Compact Representations of Markov Decision Processes and Their Application to Printer Management

João Vitor Torres[1], **Fabio Gagliardi Cozman**[1], **André Rodrigues**[2]

[1]Escola Politécnica da Universidade de São Paulo (POLI USP)
São Paulo, SP - Brazil

[2]HP-Brazil R&D, Porto Alegre
RS - Brazil
j_v_torres@yahoo.com.br, fgcozman@usp.br, andre.rodrigues@hp.com

Abstract. *Many planning problems can be framed as Markov decision processes (MDPs). In this paper we discuss situations where regularities in states and variables lead to compact MDPs, particularly when variables have many categories and strong interrelation. We develop techniques that generate optimal policies by exploiting regularities in MDPs. We illustrate these ideas with a real problem on management of printing clusters.*

1. Introduction

Situations that require sequential decision making under uncertainty are often modeled by Markov Decision Processes (MDPs) [Boutilier et al. 1999, Puterman 1994]. However, it is not always easy to specify all transition probabilities when building an MDP; moreover, it is not easy to generate optimal policies in large MDPs [Littman et al. 1995]. In this paper we propose techniques that aim at simplifying the construction and manipulation of MDPs.

The last decade has seen significant discussion of "factored" descriptions, often based on Bayesian networks (BNs) [Boutilier et al. 1999, Guestrin et al. 2003]. To further simplify the construction of MDPs, there has been interest in models that explore hierarchical and logical relations among states and variables [Glesner and Koller 1995, D. Koller 1998, Puterman 1994, R. Sharma 2005]; also, some design patterns such as "NoisyOR" or "decision-tree distributions" have been detected [C. Boutilier and Koller 1996, Glesner and Koller 1995]. Our experience in building MDPs for real problems is that there are further regularities that have even more practical significance, and that have not received due attention so far.

We offer two contributions in this paper. First, we propose a methodology that generates compact factored representations by exploiting common features of practical MDPs (Section 2). We have found these guidelines to be quite effective in our practical work, particularly when random variables have many categories and display strong interrelation. Our second contribution is a set of techniques for generation of optimal policies, where we exploit regularities in the representation of MDPs (Section 3). We also describe an application of these guidelines to a real problem in industry; in fact, the present work was motivated by our frustration with existing techniques in dealing with this real problem (Section 4).

2. MDPs and Compact Representations

A (completely observable and stationary) Markov Decision Process consists of a set of states S, a set of actions A, a transition matrix T for each action a in A, a reward function R and a cost function C [Puterman 1994]. The *finite* set S contains all possible states (each is a complete description of the system), and the agent must find a *policy* that prescribes an action for each state. A transition matrix $T(s, s', a)$ specifies the probability of moving from s to s' after action a. Finally, a reward function $R(s)$ associates a real value with each state and a cost function $C(s, a)$ associates a real value with each pair state/action. There exists algorithms that can find an optimal policy and determine the value of each state for this policy, according to several criteria (finite horizon, infinite horizon with discount, and others); two of these algorithms are *value iteration* and *policy iteration* [Bellman 1957, Boutilier et al. 1999].

The first step in constructing an MDP is to represent the system state by means of variables that capture its relevant characteristics. Let S_P be the system state at the present point in time and let S_F be the system state at the next point in time. We must specify the probability values $P(S_F = s_i | S_P = s_l, A = a_j)$. A compact representation for these probabilities can be produced using a Bayesian network (BN) per action [Boutilier et al. 1999, Guestrin et al. 2003]. We use such factored representations in the remainder of the paper.

Example 1 *Consider a printer that is modeled by two binary variables: printing queue (q) and job size (js) and action set $A = \{Print, Ignore\}$. The printer state is given by (q, js); there are 4 states. We can write $S_P = (q_P, js_P)$ and $S_F = (q_F, js_F)$. If the action is $Print$, suppose q_F depends on q_P and js_P, but js_F does not depend on any previous state (Figure 1.a shows the Bayesian network).*

The structure of the BNs implies probabilistic independences (following from the Markov condition for Bayesian networks; that is, a variable is independent of its nondescendants given its parents [Pearl 1988]).

Example 2 *In Example 1, we write the state transition probability for action $Print$ as: $P(S_F = (q_F, js_F)|S_P = (q_P, js_P)) = p(q_F|q_P, js_P) \times p(js_F)$.*

Factorization techniques are of limited help when variables have *many* categories, as conditional transition matrices may become too large for explicit specification. In practical systems we often observe regularities that go beyond independence among variables. We now introduce a few "design patterns" that have significant practical impact.

1 Suppose that categories of a variable X are ordered (for example, X is integer-valued) and any change in X occurs only by a limited increase or decrease (thus many transitions have probability zero). Then X is said to be of *limited variation*.

Example 3 *Consider the printing queue variable (q) in our printer model. Assume this variable has ten categories varying from 1 to 10. It is reasonable to assume that the queue level can vary just two levels at each transition (up or down).*

2 Suppose that $P(X_F = x + t|X_P = x)$, the probability that X makes a transition from a level x to a level $x + t$, does not depend on x. We assume this property to hold for negative and positive integer values of t, with the exception of extremities in the variable scale. Then X is said to be of *homogeneous variation*.

Example 4 *Consider again the printing queue variable (q) in our printer model. The probability of the printing queue increasing one level is in fact independent of the actual level; for instance, $P(q_F = 2|q_P = 1) = P(q_F = 3|q_P = 2)$ (however, when $q_P = 10$ then $P(q_F = 11|q_P = 10) = 0$ as this is an extreme situation).*

3 Suppose X has conditional probabilities that relate proportionaly to conditioning variables, and this relation depends on the state of the system. This property is called *proportional variation*. Note that to use this property it is necessary that *homogeneous variation* holds as well.

Example 5 *Now consider js has five categories in our printer model. The probability of the queue increasing one level is directly proportional to the job size: $p(q_F = q_P + 1|q_P, js_P) = k \times js_P$, where k is a normalization constant.*

The use of these properties together allow conditional probabilities to be specified in a very compact way. Such properties are rather valuable and should be included in a specification language, for instance PPDDL [Littman and Younes 2004], in the future.

As a final point, we note that in the presence of large space states, reward/cost functions require compact representation as well, often achieved in the literature through additivity assumptions [Boutilier et al. 1999].

Example 6 *We can write that the cost of executing $Print$ is proportional to queue level and proportional to job size: $C((q_P, js_P), Print) = k \times q_P \times js_P$.*

3. Exploiting Regularities in Factored MDPs Solution

Traditional algorithms such as *value iteration* and *policy iteration* are not efficient for large-scale MDPs [Boutilier et al. 1999, Littman et al. 1995]. We now propose two techniques that reduce the computational burden in structured MDPs.

3.1. Offline sub-division of states

Our MDP construction method brings to the surface the level of connection in the state space; usually pieces of the state space can often be solved independently. This in itself is not a new observation [Puterman 1994]. However, what is important is to notice that even if an MDP cannot be divided into independent pieces, it may have limited state transitions that can be exploited. That is, one can identify a subset of the state space that is "closed" in the sense that an optimal policy for it can be found only processing its states. The solution of this fragment can then be used when processing the whole MDP. The idea is similar to dynamic programming; however we are not operating backwards with respect to transitions but rather operating piecewise with respect to states. An example should clarify the idea.

Example 7 *Consider a printer model with two variables: printing queue (q) and tray level (tr), each with 10 categories, and an actions set $A = \{Print, Ignore\}$. If the action is $Print$, then tr can only decrease or stay constant; if the action is $Ignore$, both q and tr can only decrease or stay constant. Note that tr can never increase; thus we can start with a reduced MDP where this variable is set to its lowest level. After solving this reduced MDP, we construct a new MDP in which tr is limited to its 2 lowest levels. This adds 10 new states and the solution for the 10 initial states does not change. Thus we need to solve just 10 states in this step. This process is repeated until the whole MDP is solved.*

Dividing a state space S into n fragments decreases the number of operations per iteration of value iteration to $O(|S|^2|A|/n+|S|^3/n^2)$. Besides, the required number of iterations to convergence is polynomial in $|S|$, thus the number of iterations in each fragment is lower than in the whole MDP.

3.2. Online reduction of state space

It is often difficult even to store an optimal policy for an MDP; in these cases one might resort to online solution of the Bellman equation, or, following the discussion in the previous section, online solution of fragments of the state space. An online solution is particularly difficult when variables have many categories. Suppose then that one produces an online solution by drastically reducing the size of the state space — we consider a strategy where each variable has its categories mapped into just two values. We now have to define state transition probabilities for the resulting binary variables. Our proposal is to employ the *original* variables in the construction of algebraic expressions for these probabilities, so that each transition leads to a different approximation based on the *original* probabilities. Doing so, the information loss caused by the transformation is mitigated (however note that probability distributions are no longer stationary). Again, the best way to understand our proposal is to examine an example.

Example 8 *In Example 7, transform variables q and tr into binary variables $\hat{q}$ and $\hat{tr}$. The new variable $\hat{q}$ indicates whether the queue is full ($\hat{q} = 1$ for $q = 10$) or not ($\hat{q} = 0$ for $q < 10$). The new variable $\hat{tr}$ indicates whether the tray level is empty ($\hat{tr} = 0$ for $tr = 1$) or not ($\hat{tr} = 1$ for $tr > 1$). We can write the probability of empty tray level as directly proportional to queue level q and inversely proportional to the tray level tr; that is, $P(\hat{tr}_F = 0 \mid \hat{tr}_P = 1) = k_1 q/tr$. Also, the probability of nonempty tray level is $P(\hat{tr}_F = 1 \mid \hat{tr}_P = 1) = 1/(1 + q/tr)$.*

Note first that the homogeneous variation property does not hold in this method; however the proportional variation property is still valid. Second, the solution is not stationary; the solution is really dependent on an online scheme. Finally, it is clear that the level of reduction in the state space size depends on the number of categories in each variable.

4. Modeling Printers with MDPs

To illustrate how the patterns and techniques discussed previously can be used in practice, we discuss here a real application of planning under uncertainty. The problem is to create policies for printers connected in clusters, continuously operating and receiving jobs from many users. Some printers may be faster, some may be slower; some may have features such as color printing, while others may excel in black-and-white printing. There may be additional constraints for the cluster, such as having all printers keep an approximately identical toner level (so as to minimize visits from personnel responsible for changing toner cartridges). Jobs are sent to the cluster of printers with a variety of characteristics, and to do so in a decentralized manner (no central router of jobs, to prevent that a single failure may disrupt the whole cluster).

In this setting, we consider each printer as an “agent” in a community, evaluating its own options and adopting its own actions with the objetive of maximize user satisfation in using the cluster. The routing strategy is based on auctions, as follows. First, the user chooses his preferences for the job and sends it from a PC to any of the printers in the

Table 1. System variables and number of categories per variable.

Variable	Number of categories	Interval for categories	Name (brief description)
Q	10	1, 2, ..., 10	queue (length of queue)
Tr	10	1, 2, ..., 10	tray (level of tray)
Tb	10	1, 2, ..., 10	black toner (level)
Tc	10	1, 2, ..., 10	color toner (level)
Js	5	1, 2, ..., 5	job size (in pages, less than 100)
Jc	2	0, 1	job color (color or b/w)
Ju	2	0, 1	job urgency (urgent or not)
V	5	1, 2, ..., 5	velocity (printer speed)

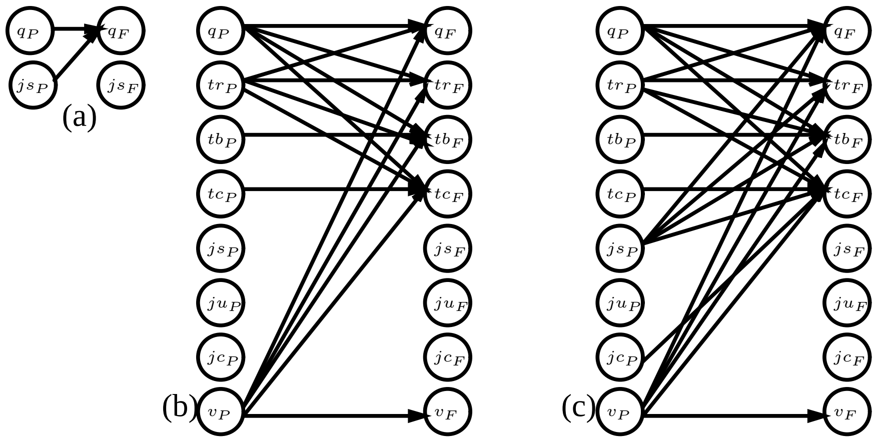


Figure 1. Bayesian Networks: (a) Example 2; (b) $Ignore$**; (b)** $Print$**.**

cluster. The receiving printer (P0) sends the job characteristics to other printers in the cluster. Then each printer assesses its state and chooses an action $Print$ or $Ignore$ using its own MDP. A printer sends its selected action and the value of its state to P0. The value sent is the "bid" for the job. Finally, P0 selects the best bid between printers with action $Print$ and sends the job to the winner printer. If all printers choose $Ignore$, then the best bit for this action is the winner.

This system allows many auctions to occur at the same time. Each printer follows an optimal "local" policy, thus deciding how to participate in the auction. Although this solution does not necessarily achieve a global optimum, it fulfills the desired objetives and it gives flexibility to the system, allowing entrance and departure of printers in the cluster. We now focus on the MDP used by each printer, as this is the focus of this paper.

4.1. State space, actions, transitions, and costs

A printer is modeled by eight variables, indicated in Table 1. These variables capture data from the printer (Q, Tr, Tb, Tc, V), the job (Js) and user preferences for the job (Jc, Ju). The space state formed by $(q, tr, tb, tc, js, jc, ju, v)$ contains 10^6 states ($10^6 = 10 \times 10 \times 10 \times 10 \times 5 \times 2 \times 2 \times 5$). Note that job size is assumed to be smaller than a hundred pages. Variables are denoted by capital letters while categories are not.

A printer has two actions: $A = \{Print, Ignore\}$; clearly the first means that the

Table 2. Conditional probabilities: q (top) and tr (bottom).

Print action			*Ignore* action		
	$tr_p = 1$	$tr_p \neq 1$		$tr_p = 1$	$tr_p \neq 1$
$\mathbf{q_F}$	$\mathbf{P(q_F \vert q_P, tr_P, js_P, v_P)}$		$\mathbf{q_F}$	$\mathbf{P(q_F \vert q_P, tr_P, v_P)}$	
$q_P + 2$	$k_0 js_P$	$k_1 js_P / v_P$			
$q_P + 1$	$4k_0 js_P$	$2k_1 js_P / v_P$			
q_P	$5k_0 / js_P$	$4k_1 js_P / v_P$	q_P	1	$5k_2 / v_P$
$q_P - 1$	0	$2k_1 v_P / js_P$	$q_P - 1$	0	$4k_2 v_P$
$q_P - 2$	0	$k_1 v_P / js_P$	$q_P - 2$	0	$k_2 v_P$

Print action		*Ignore* action	
$\mathbf{tr_F}$	$\mathbf{P(tr_F \vert q_P, tr_P, js_P, v_P)}$	$\mathbf{tr_F}$	$\mathbf{P(q_F \vert q_P, tr_P, v_P)}$
tr_P	$70k_3 / ((q_P + js_P) v_P)$	tr_P	$70k_4 / (q_P v_P)$
$tr_P - 1$	$2k_3 (q_P + js_P) v_P$	$tr_P - 1$	$2k_4 q_P v_P$
$tr_P - 2$	$k_3 (q_P + js_P) v_P$	$tr_P - 2$	$k_4 q_P v_P$

next job will be printed, while the second ignores the request. Transitions occur when a job arrives at the printer and the appropriate action is selected. We differentiate between variables "before" transitions and "after" transitions by appending subscripts P and F respectively.

Figures 1.b-c show BNs for *Print* and *Ignore* actions. The probabilities $P(S_F|S_P)$, for both actions, are then written in product form following these BNs. Terms $P(js_F)$, $P(jc_F)$ and $P(ju_F)$ are totaly independent of other variables and of the selected action. Their values are constant, equal to 0.2, 0.5 and 0.5 respectively. Likewise, the print velocity does not change; thus the term $P(v_F|v_P)$ is equal to 1 if $v_F = v_P$ and 0 otherwise. Terms $P(q_F|S_P)$, $P(tr_F|S_P)$, $P(tb_F|S_P)$ and $P(tc_F|S_P)$ are expressed through algebraic formulas of conditioning variables. This is possible because variables q, tr, tb and tc exhibit *proportional variation* and *homogeneous variation* properties. These formulas are different depending on the chosen action. In both actions, if $tr_P = 1$, then the printer has no paper and cannot print. Thus, the printer cannot consume its other resources such as toner, and its queue cannot decrease. These four variables also exhibit the *limited variation* property and this reduces the number of formulae. Printer resources do not change drastically, so they can increase or decrease their level by at most two units at each transition. Moreover, variables tr, tb and tc cannot make transitions to higher values. This happens because we are not interested in forecasting recharges of resources. Terms $P(q_F|S_P)$, $P(tr_F|S_P)$, $P(tb_F|S_P)$ and $P(tc_F|S_P)$ are summarized in Tables 2 and 3 respectively. These terms are expressed through formulas for the allowed changes in each variable. Table 3 (bottom) conveys one more important piece of information. If the action is *Print* and $tr_P \neq 1$, then the probability of change in tc depends on jc. Moreover, if the user wants a color job ($jc_P = 1$), then this probability depends on job size (js_P), otherwise js_P is irrelevant in this probability. Again, in the extreme values of q_P, tr_P, tb_P and tc_P, these formulas do not apply exactly (because a transition to a value out of the range of the variable is not allowed); the probability value must be accumulated in the probability of the closest allowed value for this variable.

The printer model does not have a reward function, because there is no fi-

Table 3. Conditional probabilities: tb (top) and tc (bottom).

Print action			*Ignore* action		
	$tr_p = 1$	$tr_p \neq 1$		$tr_p = 1$	$tr_p \neq 1$
$\mathbf{tb_F}$	$\mathbf{P(tb_F \mid q_P, tr_P, tb_P, js_P, v_P)}$		$\mathbf{tb_F}$	$\mathbf{P(tb_F \mid q_P, tr_P, tb_P, v_P)}$	
tb_P	1	$70k_5/((q_P + js_P)v_P)$	tb_P	1	$70k_6/(q_P v_P)$
$tb_P - 1$	0	$2k_5(q_P + js_P)v_P$	$tb_P - 1$	0	$2k_6 q_P v_P$
$tb_P - 2$	0	$k_5(q_P + js_P)v_P$	$tb_P - 2$	0	$k_6 q_P v_P$

Print action				*Ignore* action		
	tr_p				tr_p	
	$= 1$	$\neq 1$			$= 1$	$\neq 1$
		$jc_P = 0$	$jc_P = 1$			
$\mathbf{tc_F}$	$\mathbf{P(tc_F \mid q_P, tr_P, tc_P, js_P, jc_p, v_P)}$			$\mathbf{tc_F}$	$\mathbf{P(tc_F \mid q_P, tr_P, tc_P, v_P)}$	
tc_P	1	$\frac{70k_7}{q_P v_P}$	$\frac{70k_8}{(q_P + js_P)v_P}$	tc_P	1	$\frac{70k_9}{q_P v_P}$
$tc_P - 1$	0	$2k_7(q_P v_P)$	$2k_8(q_P + js_P)v_P$	$tc_P - 1$	0	$2k_9 q_P v_P$
$tc_P - 2$	0	$k_7(q_P v_P)$	$k_8(q_P + js_P)v_P$	$tc_P - 2$	0	$k_9 q_P v_P$

nal, objective state. However there are two cost functions, one for each action: $C(\mathit{Ignore}) = G_4 jc_P tc_P + (G_5 tr_P tb_P v_P)/(q_P js_P)$ and $C(\mathit{Print}) = (G_1 jc_P)/tc_P + (G_2 ju_P q_P js_P)/(tr_P tb_P v_P) + G_3/tb_P$, where G_i are constants. These functions "punish" idleness of resources.

4.2. Offline sub-division in the Printer MDP

We are interested in a policy that yields minimum cost as the printer operates continuously. We thus look for an infinite-horizon discounted policy [Puterman 1994]. Current algorithms would have significant difficulty dealing with 10^6 states; here we note that factored representations are just descriptive techniques and do not by themselves simplify the search for an optimal policy. However, when an MDP is constructed using common patterns in the domain, it becomes easy to identify regularities that can be exploited in the solution.

By analyzing the Printer MDP, it is easy to see that it can be divided in five independent MDPs — this is possible because printers does not change their own printing speeds. Thus, each new MDP has a fixed value for the variable v.

Second, we exploit the fact that variables tr, tb and tc can change their levels only downwards. Thus, we start solving the MDP with these variables fixed at their lowest level. When we do this, the state space contains only 200 states, a number that can be easily handled by existing algorithms. Then, after solving these small MDPs, we use the result to evaluate MDPs with a "higher" category in one of these three variables. This procedure is repeated until the original MDP is solved. Each new category adds 200 states in state space, but just these new states are solved in each step. This procedure is orders of magnitude faster than taking the whole MDP into a solution algorithm.

4.3. Online reduction in the Printer MDP

The first step to construct a "binarized" state space for our printer model is to transform the original variables. Let $\hat{x}$ be the new binary variable that relates to x, the original

Table 4. Conditional probabilities: $\hat{q}$ (top) and $\hat{tr}$ (bottom).

Print action.					*Ignore* action.				
	$\hat{q_P} = 0$		$\hat{q_P} = 1$			$\hat{q_P} = 0$		$\hat{q_P} = 1$	
	$\hat{tr_P}$		$\hat{tr_P}$			$\hat{tr_P}$		$\hat{tr_P}$	
	0	1	0	1		0	1	0	1
$\hat{\mathbf{q_F}}$	$\mathbf{P}(\hat{\mathbf{q_F}} \mid \hat{\mathbf{q_P}}, \hat{\mathbf{tr_P}}, \hat{\mathbf{js_p}}, \hat{\mathbf{v_P}})$				$\hat{\mathbf{q_F}}$	$\mathbf{P}(\hat{\mathbf{q_F}} \mid \hat{\mathbf{q_P}}, \hat{\mathbf{tr_P}}, \hat{\mathbf{v_P}})$			
0	k_0	k_1	0	$\frac{k_2 v_P}{js_P}$	0	1	1	0	$k_3 v_P$
1	$k_0 js_P q_P$	$\frac{k_1 js_P q_P}{v_P}$	1	k_2	1	0	0	1	k_3

Print action.			*Ignore* action.		
	$\hat{tr_P} = 0$	$\hat{tr_P} = 1$		$\hat{tr_P} = 0$	$\hat{tr_P} = 1$
$\hat{\mathbf{tr_F}}$	$\mathbf{P}(\hat{\mathbf{tr_F}} \mid \hat{\mathbf{q_P}}, \hat{\mathbf{tr_P}}, \hat{\mathbf{js_p}}, \hat{\mathbf{v_P}})$		$\hat{\mathbf{tr_F}}$	$\mathbf{P}(\hat{\mathbf{tr_F}} \mid \hat{\mathbf{q_P}}, \hat{\mathbf{tr_P}}, \hat{\mathbf{v_P}})$	
0	1	k_4	0	1	k_5
1	0	$k_4(js_P + q_P)v_P/tr_P$	1	0	$k_5 q_P v_P/tr_P$

variable. We use the transformations in the following table; to understand how to read this table, consider its second column: $\hat{q} = 0$ maps to $1 \leq q \leq 9$ and $\hat{q} = 1$ maps to $q = 10$. Printer velocity does not change, thus (v) is not transformed. Thus:

	$\hat{q}$	$\hat{tr}$	$\hat{tb}$	$\hat{tc}$	$\hat{js}$	$\hat{jc}$	$\hat{ju}$
0	q: 1 à 9	tr: 1	tb: 1	tc: 1	js: 1 à 2	jc: 0	ju: 0
1	q: 10	tr: 2 à 10	tb: 2 à 10	tc: 2 à 10	js: 3 à 5	jc: 1	ju: 1

In this model, binary variables point to extreme situations in a printer. For example, $\hat{q}$ points if printing queue is full ($\hat{q} = 1$) or if it is not ($\hat{q} = 0$). $\hat{tr}$ points if printer has paper ($\hat{tr} = 1$) or if it has not ($\hat{tr} = 0$). $\hat{tb}$ points if there is black toner ($\hat{tb} = 1$) or if there is not ($\hat{tb} = 0$). $\hat{tc}$ points if there is color toner ($\hat{tc} = 1$) if or there is not ($\hat{tc} = 0$). $\hat{ju}$ points if the user has urgency ($\hat{ju} = 1$) or if he has not ($\hat{ju} = 0$). In our model only $\hat{jc}$ does not have such interpretation. We emphasize that the information lost in the transformation is minimized with the use of original variables in probability expressions.

In the reduced model, variables inherit the same independence relations that original variables, thus the new BNs are the same of Figure 1, just changing variables x to $\hat{x}$. However, the key point is that, *at each transition, the variables in the original network are used to evaluate probability values*. That is, the MDP is solved online for the probability values that hold in the original model. The terms that are specified through the BNs are summarized in Tables 4 and 5. Terms $P(\hat{js_F})$, $P(\hat{jc_F})$ and $P(\hat{ju_F})$ are independent of the chosen action and of the actual state. These terms value $1/2$ each. Printer velocity does not change, thus if $v_F = v_P$ then $P(v_F \mid v_P)$ is 1 else it is 0.

In the reduced model, the MDP can be solved in real time using traditional algorithms like value iteration, as the online processing requirements are relatively simple. We again emphasize that, as probabilities change from iteration to iteration, the model is not stationary as a whole.

5. Experimental Results

A simulator program was used to analyze which printer model produces the best cluster performance. In this simulator, three different models of printer were employed in bid

Table 5. Conditional probabilities: $\hat{tb}$ (top) and $\hat{tc}$ (bottom).

Print action.					*Ignore* action.				
	$\hat{tb}_p = 0$		$\hat{tb}_p = 1$			$\hat{tb}_p = 0$		$\hat{tb}_p = 1$	
	$\hat{tr}_P$		$\hat{tr}_P$			$\hat{tr}_P$		$\hat{tr}_P$	
	0	1	0	1		0	1	0	1
$\mathbf{\hat{tb}_F}$	$\mathbf{P}(\mathbf{\hat{tb}_F} \mid \mathbf{\hat{q}_P}, \mathbf{\hat{tr}_P}, \mathbf{\hat{tb}_P}, \mathbf{\hat{js}_p}, \mathbf{\hat{v}_P})$				$\mathbf{\hat{tb}_F}$	$\mathbf{P}(\mathbf{\hat{tb}_F} \mid \mathbf{\hat{q}_P}, \mathbf{\hat{tr}_P}, \mathbf{\hat{tb}_P}, \mathbf{\hat{v}_P})$			
0	1	1	0	$\frac{k_6(q_P+js_P)v_P}{tb_P}$	0	1	1	0	$\frac{k_7 q_P v_P}{tb_P}$
1	0	0	1	k_6	1	0	0	1	k_7

Print action.						*Ignore* action.				
	$\hat{tc}_p = 0$		$\hat{tc}_p = 1$				$\hat{tc}_p = 0$		$\hat{tc}_p = 1$	
	$\hat{tr}_P$		$\hat{tr}_P$				$\hat{tr}_P$		$\hat{tr}_P$	
	0	1	0	1			0	1	0	1
				$\hat{jc} = 0$	$\hat{jc} = 1$					
$\mathbf{\hat{tc}_F}$	$\mathbf{P}(\mathbf{\hat{tc}_F} \mid \mathbf{\hat{q}_P}, \mathbf{\hat{tr}_P}, \mathbf{\hat{tc}_P}, \mathbf{\hat{js}_p}, \mathbf{\hat{jc}_p}, \mathbf{\hat{v}_P})$					$\mathbf{\hat{tc}_F}$	$\mathbf{P}(\mathbf{\hat{tc}_F} \mid \mathbf{\hat{q}_P}, \mathbf{\hat{tr}_P}, \mathbf{\hat{tc}_P}, \mathbf{\hat{v}_P})$			
0	1	1	0	$\frac{k_8 q_P v_P}{tc_P}$	$\frac{k_9(q_P+js_P)v_P}{tc_P}$	0	1	1	0	$\frac{k_{10} q_P v_P}{tc_P}$
1	0	0	1	k_8	k_9	1	0	0	1	k_{10}

generation for jobs: a heuristic printer model, a model with offline division of states, and a model with online reduction of states. The heuristic print model calculates its bid by adding variables: $tr + \max(tb, tc) - q$. This model does not consider uncertainty and is considered a baseline for comparisons. The tests were conducted under four different cluster configurations:

- 4 printers with different velocities, where 2 are color printers, and 10 users;
- 4 printers with different velocities, where 2 are color printers, and 15 users;
- 4 equal color printers and 10 users;
- 4 equal color printers and 15 users.

This way homogeneous cluster were compared against heterogeneous ones. Also, different workloads (10 and 15 users) were compared too. Two measures were utilized to analyze cluster performance: the mean time for a urgent job execution, and compliance with color requisition for the jobs (for instance, a color job could be executed in black and white due to redirection to a printer without color toner). Figure 2 shows the simulation results for the first measure. We notice that the offline approach has a better performance in a heterogeneous cluster. In an homogeneous cluster the online solution is better than the offline solution. The online approach has to solve the MDP for each job, adding some time to the execution. The offline approach has the solution previously calculated and stored, thus requiring only more memory space. With respect to the second measure, the two MDP solutions have an equivalent performance near to 100%. The heuristic model has the best first measure for a homogeneous cluster, but it has a poor performance in the second measure for this case.

6. Conclusion

Recent years have seen growing interest in "structured" MDPs, where the structure may be due to factorization, or hierarchical relationships among variables, or logical constraints. It is important to have guidelines for constructing MDPs so that these regularities surface naturally. We have tried in this paper to contribute with new ideas in this direction. Concepts such as limited and homogeneous variation should be in the hands of designers, so that a solid methodology can be in place for the construction of structured MDPs. We have also shown how regularities can be exploited during selection of optimal policies. Finally, we have described an application of our ideas to a real domain: we have shown an application where jobs are routed in a cluster of printers, where each printer bids for jobs using its own MDP.

Acknowledgements

The first author was supported by HP Brazil R&D; the second author was partially supported by CNPq. We acknowledge help from Instituto de Pesquisas Eldorado and FDTE.

References

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

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

C. Boutilier, N. Friedman, M. G. and Koller, D. (1996). Context-specific independence in bayesian networks. *Uncertainty in Artificial Intelligence*, pages 115 – 123.

D. Koller, A. P. (1998). Probabilistic frame-based systems. *American Association for Artificial Intelligence*, pages 580 – 587.

Glesner, S. and Koller, D. (1995). Constructing flexible dynamic belief networks from first-order probalistic knowledge bases. *ECSQARU*, pages 217–226.

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

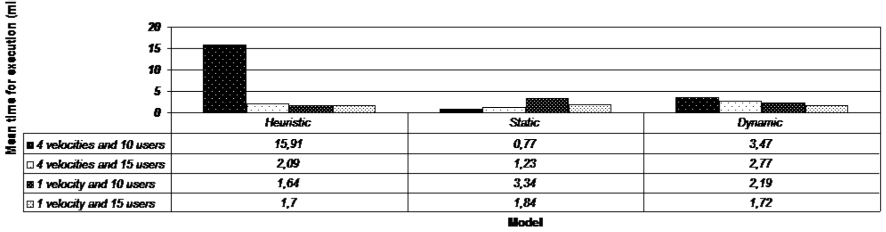
Littman, M. L., Dean, T. L., and Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 294–402, Montreal, Canada. AUAI.

Littman, M. L. and Younes, H. L. S. (2004). PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. *International Planning Competition*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Los Angeles, California.

Puterman, M. L. (1994). *Markov Decision Process*. J. Wiley & Sons, New York.

R. Sharma, P. P. (2005). Probabilistic reasoning with hierarchically structured variables. *International Joint Conference on Artificial Intelligence*.



	Heuristic	Static	Dynamic
■ 4 velocities and 10 users	15,91	0,77	3,47
□ 4 velocities and 15 users	2,09	1,23	2,77
▩ 1 velocity and 10 users	1,64	3,34	2,19
▨ 1 velocity and 15 users	1,7	1,84	1,72

Figure 2. Mean time urgent jobs execution