumerated state MDP-IPs based on dynamic programming (Satia and Lave Jr. 1970; White III and El-Deib 1994). In this work, we build on a value iteration solution to MDP-IPs (Satia and Lave Jr. 1970):

$$V^t(s) = \max_{a \in A} \min_{P \in K} \left\{ R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{t-1}(s') \right\} \quad (3)$$

Value iteration for MDP-IPs is the same as for MDPs except that now for *every* state s, we optimize our action choice $a \in A$ w.r.t. the *worst-case* distribution $P \in K$ that minimizes the future expected value. Thus we ensure that the resulting value function and policy are robust to the worst outcome that Nature could choose in light of the future value $V^{t-1}(s')$ that we expect to achieve.

The Bellman equation can be also solved through a multilinear program (Shirota et al. 2007):

$$\begin{aligned} \min_{V^*, P} \quad & : \quad \sum_s V^*(s) \quad (4) \\ \text{s.t.} \quad & : \quad V^*(s) \geq R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s'), \\ & \qquad \forall s \in S, a \in A, P(s'|s,a) \in K(s'|s,a). \end{aligned}$$

Notice that the constraints force $V^*(s)$ to be greater than or equal to $\max_{a \in A} \min_{P \in K} \{R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s')\}$, considering all $a \in A$, and then minimizing $\sum_s V^*(s)$ enforces that the minimal $V^*(s)$ is obtained.

Factored MDP-IPs

The definitions of MDP-IP given in the previous section models an enumerated MDP-IP (also called flat MDP-IP), where states are seen as black boxes.

We have extended the factored MDP representation (Boutilier, Hanks, and Dean 1999) to compactly represent MDP-IPs. This requires modifying the DBN transition representation to account for uncertainty over the exact transition probabilities.

Like the previous definition of an enumerated state MDP-IP, the set of all legal transition distributions for a factored MDP-IP is defined as a *credal set* K. The challenge then was to specify such transition credal sets in a factored manner that is itself compact. For this, we have proposed to use *dynamic credal networks (DCNs)*, a special case of credal networks (Cozman 2000; 2005), as an appropriate language to express factored transition credal sets.

Definition 2. Factored transition credal set. *A credal set containing conditional distributions over the values of a variable x_i, given the values of $pa_a(x_i)$ (the parents of x_i in the graph for action a), is referred to as a* factored transition credal set *and denoted by $K_a(x_i|pa_a(x_i))$.*

Definition 3. Dynamic credal network. *A Dynamic credal network (DCN) is a generalization of a DBN. Different from the definition of a DBN, in a DCN each variable x_i is associated with factored transition credal sets $K_a(x_i|pa_a(x_i))$ for each value of $pa_a(x_i)$. We assume that a DCN represents a* joint credal set *(Cozman 2005; 2000) over all of its variables consisting of all distributions that satisfy the factorization in Equation (1), where each CPT distribution $P(x'_i|pa_a(x'_i), a)$ is an element of the transition credal set $K_a(x'_i|pa_a(x'_i))$ associated with the DCN, i.e. $P(x'_i|pa_a(x'_i), a) \in K_a(x'_i|pa_a(x'_i))$.*

A DCN example is shown in Figure 3a. For each variable x'_i in a DCN, we have a *conditional probability table (CPT)* with imprecise probabilities. If we examine the CPTs in Figure 3b, we note that entries are specified by probability parameters p_{ij} (i for variable x'_i and j for the jth parameter in the CPT for x'_i). Furthermore, we note that we have a set of side linear constraints on these p_{ij} (shown in the boxes below the CPT, collectively call this constraint set C). We use $\vec{p}$ to denote a vector containing all parameter values that are free to vary within the given credal sets (i.e., that satisfy the probability constraints C of the DCN).

We note that the joint transition probability may be nonlinear in the probability parameters $\vec{p}$. However, we explicitly introduce the following restriction to prevent exponents exceeding 1: a parameter p_{ij} may only appear in the CPT for x'_i. This restriction prevents the multiplication of p_{ij} by itself when CPTs for each x'_i are multiplied together to determine the joint transition distribution in the DCN. This subset of nonlinear expressions, where the exponent of each p_{ij} is either 0 or 1, is referred to as a *multilinear* expression. To see the multilinearity of the transition probability in Figure 3, we observe $P(x'_1 = 1, x'_2 = 1 | x_1 = 1, x_2 = 1, notreboot) = p_{11} p_{21}$.

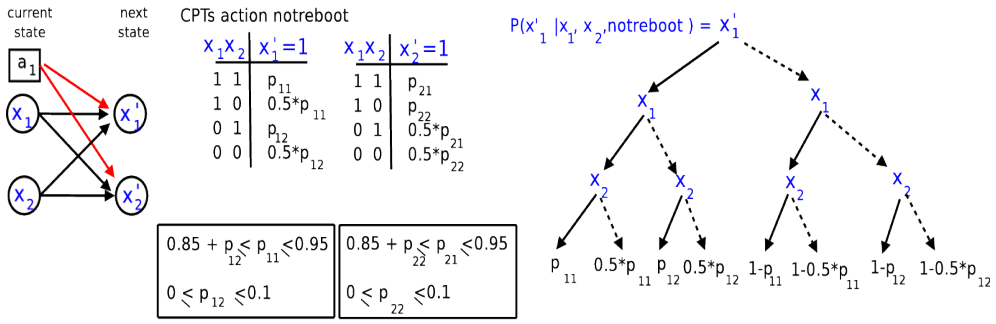
Exact Solution for Factored MDP-IPs

SPUDD-IP

SPUDD (Hoey et al. 1999) is an efficient factored version of Value Iteration for MDPs that represents CPTs, rewards and value functions as *algebraic decision diagrams (ADDs)* (Bahar et al. 1993). Inspired on that we have proposed *SPUDD-IP* (Delgado, Sanner, and de Barros 2011) to solve factored MDP-IPs.

ADDs compactly represent *context-specific independence* (CSI) (Boutilier et al. 1996) that are not evident in the DBNs. In order to compactly represent a CSI and shared function structure in the CPTs for an MDP-IP, we have proposed a novel extension of ADDs called *parameterized ADDs (PADDs)* (Delgado, Sanner, and de Barros 2011) since the leaves are parameterized expressions as shown in Figure 3c. PADDs do not only allow us to compactly represent the CPTs for factored MDP-IPs, but they also enable efficient computations for factored MDP-IP value iteration operations.

We begin by expressing MDP-IP value iteration from (3) in the following factored form using the transition represen-



a) b) c)

Figure 3: a) DCN for action a. b) Conditional probability table for the state variables $x'_1 = 1$ and $x'_2 = 1$ and the constraints related to the probabilities. c) The Parameterized ADD representation of $P(x'_1|x_1, x_2, a)$. Solid lines indicate the true (1) branch of a variable test and dashed lines indicate the false (0) branch.

tation of (1) and operations on decision diagrams:[1]

$$V^t_{DD}(\vec{x}) = \max_{a \in A} \left\{ R_{DD}(\vec{x}, a) \oplus \gamma \min_{\vec{p}} \boxed{\sum_{\vec{x}'} \bigotimes_{i=1}^{n} P_{DD}(x'_i|pa_a(x'_i), a) V^{t-1}_{DD}(\vec{x}')} \right\} \quad (5)$$

Because the transition CPTs in the MDP-IP DCN contain parameters $\vec{p}$, these CPTs must be represented in decision diagram format as PADDs ($P_{DD}(x'_i|pa_a(x'_i), a) \in K_a(x'_i|pa_a(x'_i))$). The reward $R_{DD}(\vec{x}, a)$ can be represented as an ADD since it contains only constants (for the purpose of operations, recall that ADDs are special cases of PADDs). Although it may appear that the form of $V^t_{DD}(\vec{x})$ is a PADD, we note that the parameters $\vec{p}$ are "minimized"-out w.r.t. the side constraints on $\vec{p}$ during the $\min_{\vec{p}} \square$ operation in (5) ($\min_{\vec{p}} \square$ is the *MinParameterOut* operation on PADDs, that performs the minimization over the parameters by calling a nonlinear solver for each leaf and returns an ADD). Thus the resulting $V^t_{DD}(\vec{x})$ computed from the $\max_{a \in A}$ has constant leaves and can be expressed as the ADD special case of PADDs.

To explain the efficient evaluation of (5) in more detail, we can exploit the variable elimination algorithm (Zhang and Poole 1994) in the marginalization over all next states $\sum_{\vec{x}'}$. For example, if x'_1 is not dependent on any other x'_i for $i \neq 1$, we can "push" the sum over x'_1 inwards to obtain:

$$V^t_{DD}(\vec{x}) = \max_{a \in A} \left\{ R_{DD}(\vec{x}, a) \oplus \gamma \min_{\vec{p}} \boxed{\sum_{x'_i (i \neq 1)} \bigotimes_{i=1 (i \neq 1)}^{n} P_{DD}(x'_i|pa_a(x'_i), a) \sum_{x'_1} P_{DD}(x'_1|pa_a(x'_1), a) V^{t-1}_{DD}(\vec{x}')} \right\} \quad (6)$$

Then we can continue with x'_2, multiplying this result by the $P_{DD}(x'_2|pa_a(x'_2), a)$, summing out over x'_2, and repeating for all x'_i to compute $\square$. After this $\square$ does not contain anymore the variables x'_i, but only the variables x_i.

The SPUDD-IP value iteration solution to factored MDP-IPs returns a robust optimal policy and often yields an improvement over flat value iteration. Figure 4 shows that for large state spaces (2^{12} to 2^{22}) flat Value Iteration does not return a solution due to time and space limitations, while SPUDD-IP can solve the largest problems.

[1] We use DD for the functions represented by ADDs or PADDs.

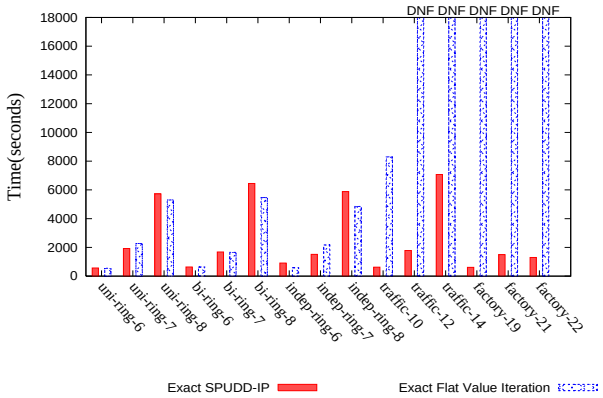


Figure 4: Time performance comparison for TRAFFIC, SYSADMIN and FACTORY problems using SPUDD-IP and Flat Value Iteration. The name includes the number of variables in each problem, so the corresponding number of states is $2^{\#variables}$.

Approximate Solutions for Factored MDP-IPs

As the number of state variables in a problem grows larger SPUDD-IP also becomes inefficient. Thus in this section we describe three approximate solutions for MDP-IP: APRICODD-IP (Delgado, Sanner, and de Barros 2011) and Objective-IP (Delgado, Sanner, and de Barros 2011), two approximate versions of Value Iteration for MDP-IP; and AMP (Delgado et al. 2011) an approximate Multilinear Programming algorithm for MDP-IP.

Approximate Value Iteration

Approximate value iteration (AVI) is one way to trade off time and space with error by approximating the value function after each iteration.

APRICODD-IP algorithm is an extension of the APRICODD algorithm (St-Aubin, Hoey, and Boutilier 2000) that

provides an efficient way of *approximating the ADD value representation* for a factored MDP, reducing its size and thus reducing computation time per iteration. This is done by a method that has two inputs: (1) a value function represented as an ADD and (2) an approximation error to merge the leaves. The output is a new ADD with the merged leaves. The algorithm first collects all leaves of the ADD and determines which can be merged to form new values without approximating more than *error*. The old values are then replaced with these new values creating a new (minimally reduced) ADD that represents the approximated value function. This approach immediately generalizes to MDP-IPs since the value function V_{DD}^t is also an ADD.

However, in solving (factored) MDP-IPs, the time is dictated less by the size of the value function ADD and more by the number of calls to the multilinear optimizer. Thus we also proposed a new approximate algorithm, Objective-IP, that attempts to make less calls to the solver by *approximating the objectives* (the mim of the optimization call) in an attempt to avoid calling the solver altogether. Different from APRICODD-IP, Objective-IP approximates the leaves of the PADD just prior to carrying out the multilinear optimization. The approximation method takes as input a PADD and the maximum *error* and returns a new PADD with approximated leaves using the upper and lower bounds of the parameters p_i. Note that each leave is approximated independently, this can be done since each leaf corresponds to a different state (or set of states) and the system can only be in one state at a time. Furthermore, we can guarantee that no objective pruning at the leaves of the PADD incurs more than *error* after the multilinear optimization is performed.

In order to evaluate the policy returned by our approximate solutions, we compute the True Approximation Error (TAE) given by:

$$max_{\vec{x}}|V^*(\vec{x}) - V_{approx}(\vec{x})| \tag{7}$$

where $V_{approx}(\vec{x})$ is the value returned by the approximate solutions and $V^*(\vec{x})$ is the optimal value computed by SPUDD-IP.

In Figure 5 we show a comparison of the True Approximation Error (TAE) vs. running times for the bi-ring-6 problem of an imprecise version of the SysAdmin domain (Guestrin et al. 2003). The results echo one conclusion: Objective-IP consistently takes less time than APRICODD-IP to achieve the same approximation error and *up to one order of magnitude less time* than APRICODD-IP (Delgado, Sanner, and de Barros 2011). This time reduction can be explained by the decreased number of calls to the multilinear solver.

Approximate Multilinear Programming

Another approximate solution for Factored MDP-IPs we have proposed is the Approximate Multilinear Programming (AMP) (Delgado et al. 2011). We extend previous work (Guestrin et al. 2003) that obtain efficient approximate linear programming solutions for Factored MDPs. In this approximate solution we use the approximate value function denoted by $\widehat{V}(\vec{x})$. Given $\vec{x} \in \mathcal{S}$ and a set of basis functions $H = \{h_1, ..., h_k\}$, $V^*(\vec{x})$ is approximated using a linear combination:

$$\widehat{V}(\vec{x}) = \sum_{j=1}^{k} w_j h_j(\vec{x}). \tag{8}$$

We can use this approximate value function and replace it in the multilinear formulation (Problem (4)) of an MDP-IP so as to obtain the factored multilinear programming problem:

$$\min_{w,P} \quad \sum_{\vec{x}} \sum_{i=0}^{k} w_i h_i(\vec{x}) \tag{9}$$

$$\text{subject to} \quad : \quad \sum_{i=0}^{k} w_i h_i(\vec{x}) \geq R(\vec{x}, a) +$$

$$\gamma \sum_{\vec{x}' \in \mathcal{S}} P(\vec{x}'|\vec{x}, a) \sum_{i=0}^{k} w_i h_i(\vec{x}'), \forall \vec{x} \in S, a \in A$$

$$P(x_i'|pa(X_i'), a) \in K_a(X_i'|pa(X_i')),$$

$$P(\vec{x}'|\vec{x}, a) = \prod_i P(x_i'|pa(X_i'), a).$$

To solve this optimization problem, we exploit the Factored MDP-IP 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.

In Figure 5 we compare the three approximate solution methods, APRICODD-IP and Objective-IP and AMP. We used *simple* basis functions (one for each variable in the problem description) and *pairwise* basis functions (one for each pair of variables that have a common child variable in the DCN).

When it does finish within a limit of ten hours, AMP takes only a few seconds to produce an approximate solution for each problem (except for the FACTORY domain for which it did not return a solution). Comparing the algorithms in terms of their true approximation error, we observe that in the bi-ring-6 problem, AMP with *pair* basis functions outperforms APRICODD-IP and obtains a solution 2-3$\times$ larger than the error of Objective-IP, but in significantly less time. This experiment and the results obtained in other domains lead us to conclude that Objective-IP consistently gives an error at least 2-3$\times$ lower than AMP and sometimes runs as fast as the AMP solution, while in other cases running slower. However, the bottleneck of the AMP solution is to define appropriate basis functions (Guestrin et al. 2003).

Concluding Remarks

In this work we make a short survey on our contribution for Markov Decision Processes with Imprecise Probabilities (MDP-IPs), a class of models that adds considerable flexibility and realism to probabilistic planning allowing the representation of imprecise transition probabilities. 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, which can reveal the structure of an application domain and

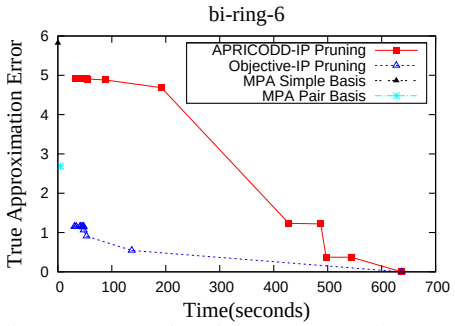


Figure 5: True Approximation Error vs. time required for APRICODD-IP, Objective-IP and MPA with simple basis and pairwise basis functions for SYSADMIN problem with bidirectional-ring topology.

allows for the construction of efficient solutions. The proposed solution, exact or approximate were of two types: based on dynamic programming and based on multilinear programming. The first extends the factored approaches for MDPs, SPUDD and APRICODD, plus a new way of approximation that avoids calling the optimization solver. The second extends the approximate linear programming solution for MDPs, to the Approximate Multilinear Programming for MDP-IPs, that can be very efficient when we have appropriate basis functions.

We are currently working on an asynchronous dynamic programming, named RTDP-IP for MDP-IPs. The main challenges of this solution are: (i) how to sample the next state in a trial? (ii) how to ensure convergence of an asynchronous dynamic programming solution, having a range of possible probabilities in the state transition matrix? Another work we are currently working is how to extract an MDP-IP transition function from data.

References

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic Decision Diagrams and their Applications. In *Proceedings of ICCAD*, 188–191. Los Alamitos, CA, USA: IEEE Computer Society Press.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific Independence in Bayesian Networks. In *Proc. 12th UAI*, 115–123.

Boutilier, C.; Hanks, S.; and Dean, T. 1999. Decision-theoretic Planning: Structural Assumptions and Computational Leverage. *JAIR* 11:1–94.

Buffet, O., and Aberdeen, D. 2005. Robust Planning with LRTDP. In *Proc. of the IJCAI*, 1214–1219.

Cozman, F. G. 2000. Credal Networks. *Artificial Intelligence* 120:199–233.

Cozman, F. G. 2005. Graphical Models for Imprecise Probabilities. *International Journal of Approximate Reasoning* 39(2-3):167–184.

Cui, S.; Sun, J.; Yin, M.; and Lu, S. 2006. Solving Uncertain Markov Decision Problems: An Interval-Based Method. In *ICNC (2)*, 948–957.

Dean, T., and Kanazawa, K. 1990. A Model for Reasoning about Persistence and Causation. *Comput. Intell.* 5(3):142–150.

Delgado, K. V.; de Barros, L. N.; Cozman, F. G.; and Shirota, R. 2009. Representing and Solving Factored Markov Decision Processes with Imprecise Probabilities. In *6th ISIPTA*.

Delgado, K. V.; de Barros, L. N.; Cozman, F. G.; and Sanner, S. 2011. Using mathematical programming to solve factored markov decision processes with imprecise probabilities. *Int. J. Approx. Reasoning* 52(7):1000–1017.

Delgado, K. V.; Sanner, S.; and de Barros, L. N. 2011. Efficient solutions to factored mdps with imprecise transition probabilities. *Artif. Intell.* 175(9-10):1498–1527.

Givan, R.; Leach, S.; and Dean, T. 2000. Bounded-parameter Markov Decision Processes. *Artificial Intelligence* 122:71–109(39).

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient Solution Algorithms for Factored MDPs. *JAIR* 19:399–468.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic Planning using Decision Diagrams. In *Fifteenth Conference on Uncertainty in Artificial Intelligence*, 279–288. Morgan Kaufmann.

Nilim, A., and El Ghaoui, L. 2005. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Oper. Res.* 53(5):780–798.

Puterman, M. L. 1994. *Markov Decision Processes*. Wiley Series in Probability and Mathematical Statistics. New York: John Wiley and Sons.

Satia, J. K., and Lave Jr., R. E. 1970. Markovian Decision Processes with Uncertain Transition Probabilities. *Operations Research* 21:728–740.

Shirota, R.; Cozman, F. G.; Trevizan, F. W.; de Campos, C. P.; and de Barros, L. N. 2007. Multilinear and Integer Programming for Markov Decision Processes with Imprecise Probabilities. In *5th ISIPTA*, 395–404.

St-Aubin, R.; Hoey, J.; and Boutilier, C. 2000. APRICODD: Approximate Policy Construction using Decision Diagrams. In *Proceedings NIPS*, 1089–1095. MIT Press.

Trevizan, F. W.; Cozman, F. G.; and de Barros, L. N. 2007. Planning under Risk and Knightian Uncertainty. In *IJCAI*, 2023–2028.

Walley, P. 1991. *Statistical Reasoning with Imprecise Probabilities*. London: Chapman and Hall.

White III, C. C., and El-Deib, H. K. 1994. Markov Decision Processes with Imprecise Transition Probabilities. *Operations Research* 42(4):739–749.

Yin, M.; Wang, J.; and Gu, W. 2007. Solving Planning Under Uncertainty: Quantitative and Qualitative Approach. In *IFSA (2)*, 612–620.

Zhang, N. L., and Poole, D. 1994. A Simple Approach to Bayesian Network Computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, 171–178.