To appear, *Proc. Twelfth Nat. Conf. on AI (AAAI-94),* Seattle, July, 1994

Using Abstractions for Decision-Theoretic Planning with Time Constraints

Craig Boutilier and Richard Dearden
Department of Computer Science
University of British Columbia
Vancouver, BC, CANADA, V6T 1Z4
email: {cebly,dearden}@cs.ubc.ca

Abstract

Recently Markov decision processes and optimal control policies have been applied to the problem of decision-theoretic planning. However, the classical methods for generating optimal policies are highly intractable, requiring explicit enumeration of large state spaces. We explore a method for generating abstractions that allow approximately optimal policies to be constructed; computational gains are achieved through reduction of the state space. Abstractions are generated by identifying propositions that are "relevant" either through their direct impact on utility, or their influence on actions. This information is gleaned from the representation of utilities and actions. We prove bounds on the loss in value due to abstraction and describe some preliminary experimental results.

1 Introduction

Recently there has been considerable interest in probabilistic and decision-theoretic planning (DTP) [5, 9, 14, 3]. A probabilistic framework allows agents to plan in situations of uncertainty, while decision-theoretic methods permit comparison of various courses of action, or the construction of appropriate nearly-optimal behavior when (optimal) goals are unachievable. Dean et al. [2] have investigated planning in such contexts as a question of stochastic optimal control, in particular, modeling the effects of actions on the environment as a (completely observable) Markov decision process (MDP) [7]. This model allows one to view each action as a stochastic mapping among states of the environment, and allows one to associate various rewards or utilities with these states. With such a model, standard techniques can be used to construct an optimal *policy* of action that maximizes the expected reward of the agent. Unfortunately, these methods quickly become intractable as the state space grows. As a concession to these considerations, Dean et al. [2] explore anytime algorithms for policy generation using restricted *envelopes* within the state space.

We explore a different way of coping with the computational difficulties involved in optimal policy generation. By assuming a particular representation of actions, we can generate an *abstract* state space in which (concrete) states are clustered together. Standard techniques may be used in this reduced space. Our approach has several advantages over the envelope method. Foremost among these is the fact that no states are ignored in abstract policy generation – each state may have some influence on the constructed policy by membership in an abstract state. This allows us to prove bounds on the value of abstract policies (with respect to an optimal policy). Furthermore, finer-grained abstractions are guaranteed to increase the value of policies. Finally, abstractions can be generated quickly. These factors allow abstract policies of varying degrees of accuracy to be constructed in response to time pressures. The information obtained in abstract policy generation can then be used in a real-time fashion to refine the abstract policy, as we describe in the concluding section. This is also well-suited to circumstances where the goals (or reward structure) communicated to an agent change frequently; thus problem-specific abstractions can be generated as needed.

In the next section we describe the MDPs, Howard's [7] *policy iteration* algorithm for optimal policy construction and (briefly) the anytime approach of [2]. In Section 3, we discuss a possible knowledge representation scheme for actions and utilities. The information implicit in such a specification will be crucial in generating useful abstractions. In Section 4, we present an algorithm for generating an abstract state space and an appropriate decision model. We show how policy iteration is used to generate abstract policies in this state space that are directly applicable to the original (concrete) space, and prove bounds on the possible loss due to abstraction. We also discuss preliminary experimental results that suggest that abstraction of this form is quite valuable in certain types of domains.

2 Markov Decision Processes

Let W be a finite set of states or worlds, the possible situations in which a planning agent may find itself. We assume that this set of worlds is associated with some logical propositional language $\mathcal{L}$, and is thus exponential in the number of atoms generating $\mathcal{L}$. Let $\mathcal{A}$ be a finite set of actions available to an agent. An action takes the agent from one world to another, but the result of an action is known only with some probability. An action may then be viewed as a mapping from W into probability distributions over W. We write $Pr(w_1, a, w_2)$ to denote the probability that w_2 is reached given that action a is performed in state w_1. These transition probabilities can be encoded in a $|W| \times |W|$ matrix for each action. This notation embodies the usual Markov assumption that the transition probabilities depend only on the current state.

While an agent cannot (generally) predict with certainty the state that will result from its action, we assume it can observe with certainty the resulting state once the transition is made. Hence the process is *completely observable*. All uncertainty is due to the unpredictability of actions. While some have this property, there will be many domains in which this is not the case. However, complete observability is a useful simplifying assumption that allows us to explore the fundamentals of abstraction, ignoring the technical difficulties of the partially observable case.

We assume a real-valued *reward function* R, with $R(w)$ denoting the (immediate) utility of being in state w. For our purposes an MDP consists of W, $\mathcal{A}$, R and the set of transition distributions $\{Pr(\cdot, a, \cdot) : a \in \mathcal{A}\}$.

A control *policy* π is a function $\pi : W \to \mathcal{A}$. If this policy is adopted, $\pi(w)$ is the action an agent will perform whenever it finds itself in state w. Given an MDP, an agent ought to adopt an optimal policy that maximizes the expected rewards accumulated as it performs the specified actions. We concentrate here on *discounted infinite horizon* problems: the current value of future rewards is discounted by some factor β ($0 < \beta < 1$); and we want to maximize the expected accumulated discounted rewards over an infinite time period. However, our methods are suitable for finite horizon techniques such as *value iteration* [7] as well. Intuitively, a DTP problem can be viewed as finding an optimal policy.[1]

The expected *value* of a fixed policy π at any given state w is specified by

$$V_\pi(w) = R(w) + \beta \sum_{v \in W} Pr(w, \pi(w), v) \cdot V_\pi(v)$$

Since the factors $V_\pi(w)$ are mutually dependent, the value of π at any initial state w can be computed by solving this system of linear equations. A policy π is *optimal* if $V_\pi(w) \geq V_{\pi'}(w)$ for all $w \in W$ and policies π'. Howard's [7] policy iteration algorithm works by starting with a random policy and trying to improve this policy by finding for each world some action better than the action specified by the policy. Each iteration of the algorithm involves the following two steps:

1. For each $w \in W$, compute $V_\pi(w)$.
2. For each $w \in W$, find some action a such that
$$R(w) + \beta \sum_{v \in W} Pr(w, a, v) \cdot V_\pi(v) > V_\pi(w)$$
Let policy π' be such that $\pi'(w) = a$ if such an improvement exists, $\pi'(w) = \pi(w)$ otherwise.

The algorithm iterates on each new policy π' until no improvement is found. The algorithm will converge on an optimal policy, and in practice tends to converge reasonably (given, e.g., a *greedy* initial policy). The first step requires the solution of a set of $|W|$ linear equations in $|W|$ unknowns (requiring polynomial time).

Unfortunately, the factor $|W|$ will be exponential in the number of atoms in our underlying language. Optimal policy construction is thus computationally demanding. Such solutions methods may be reasonable in the design of an agent requiring a fixed policy. A solution might be computed off-line and a corresponding reactive policy embodied in the agent "once and for all." However, a fixed policy of this type is not feasible in a setting where an agent must respond to the changing goals or preferences of a user. While in many domains the system dynamics may be relatively stable, the reward structure for which an agent's behavior is designed might change frequently (e.g., in response to different task assignments). Therefore, fast on-line computation of policies will be necessary and the computational bottleneck must be addressed. We expect optimality (of policy) to be sacrificed for computational gain.

To deal with the difficulties of policy construction, Dean et al. [2] assume that it will be sufficient in many circumstances to consider a very restricted subset of the state space. Their basic approach is as follows: an initial *envelope* $\mathcal{E}$, or subset of worlds, is chosen and a *partial policy* is computed for $\mathcal{E}$ (i.e., a policy applicable only for states in $\mathcal{E}$) using policy iteration. Since an agent might fall out of the envelope while executing a policy, all transitions out of $\mathcal{E}$ are assumed to fall into a distinguished OUT state. If an agent ends up in this state, it must extend (or alter) the current envelope and compute a new partial policy. The anytime aspect of this model is captured by an algorithm which constructs a partial policy for $\mathcal{E}$, and if time permits extends $\mathcal{E}$ to include more states. Given more time the algorithm will compute a more complete partial policy.

[1] If a "final" state stops the process, we may use absorbing states (at which no action is applicable). Classical (categorical) goals can also be specified [2].

This model requires an estimate of the penalty associated with the OUT state. In [2] it is suggested that the expected value of all "out states" and some factor accounting for the time to recompute a policy be used; but determining this expected value requires at least some approximation to an optimal policy (though in certain domains heuristics may be available). An initial envelope must also be provided. In general, it is not clear how a good initial envelope should be generated, although in [2] some reasonable guidelines are suggested for certain domains (such as navigation).

We propose an alternative anytime model for nearly optimal policy construction based on *abstraction* of the state space. While considering a restricted envelope may be appropriate in many instances, in general finding a suitable subset may be difficult, and partial policies are not suitable for an agent that may find itself in arbitrary start states. In our approach, we ignore "irrelevant aspects" to the domain by grouping together states that differ only in these aspects. Approximately optimal policies can be generated in this smaller state space. Since irrelevance is a matter of degree, more accurate policies can be constructed (at greater computational expense) by incorporating additional details. Our model provides several advantages over the envelope method. First, policies are applicable at all states of the process. Second, we may provably bound the degree to which policies fall short of optimal. This factor can be used to influence how detailed an abstraction is required (and also the direction in which abstractions should be refined). Finally, our model has the feature that more refined abstractions lead to better policies.

3 Representation of MDPs

It is unreasonable to expect that a DTP problem will be specified using an explicit stochastic transition matrix for each action and an explicit reward function. Regularities in action effects and reward structure will usually permit more concise representations. We discuss one possible representation for actions and utilities, and show how this information can be exploited in abstraction generation. While our algorithm depends on the particular representation given, the nature of our method does not. More natural and sophisticated representations can be used (e.g., causal networks).

3.1 Action Representation

To represent actions that have "probabilistic effects" we will adopt a modification of the basic scheme presented in [9], itself a modification of the STRIPS representation allowing effects (add/delete lists) to be applied with a certain probability. We start by defining an *effect* to be a (finite) consistent set of literals. If E is an effect, its occurrence changes the world. We let $E(w)$ denote the world that results when effect E is applied to w. In the usual STRIPS fashion, $E(w)$ satisfies all literals in E and agrees with w on all other literals.

To deal with nondeterministic actions, we assume that possible effects occur with specified probabilities. A *probabilistic effect* is a finite set of effects $E_1, \ldots E_n$ with associated probabilities $p_1, \ldots p_n$, written $\langle E_1, p_1; \ldots E_n, p_n\rangle$. We insist that $\sum p_i = 1$. An effects list EL applied to w induces a discrete distribution over W; the likelihood of moving to v when EL occurs at w is given by

$$\mathit{Pr}(v|EL, w) = \sum\{p_i : E_i(w) = v\}$$

An action can have different effects in different contexts. We associate with each action a finite set $D_1, \ldots D_n$ of mutually exclusive and exhaustive propositions called *discriminants*; and associated with each discriminant is a probabilistic effects list EL_i. An action a applied at w yields the distribution over outcomes induced by EL_k, where D_k is the (unique) discriminant satisfied by w.

Parting from [9], we add the notion of an *action aspect*. Some actions have different classes of effects that occur independently of each other. For instance, under a given action, a certain literal may be made true if some condition holds. A distinct literal may independently be made true if another condition holds. To capture this, an action can be specified using different aspects, each of which has the form of an action as described above (i.e., each aspect has its own discriminant set). The actual effect of an action at a world is determined by applying the effects list of the relevant discriminant for *each* aspect of that action. More precisely, let w be some world to which we apply an action with k aspects. Since each aspect has a proper discriminant set associated with it, w satisfies exactly one discriminant for each aspect. Assume these are $D^1, \cdots, D^k$ and that each D^i has an associated effects list $\langle E^i_1, p^i_1; \ldots E^i_n, p^i_n\rangle$. An effect from each applicable list will occur with the specified probability, these probabilities being independent. Intuitively, action aspects capture the kind of independence assumptions one might find in a causal network or influence diagram. Thus, the net effect of an action A at w is the union of these effects (sets of literals), one chosen from each aspect. The probability of this combined effect is determined by multiplying these probabilities. Thus, we have

$$\mathit{Pr}(v|A, w) = \sum\{p^1_{j_1} \cdot p^2_{j_2} \cdots p^k_{j_k} : E(w) = v\}$$

where E is an effect such that

$$E = E^1_{j_1} \cup E^2_{j_2} \cup \cdots \cup E^k_{j_k}$$

To ensure that actions are well-formed we impose the following consistency condition: if D^i and D^j are mutually consistent discriminants taken from distinct aspects of a given action, then their effects lists must contain no atoms in common (thus, the union above is consistent).

Action	*Discr.*	*Effect*	*Prob.*	*Action*	*Discr.*	*Effect*	*Prob.*
GoL1	$\overline{L1}, L2$	$L1, \overline{L2}$	0.9	GoL2	$\overline{L2}, L1$	$L2, \overline{L1}$	0.9
(aspect1)		$\emptyset$	0.1	*(aspect1)*		$\emptyset$	0.1
	`else`	$\emptyset$	1.0		`else`	$\emptyset$	1.0
GoL1	$R, \overline{U}$	W	0.9	GoL2	$R, \overline{U}$	W	0.9
(aspect2)		$\emptyset$	0.1	*(aspect2)*		$\emptyset$	0.1
	`else`	$\emptyset$	1.0		`else`	$\emptyset$	1.0
BuyC	$L2$	HCR	0.8	DelC	$L1, HCR$	$HCU, \overline{HCR}$	0.8
		$\emptyset$	0.2			$\overline{HCR}$	0.1
	`else`	$\emptyset$	1.0			$\emptyset$	0.1
GetU	$L1$	U	0.9		$\overline{L1}, HCR$	$\overline{HCR}$	0.9
		$\emptyset$	0.1			$\emptyset$	0.1
	`else`	$\emptyset$	1.0		`else`	$\emptyset$	1.0

Figure 1: An example of STRIPS-style action descriptions.

An example best illustrates this representation. We assume a user at location $L1$ instructs a robot to get her coffee at $L2$ across the street. The robot can have coffee (HCR) and an umbrella (U). It can get wet (W) if it is raining (R), and the user can have coffee (HCU) as well. Actions include going to $L1$ or $L2$, buying coffee, delivering coffee to the user and getting an umbrella. These action specifications are listed in Figure 1.[2] The actions GoL1 and GoL2 each have two aspects. GoL1 induces transition probabilities from any world w satisfying $\overline{L1}$, $L2$, R and $\overline{U}$ as follows: the effect $\{L1, \overline{L2}, W\}$ occurs with probability $.81$; $\{L1, \overline{L2}\}$ occurs with probability $.09$; $\{W\}$ occurs with probability $.09$; and the null effect $\emptyset$ occurs with probability $.01$.

3.2 Utility Representation

To represent the immediate rewards or utilities associated with world states, we assume a user specifies a partition of the state space that groups worlds together if they have the same utility. This is achieved by providing a mutually exclusive and exhaustive set of propositions and associating a utility with each proposition in this set. There are more natural and concise methods for utility representation. For example, if the utilities of propositions are independent and additive, these can be directly specified (relative to some base level). Indeed, such a scheme will generally make the problem we address in the next section easier. But this simple scheme will be sufficient for our purposes. Note that any reward function over a state space generated by a set of propositions can be represented in this fashion.

In our example, the primary goal of the agent is to get coffee; but we would like it to stay dry in the process. No other propositions influence the immediate reward of a state. We obtain the following specification of our reward function:

Discr.	*Reward.*	*Discr.*	*Reward.*
$HCU, \overline{W}$	1.0	HCU, W	.9
$\overline{HCU}, \overline{W}$	.1	$\overline{HCU}, W$	0.0

We dub the propositions that determine the immediate utility of a state *utility discriminants.*

4 Generating an Abstract Model

State-aggregation methods have been used to accelerate convergence of MDP solution methods with some success (e.g., [12]). However, the emphasis has not been on the automatic generation of aggregated states, nor on the exploitation of regularities implicit of the representation of an MDP. Abstraction has also been used in classical planning to guide the search for concrete, fully-specified plans [11]. In particular, Knoblock [8] has proposed methods for generating abstractions by exploiting a STRIPS-style action representation. Our procedure uses the representation scheme for actions in much the same fashion, as well as utilities, to decide which propositions are most important in the construction of a good policy, and which details can be ignored with little penalty. Once certain propositions are shown to be irrelevant, the state space can be collapsed by clustering together worlds in which only irrelevant propositions differ (i.e., worlds are distinguished by relevant propositions only). Policy iteration can then be performed in this abstract space and an approximately optimal policy can be generated. Unlike the classical setting, an abstract policy can be used immediately and can be refined on-line.

There are three issues that must be addressed using such a scheme: 1) which propositions should be deemed relevant? 2) how should actions be mapped onto the abstract space? 3) how should utilities be mapped onto the abstract space?

4.1 The Abstract State Space

In order to generate an abstract state space, a set of relevant propositions must be chosen. From the perspective of immediate utility, only those propositions that occur among the set of utility discriminants are of direct relevance. In our example, W and HCU are the only (immediately) relevant atoms.

[2] We ignore preconditions for actions here, assuming that an action can be "attempted" in any circumstance. However, preconditions may play a useful role by capturing user-supplied heuristics that filter out actions in situations in which they *ought not* (rather than *cannot*) be attempted. The `else` discriminant is simply a convenient notation for the negation of all action discriminants that appear earlier in the list.

Of course, immediate relevance is a matter of degree. The truth or falsity of HCU has a greater immediate impact on utility than W. It is this observation that will allow us to ignore certain atomic propositions.

Initially, we imagine an agent generates some set $\mathcal{IR}$ of *immediately relevant* propositions. The larger this set is, the more fine-grained an abstraction will be. This is the crucial factor in the anytime nature of our approach. A larger number of abstract states will require more computation, but will yield more accurate results. It is therefore important that the relevant propositions be chosen carefully so as to take full advantage of this tradeoff. Propositions with the greatest impact on utility are most relevant. A number of strategies might be employed for discovering the most relevant propositions. We discuss one such strategy below, once the exact nature of our algorithm has been elaborated. In our example, we decide that HCU is the most important proposition, setting $\mathcal{IR} = \{HCU\}$. If we add W to $\mathcal{IR}$, then the entire range of immediate utility is captured (and optimal solutions will be generated, but at added computational cost – see below). We will assume for simplicity that utility discriminants are conjunctions of literals (this is sufficient for any utility function) and that $\mathcal{IR}$ consists of atoms.

An agent should make distinctions based not only on immediately relevant propositions, but on propositions that may influence the achievement of these. Thus, we provide a recursive definition for the set $\mathcal{R}$ of *relevant propositions*. The idea is based on the construction of abstraction hierarchies by Knoblock [8] in a classical STRIPS domain. It relies on the particular action representation above; but the general idea is well-suited to other action representations (e.g., the situation calculus and, especially, causal networks).

Definition The set $\mathcal{R}$ of *relevant propositions* is the smallest set such that: 1) $\mathcal{IR} \subseteq \mathcal{R}$; and 2) if $P \in \mathcal{R}$ "occurs" in an effect list of some action aspect, each proposition occurring in the corresponding discriminant is in $\mathcal{R}$.

Again, for simplicity, we will assume that $\mathcal{R}$ consists of atoms and that an atom occurs in a list if the associated positive or negative literal occurs. Notice that only the atoms of a discriminant that might (probabilistically) lead to a certain effect are deemed relevant; other conditions associated with the same action aspect can be ignored.[3] We call such discriminants *relevant*. We leave aside the question of an algorithm for generating the set $\mathcal{R}$ given $\mathcal{IR}$ (see [1] for details); an obvious modification of Knoblock's algorithm for generating *problem specific constraints* suffices. The *operator graph* construct of [13] might also prove useful in determining relevant discriminants. The "branching factor" of stochastic actions, the average size of discriminant and effects lists, and the degree of "interconnection" will determine the time required to generate $\mathcal{R}$; it will certainly be insignificant in relation to the time required to produce the abstract policy.

[3] This connection can be weakened further by ignoring discriminant atoms whose influence on utility is marginal (see the concluding section).

In our example, HCU is influenced by $L1$ and HCR. Both are, in turn, influenced by $L2$. Thus $\mathcal{R} = \{L1, L2, HCR, HCU\}$. Notice that the use of action aspects, while not necessary, can be useful not only as a convenient representational device, but also for reducing the number of relevant atoms for a given problem.

Given the set of relevant atoms, we can generate an abstract state space by clustering together worlds that agree on the members or $\mathcal{R}$, ignoring irrelevant details.

Definition The abstract state space generated by $\mathcal{R}$ is $\widetilde{W} = \{\widetilde{w}_1, \ldots \widetilde{w}_n\}$, where: a) $\widetilde{w}_i \subseteq W$; b) $\cup\{\widetilde{w}_i\} = W$; c) $\widetilde{w}_i \cap \widetilde{w}_j = \emptyset$ if $i \neq j$; and d) $w, v \in \widetilde{w}_i$ iff $w \models P$ implies $v \models P$ for all $P \in \mathcal{R}$.

Any worlds that agree on the truth of the elements of $\mathcal{R}$ are clustered together — in our example, the atoms R, U and W are ignored. Thus, $\widetilde{W}$ contains just 16 states rather than the 128 contained in W.

4.2 Abstract Actions and Utilities

If an optimal policy is to be constructed over this abstract state space, we require actions and a reward function which are applicable in this space. In general, computing the transition probabilities for actions associated with an arbitrary clustering of states is computationally prohibitive; for it requires that one consider the effect of an action on each world in an abstract state. Furthermore, computing the probability of moving from one cluster to another under a given action requires that a prior distribution over worlds in the first cluster be known, which cannot be known in general.

Fortunately, our abstraction mechanism is designed to avoid such difficulties. The action descriptions for the concrete space can be readily modified to fit the abstract space as shown by the following propositions.

Proposition 1 *Let $\widetilde{w}$ be an abstract state and let $w, v \in \widetilde{w}$. Then w satisfies a relevant discriminant for some action aspect iff v does.*

Proposition 2 *Let E be any effect. i) If E is associated with an irrelevant discriminant, then $E(w) \in \widetilde{w}$; and ii) $E(w) \in \widetilde{u}$ iff $E(v) \in \widetilde{u}$.*

Intuitively, these conditions ensure that for any two worlds in a given cluster, an action maps these with equal probability to worlds in any other cluster. In other words, actions can be viewed as applying directly to clusters. Furthermore, the action discriminants and probabilities can be used within the abstract space to determine the probability of a transition from one cluster to another when an action is performed. (We give a general algorithm in [1].) Because of these factors no new abstract actions are required and the abstract state space and transition matrices induced by the original actions enjoy the Markov property.

In our example, the cluster containing those worlds that satisfy $L1, \overline{L2}, \overline{HCU}, \overline{HCR}$ maps to cluster $\overline{L1}, L2, \overline{HCU}, \overline{HCR}$ with probability 0.9 under GoL1 and maps to itself with probability 0.1. Under action GetU, it maps to itself with probability 1.0 (since GetU affects no relevant atoms).

To associate an immediate utility with a given cluster, we use the midpoint of the range of utilities for worlds within that cluster. For any cluster $\widetilde{w}$, let $\min(\widetilde{w})$ denote the minimum of the set $\{R(w) : w \in \widetilde{w}\}$ and $\max(\widetilde{w})$ denote the corresponding maximum. Our *abstract reward function* is:

$$R(\widetilde{w}) = \frac{\max(\widetilde{w}) + \min(\widetilde{w})}{2}$$

This choice of $R(\widetilde{w})$ minimizes the possible difference between $R(w)$ and $R(\widetilde{w})$ for any $w \in \widetilde{w}$, and is adopted for reasons we explain below. Any cluster satisfying HCU has an abstract utility of $.95$ (since some worlds have a reward of 1.0 and some 0.9), while $\overline{HCU}$ ensures a utility of $.05$.

4.3 Abstract Policies and their Properties

With the abstract state space, actions and reward function in place, we now have a Markov decision process for which an optimal policy can be constructed using policy iteration. Since computation time for an optimal policy is a function of the number of states, the cardinality of $\mathcal{R}$ will determine the savings over optimal policy construction in the original state space. Since the state space increases exponentially in size as the number of relevant atoms increase, any reduction can result in tremendous speed-up.

Of course, this speed-up comes at the cost of generating possibly less-than-optimal policies. Thus, some measure of the loss associated with constructing policies in the abstract space must be proposed. Let us denote by $\widetilde{\pi}$ the *optimal abstract policy* (that generated for our abstract MDP). We take $\widetilde{\pi}$ to be mapped into a concrete policy π in the obvious way: $\pi(w) = \widetilde{\pi}(\widetilde{w})$ where $w \in \widetilde{w}$.

Along with $\widetilde{\pi}$, policy iteration will produce an abstract value function $V_{\widetilde{\pi}}$. We can take $V_{\widetilde{\pi}}$ to be an estimate of the true value of the concrete policy π; that is, $V_\pi(w)$ is approximated by $V_{\widetilde{\pi}}(\widetilde{w})$ where $w \in \widetilde{w}$. The difference between $V_\pi(w)$ and $V_{\widetilde{\pi}}(\widetilde{w})$ is a measure of the accuracy of policy iteration over the abstract space in estimating the value of the induced concrete policy.

Of more interest is the degree to which the generated abstract policy differs from truly optimal policy. Let $\pi*$ denote some optimal policy for the original process, with corresponding value function $V_{\pi*}$. The true measure of goodness for an abstract policy $\widetilde{\pi}$ is the degree to which the induced concrete policy π differs from $\pi*$; more precisely, we should be interested in the difference between $V_\pi(w)$ and $V_{\pi*}(w)$ (for any world w).

Bounds on the magnitudes of these differences can be computed using the *utility span* for a cluster $\widetilde{w}$: *span*$(\widetilde{w}) = \max(\widetilde{w}) - \min(\widetilde{w})$. This is the maximum degree to which the estimate $R(\widetilde{w})$ of the immediate utility of a world in that cluster differs from the world's true utility $R(w)$. Let δ denote the maximum span among all clusters in $\widetilde{W}$. We have the following bounds (recall β is the discounting factor):

Theorem 3 $|V_{\widetilde{\pi}}(\widetilde{w}) - V_\pi(w)| \leq \frac{\delta}{2(1-\beta)}$, *for any* $w \in W$.

Theorem 4 $|V_{\pi*}(w) - V_\pi(w)| \leq \frac{\beta\delta}{1-\beta}$, *for any* $w \in W$.

Thus we have some reasonable guarantees about the effectiveness of the computed policy. The key factor in the effectiveness of an abstraction is the size of δ in relation to the ranges of possible values. Intuitively, the abstract policy can lose no more than δ reward per time step or action taken (compared to optimal). This a very facile worst-case analysis and is unlikely to ever be reached for any world (let alone all worlds). Some preliminary experimental results have borne out this intuition.

In our example, with $\mathcal{R} = \{L1, L2, HCR, HCU\}$, the abstract policy $\widetilde{\pi}$ generated essentially requires the robot to get coffee directly, ignoring the umbrella, whereas the true optimal policy $\pi*$ will have the robot get the umbrella if it is raining (if it starts at $L1$). With a discounting factor β of 0.9, Theorem 4 guarantees that the expected value of the abstract policy, for any state, will be within 0.9 of optimal. To calibrate this, we note that the possible *optimal values* (over W) range from 0 and 10. Computing the abstract policy $\widetilde{\pi}$ shows that for all $w \in W$ we have $|V_{\pi*}(w) - V_\pi(w)| \leq 0.8901$. Furthermore, at only 12 of 128 states did the concrete and optimal values differ at all. The time required to produce the abstract policy was 0.12 seconds compared with 21 seconds for policy iteration performed on the complete network. Although we cannot expect such performance in all domains, these results, as well as other experiments, show that in many cases the algorithm performs extremely well, producing policies that are close to optimal and requiring considerably less computation time than policy iteration.

The utility span formulation of abstraction value shows the direction in which one should refine abstractions: the propositions that should be incorporated into a new abstraction are those that reduce the maximum utility span δ the most. At some point in refinement, should δ reach 0, optimal policies will be generated. Finally, should a more refined abstraction be used, the generated policy cannot be worse (and will typically be better, if any utility span is reduced, even if the maximum span δ remains constant). Let $\mathcal{IR}_1 \subseteq \mathcal{IR}_2$ be two sets of immediately relevant atoms, and let π and ψ be the concrete policies induced by $\mathcal{IR}_1$ and $\mathcal{IR}_2$, respectively.

Theorem 5 $|V_{\psi*}(w) - V_{\psi}(w)| \leq |V_{\pi*}(w) - V_{\pi}(w)|$ *for any* $w \in W$.

Naturally, this analysis shows how one should determine the initial set $\mathcal{IR}$ of immediately relevant atoms. For a particular set $\mathcal{IR}$, the corresponding set $\mathcal{R}$ of relevant atoms is not immediately obvious, but can be computed as described above (in negligible time). In the case where abstractions are to be generated frequently for different problems, appropriate information of this type can be re-used. The size of $\mathcal{R}$ is a good predictor of the time required to generate an abstract policy. Thus, our algorithm has a "contract anytime" nature (relative to the computation of $\mathcal{R}$). The quality of the abstract policy can be bounded by Theorem 4, and the "quality" of a particular $\mathcal{IR}$ can be computed easily by considering the abstract states it induces. More precisely, let $T_{\mathcal{IR}}$ be the set of truth assignments to $\mathcal{IR}$ (we treat these loosely as conjunctions of literals). Let D be the set of utility discriminants. For any $t \in T_{\mathcal{IR}}$, let

$$\max(t) = \max_{d \in D}\{R(d) : d \not\models \neg t\}$$

and let $\min(t)$ denote the corresponding minimal value. The "goodness" of $\mathcal{IR}$ is measured by

$$\max_{t \in T_{\mathcal{IR}}} \{\max(t) - \min(t)\}$$

The smaller this value (the maximal utility span), the tighter the guarantee on the optimality of the abstract policy. While the computation of this maximal span is exponential in the number of immediately relevant atoms, $\mathcal{IR}$ will always be restricted to atoms mentioned in the reward function R, which will be a rather small subset of atoms.

The idea of using utility spans to generate abstractions is proposed by Horvitz and Klein [6], who use the notion in single-step decision making. Our analysis can be applied to their framework to establish bounds on the degree to which an "abstract decision" can be less than optimal. Furthermore, the notion is useful in more general circumstances, as our results illustrate.

5 Concluding Remarks

We have shown that abstraction can be a valuable tool for computing close-to-optimal policies for MDPs and DTP. Our approach is one that is amenable to both theoretical and experimental analysis, and appears promising given our preliminary results. Our model provides "contract anytime behavior" since the computation time required is determined by the number of relevant propositions chosen. Our approach has a number of interesting benefits. Since abstractions can be generated relatively easily, our approach is well-suited to problem-specific abstractions, for instance, to particular reward functions or starting state distributions (see Knoblock [8], who also discusses problem-specific abstraction). Furthermore, since abstractions cover all possible states, the abstract state space offers a useful method for representing reactive strategies. A close-to-optimal strategy can be encoded with exponential space-saving. This may be useful also in determining which bits of information a reactive agent should ignore when sensor costs are high.

There are a great number of directions in which this work is being extended. We are currently exploring an expected-case analysis by making certain assumptions about problem distributions, augmenting the worst-case results provided here. We are also exploring other methods of ignoring details. In particular, we have developed some methods for considering only discriminants whose relevant effects are sufficiently probable or sufficiently important [1]. In our example, carrying the umbrella might *slightly* decrease the chance of successful coffee delivery, but can be ignored. While the concrete action probabilities are not accurate in such an abstract space, they are roughly correct. The Markov assumption is "approximately" true and the error associated with solving the problem with inaccurate transition probabilities can be bounded. Discounting can be incorporated in such a model to further reduce the number of relevant atoms; essentially, effects from a "distance" can be given less weight. A crucial feature of this extension is the fact that abstractions are generated reasonably quickly. Nicholson and Kaelbling [10] have proposed abstracting state spaces in a similar fashion using sensitivity analysis to determine relevant variables; however, such a method has high computational cost.

A key problem is the adaptation of our method to different action and utility representations (e.g., using causal networks, or general propositional action and utility discriminants). This should lead to adaptive and nonuniform clustering techniques. However, there are certain technical difficulties associated with nonuniform clusters. We hope to investigate the features of both the envelope and abstraction methods and determine to which types of domains each is best suited and how the intuitions of both might be combined (see [10]). Features that will ensure the success of our technique include: a propositional domain representation; approximately additive utilities over features; a wide range of utilities; goals with possible minor improvements, and so on. The extent to which real domains possess these qualities is ultimately an empirical question.

We are also exploring search methods that can be used to refine abstract policies [4]. While an abstract policy might not be ultimately acceptable, it may be suitable as a set of default reactions under time-pressure. As time permits, finite-horizon decision-tree search can be used to refine the policy. The abstract value function, a by-product of abstract policy construction, can be used quite profitably as a heuristic function to guide this search. Preliminary results appear quite promising. In our example, search of depth 4 guarantees optimal action [4]. Finally, we hope to generalize our techniques to semi-Markov and partially observable processes. The computational difficulties associated with the partially observable case make abstraction especially attractive in that setting.

Acknowledgements

Discussions with Moisés Goldszmidt have considerably influenced our view and use of abstraction for MDPs. Thanks to Eric Horvitz, Ann Nicholson and an anonymous referee for helpful comments. This research was supported by NSERC Research Grant OGP0121843 and a UBC University Graduate Fellowship.

References

[1] Craig Boutilier and Richard Dearden. Using abstractions for decision-theoretic planning with time constraints. Technical report, University of British Columbia, Vancouver, 1994. (Forthcoming).

[2] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *Proc. of AAAI-93*, pages 574–579, Washington, D.C., 1993.

[3] Thomas Dean and Michael Wellman. *Planning and Control*. Morgan Kaufmann, San Mateo, 1991.

[4] Richard Dearden and Craig Boutilier. Integrating planning and execution in stochastic domains. In *AAAI Spring Symposium on Decision Theoretic Planning*, pages 55–61, Stanford, 1994.

[5] Mark Drummond and John Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proc. of AAAI-90*, pages 138–144, Boston, 1990.

[6] Eric J. Horvitz and Adrian C. Klein. Utility-based abstraction and categorization. In *Proc. of UAI-93*, pages 128–135, Washington, D.C., 1993.

[7] Ronald A. Howard. *Dynamic Probabilistic Systems*. Wiley, New York, 1971.

[8] Craig A. Knoblock. *Generating Abstraction Hierarchies: An Automated Approach to Reducing Search in Planning*. Kluwer, Boston, 1993.

[9] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. Technical Report 93-06-04, University of Washington, Seattle, June 1993.

[10] Ann E. Nicholson and Leslie Pack Kaelbling. Toward approximate planning in very large stochastic domains. In *AAAI Spring Symposium on Decision Theoretic Planning*, pages 190–196, Stanford, 1994.

[11] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[12] Paul L. Schweitzer, Martin L. Puterman, and Kyle W. Kindle. Iterative aggregation-disaggregation procedures for discounted semi-Markov reward processes. *Operations Research*, 33:589–605, 1985.

[13] David E. Smith and Mark A. Peot. Postponing threats in partial-order planning. In *Proc. of AAAI-93*, pages 500–506, Washington, D.C., 1993.

[14] Michael P. Wellman and Jon Doyle. Modular utility representation for decision-theoretic planning. In *Proc. of AIPS-92*, pages 236–242, College Park, MD, 1992.