Modeling Probabilistic Actions for Practical Decision-Theoretic Planning

AnHai Doan
anhai@cs.uwm.edu
Decision Systems and Artificial Intelligence Laboratory
Department of Electrical Engineering and Computer Science
University of Wisconsin-Milwaukee
Milwaukee, WI 53201

Abstract

Most existing decision-theoretic planners represent uncertainty about the state of the world with a precisely specified probability distribution over world states. This representation is not expressive enough to model many interesting classes of practical planning problems, and renders inapplicable some abstraction-based planning approaches. In this paper we propose as a remedy a more general world and action model with a well-founded semantics based on probability intervals. We introduce the concept of *interval mass assigment*. Unlike mass assignments, which assign a probability mass to each set of states, interval mass assignments assign a probability interval to each set of states and are more expressive. Interval mass assignments are interpreted as representing sets of probability distributions and are used in our framework to represent the uncertainty about the state of the world. Within this representation, we present a projection rule and a method for computing a plan's expected utility. We compare our approach with existing probability-interval approaches. We provide complexity results and empirical evidence which suggest evaluating plans (projecting plans and computing the expected utility) in our framework is efficient, and the action model is applicable in real-world planning domains.

Introduction

Any planning model that strives to solve real-world problems must deal with the inherent uncertainty in the domains. Various approaches have been suggested and the generally accepted and traditional solution is to use probability to model domain uncertainty (Dean *et al.* 1993; Goldman & Boddy 1994; Kushmerick, Hanks, & Weld 1993). A representative of this approach is the BURIDAN planner (Kushmerick, Hanks, & Weld 1993). In BURIDAN uncertainty about the true state of the world is modeled with a probability distribution over the state space. Actions have uncertain effects, and each of these effects is also modeled with a probability distribution.

The BURIDAN representation, which we will call the ***single probability distribution (SPD) model***, has a well-founded semantics and is the underlying representation for the majority of existing probabilistic planning systems (Kushmerick, Hanks, & Weld 1993; Dean *et al.* 1993). It is very well suited, for example, to model classes of Markov Decision Process in which the state space is finite and the probabilities of transitions among the states can be relatively easy to acquire (Dean *et al.* 1993). For many interesting problem classes, however, the SPD model cannot be used because exact probability information is either not available, or too costly to acquire, or simply can not be manipulated to suit the representation (Snow 1986; Strat 1990). For example, we may know that fuel level can take on the values 3, 10, and 12; but 10 and 12 may be too close to distinguish, in which case we could obtain only the probability that the fuel level is 10 or 12.

Another problem with the SPD model arises when some abstraction techniques are employed to reduce the complexity of decision-theoretic planning (Haddawy & Doan 1994; Haddawy, Doan, & Goodwin 1995). Some approaches use abstract worlds, each of which represents a set of worlds at the base level and thus is necessarily represented with constraints characterizing a set of probability distributions (Doan 1996). Such abstract worlds cannot be represented in the SPD framework.

Most of the work on computing with probability intervals has been done in the context of probabilistic logic, Bayesian networks, and influence diagrams (Nilsson 1986; Frisch & Haddawy 1994; Breese & Fertig 1991; Ramoni 1995). There has been little work dealing directly with the restrictiveness of the single probability distribution model in planning. An exception is the work of Chrisman (Chrisman 1992), who develops an action model based on the notion of Dempster-Shafer mass functions. His work concentrates on presenting a new world and action model and rules for projecting actions. Practical isssues such as projection complexity are not considered and empirical evidence that the model works is not offered. Finally, in some cases his projection rule is not closed under his representation and thus cannot be used to project plans (see the section on related work and discussion). In a previous paper (Haddawy & Doan 1994), we identified several useful types of action abstraction, presented procedures for creating the abstractions, and presented sound projection rules. But because we were working within an SPD model of actions, the abstraction techniques presented in that paper cannot be applied to abstract actions. This is a severe limitation since one expects to use such techniques to create abstraction hierarchies.

This paper makes the following contributions: We introduce a world and action model which have well-founded semantics based on probability intervals. We propose representing the uncertainty about the world with a set of probability distributions over the state space. Mass assignments, which assign a probability mass to each set of states, have typically been used to represent sets of probability distributions (Chrisman 1992). In our framework, however, we introduce the concept of interval mass assignment – defined as assigning a probability interval to each set of states – and use interval mass assignments to represent sets of probability distributions. We discuss several theoretical and practical advantages of the interval mass assignment over the mass assignment representation. The two key advantages are that the former is more expressive than the latter, and the former can also accommodate projection and abstraction techniques that the latter cannot. Within the interval mass assignment representation, we present a correct projection rule and a method for computing a plan's expected utility. We compare our approach with the existing probability-interval approaches, provide bounds on the complexity of plan evaluation, and show empirical evidence that the action model is applicable in solving real world problems.

Since our framework is a generalization of the SPD framework, we start in the next section by briefly reviewing the BURIDAN representation. We then introduce our world and action model, together with a projection rule and proof of correctness. We show that the projection rule is also correct in projecting plans, and suggests an efficient algorithm to compute the expected utility of a plan given the final world resulting from plan projection. We briefly show how the model has been implemented in the DRIPS decision-theoretic planner (Haddawy, Doan, & Goodwin 1995) and how DRIPS imposes practical requirements to improve the efficiency of computing with the model. We conclude with related work and discussion of practical applicability issues.

The BURIDAN Representation

In this section we briefly review the BURIDAN representation. In the subsequent sections our representation will be developed as a natural generalization of the BURIDAN model.

World and Action A *state* contains all relevant information about the world at a particular moment. The set of all possible states will be denoted by Ω. Uncertainty about the state of the world is represented with a probability distribution over the state space.

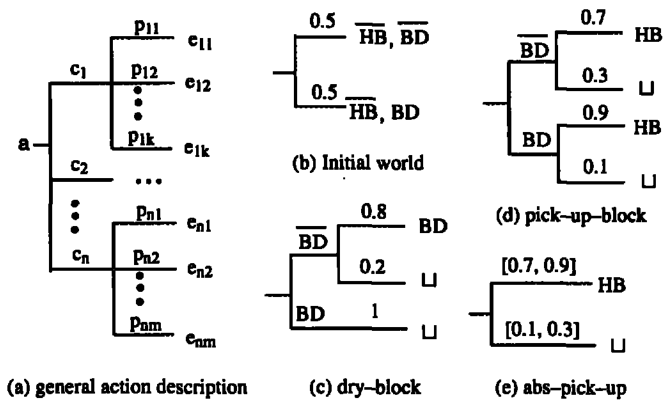

Figure 1: Action and world descriptions.

Figure 1.b shows, for example, a world in which the block is not held ($\overline{HB}$) and the probability of its being dry (BD) is 0.5.

An action is represented with a finite set of mutually exclusive and jointly exhaustive conditions $\{c_1, c_2, \cdots, c_n\}$ on Ω, each of which is associated with a probability distribution over Ω. We shall interpret the c_i as sets or as logical sentences, depending on the context. An action can be depicted with a tree structure as shown in Figure 1.a, where the p_{ij} are probabilities summing to 1 for each index i, and the e_{ij} are the effects. Each effect e_{ij} is a function mapping a state into a state: for a state b, the resulting state is denoted by $e_{ij}(b)$. Semantically, when an action a is executed on a state b where the condition c_i holds ($b \in c_i$), the effect e_{ik}, k = 1, 2, ..., will be realized with the specified probability p_{ik}, that is, $P(e_{ik}(b)|\mathrm{a}, b) = p_{ik}$[1]. So the result of executing action a on state b will be a probability distribution in which each state $e_{ik}(b)$, k = 1, 2,..., is assigned the probability p_{ik}, respectively, and all the remaining states receive the probability zero. We denote this probability distribution with P_{ib}.

Figures 1.c and 1.d show two slightly modified sample action descriptions taken from (Kushmerick, Hanks, & Weld 1993). The "dry-block" action states that if the block is not dry ($\overline{BD}$) prior to executing the action then with probability 0.8 it will be dry after the action (BD) and with probability 0.2 it will remain not dry (the action changes nothing; the effect is denoted with ⊔). The "pick-up-block" action states that if the block is not dry previously then with probability 0.7 it will be held after the action (HB) and with probability 0.3 the action changes nothing.

Projecting Actions *Projecting* an action on a world

[1]Naturally if two or more effects produce the same state, e.g., $e_{ik}(b) = e_{il}(b)$, then the conditional probability of that state is the sum of all probabilities associated with those effects.

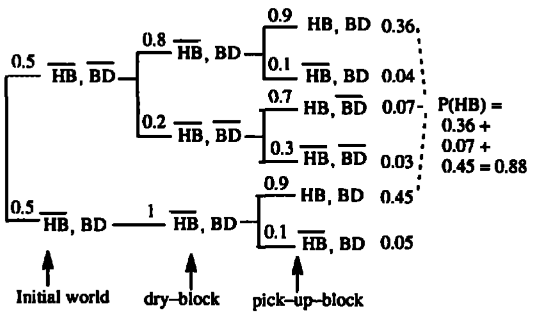

Figure 2: Projecting the plan "dry-block" "pick-up-block".

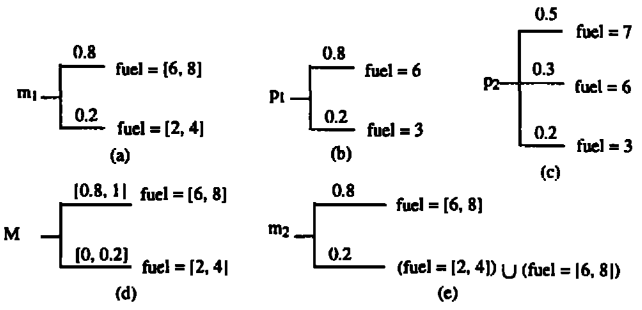

Figure 3: (a)-(c) A mass assignment and two probability distributions it represents (d)-(e) An interval mass assignment and an equivalent mass assignment.

is the problem of computing the probability of any subset of the state space conditioned on the action being executed in that world. Based on the action semantics, the probability of a set $A \subseteq \Omega$ conditioned on an action a being executed on a state b can be computed as $P(A|\mathsf{a}, b) = \sum_i P(c_i|b) \cdot P_{ib}(A)$, where $P(c_i|b)$ is 1 if $b \in c_i$ and 0 otherwise, and $P_{ib}(A)$ is the probability of A in the distribution P_{ib} (i.e., $\sum_{e_{ik}(b) \in A} p_{ik}$). From the above equation we have the following projection rule to compute the probability of a set A conditioned on an action a being executed on a world represented by a probability distribution P_{pre} over the states

$$P(A|\mathsf{a}, P_{pre}) = \sum_{b\in\Omega} P_{pre}(b) \cdot P(A|\mathsf{a}, b)$$

$$= \sum_{b\in\Omega} P_{pre}(b) \cdot (\sum_i P(c_i|b) \cdot P_{ib}(A)) \qquad (1)$$

Projecting Plans A plan is a finite sequence of actions. Computing a plan's expected utility can be done by first sequentially projecting the actions in the plan on the initial world to compute the probability distribution over the plan's outcomes. Figure 2 shows how the plan "dry-block" "pick-up-block" is projected by BURIDAN on the initial world depicted in Figure 1.b. This projection is called *forward projection* and is sanctioned by the projection rule described in (1). The probability of each final state is computed by multiplying all the probabilities along the path leading to that state. The probability that the agent is holding the block after executing the plan is 0.88. (Using the utility function assigning 1 to any state in which HB is true and 0 to all other states, we would get this probability as the plan's expected utility.)

Forward projection is the most frequently executed operation in many planners (Kushmerick, Hanks, & Weld 1993; Haddawy, Doan, & Goodwin 1995). Since projection time grows exponentially with the number of actions in a plan and the number of branches in each action, an efficient projection algorithm is necessary.

World Model

Arguments about the restrictiveness of the SPD world and action model in the introduction suggest that uncertainty about the world should be represented with a set of probability distributions instead of an exactly specified distribution. Mass assignments, which assign a probability mass to each set of states, have typically been used to represent sets of probability distributions (Shafer 1981; DeRobertis & Hartigan 1981; Chrisman 1992). Since an understanding of the mass assignment representation facilitates an understanding of interval mass assignments, we start this section with a brief review of the mass assignment representation.

Let Ω denote the state space; a mass assignment $m : 2^{\Omega} \rightarrow [0, 1]$ assigns to each subset of Ω a portion of the probability mass. We have $\sum_{B\subseteq\Omega} m(B) = 1$ and $m(\emptyset) = 0$. The set $B\subseteq\Omega$ is called a *focal element* of m if $m(B) > 0$; and we will say the pair $\langle B, m(B)\rangle$ forms a *branch* of m. A probability distribution P is said to be *consistent* with a mass assignment m iff $P(B) = \sum_{b\in B} P(b) \geq \sum_{C\subseteq B} m(C)$ for all $B \subseteq \Omega$. The intuition behind such consistency is that given P consistent with m, for any $B\subseteq\Omega$ such that $m(B) \neq 0$ there is a way to divide up the probability mass $m(B)$ among $b \in B$ such that the total mass assigned to a state b is $P(b)$ (Chrisman 1992). The set of the probability distributions consistent with m will be denoted as $\wp(m)$.

A mass assignment m can be interpreted as bounding the true probability of any subset of Ω with a probability interval (Shafer 1981). It can also be interpreted as representing the uncertainty concerning the true probability distribution over Ω with a set of probability distributions, namely $\wp(m)$. Figure 3.a shows a mass assignment, the first (second) focal element of which consists of all the worlds in which $fuel$ is between 6 and 8 (2 and 4). Examples of the probability distributions consistent with this mass assignment are shown in Figures 3.b and 3.c.

We now define the concept of *interval mass assignment*. Denote the set of all intervals in $[0, 1]$ as I, an ***interval mass assignment (IMA)*** $M : 2^{\Omega} \rightarrow I$ assigns to each subset of Ω a probability interval in I. M can be understood as a set of mass assignments $\{m | m : 2^{\Omega} \rightarrow [0,1]$ such that $\forall B \subseteq \Omega\ m(B) \in M(B)\}$[2] and sometimes will be interpreted as such. We define the set of all probability distributions consistent with M to be $\bigcup_{m \in M} \wp(m)$, and denote this set with $\wp(M)$. The concepts of focal elements and branches for interval mass assignments are straightforward generalizations of the definitions for mass assignments.

Adopting the IMA rather than the mass assignment representation to model the world offers several theoretical and practical advantages. First, the IMA representation is more expressive than the mass assignment representation. It is not difficult to see that any set of probability distributions that can be represented with a mass assignment can also be represented with an IMA. The converse, however, is not true. For example, the set of probability distributions represented by the IMA with three branches $\{\langle [0,0.5], b_1 \rangle, \langle [0,0.5], b_2 \rangle, \langle [0,0.5], b_3 \rangle\}$, where the b_i are distinct states, cannot be represented with any mass assignment[3].

Second, even in those cases where theoretically one can use either an IMA or a mass assignment to represent the same set of probability distributions, limits on the expressiveness of the implementation language may not allow such use of mass assignments. For example, we can convert the IMA M shown in Figure 3.d into the mass assignment shown in Figure 3.e; the union of M's focal elements is taken to be the second focal element of the resulting mass assignment. However, in the language of the DRIPS planner (see the section on an implementation) this union can not be represented directly and must be replaced by the set $\{fuel = [2,8]\}$, which clearly contains more states than the original union. The resulting mass assignment now contains more probability distributions than the original IMA.

Finally, the IMA representation can accommodate projection rules and abstraction operators that are not closed under the mass assignment representation. An example of such projection rules is the one we shall introduce in the next section. An example of abstraction operators that cannot be used within the mass assignment framework is the intra-action abstraction operator used in the DRIPS planner (Doan 1996). When we use this abstraction operator to abstract away the conditions in the action "pick-up-block" shown in Figure 1.d and group together similar outcomes, we end up with the abstract action "abs-pick-up" shown in Figure 1.e, whose branches are characterized by probability intervals.

[2] Actually a IMA M needs the condition $\sum_{A \subseteq \Omega} \min M(A) \leq 1 \leq \sum_{A \subseteq \Omega} \max M(A)$ to contain any mass assignment, where min (max) denote the lower (upper) endpoint of the interval $M(A)$. Checking this condition can be done in a manner similar to checking whether the probabilities in a probability distribution sum to 1.

[3] We thank Vu Ha for supplying this example.

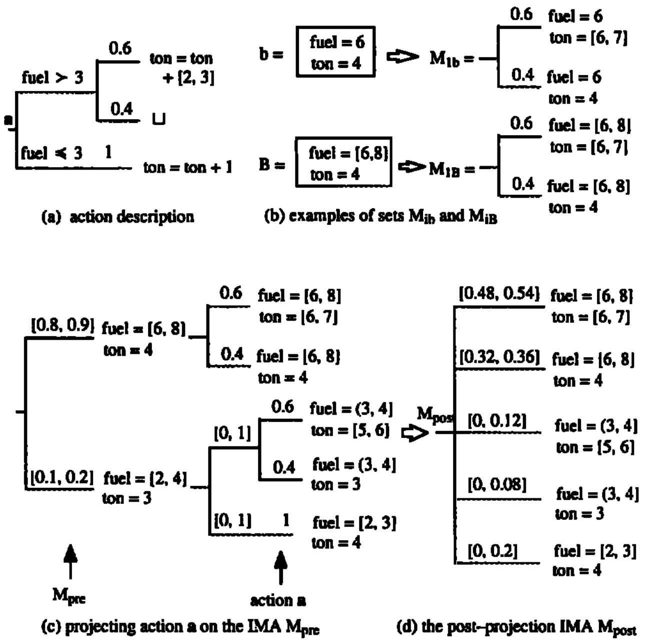

Figure 4: Examples of action description, sets M_{ib} and M_{iB}, action projection, and the post-projection IMA.

Because of all the above theoretical and practical advantages, we adopt the IMA instead of the mass assignment representation to model uncertainty about the state of the world. A world is modeled with an IMA; the true probability distribution representing the world is among those represented by the IMA.

Action Model

Action Syntax We retain the action syntax discussed in the section on the BURIDAN representation and depicted in Figure 1.a, but replace the p_{ij} with probability intervals I_{ij}, and the effects e_{ij} with E_{ij} mapping states into sets of states. Figure 4.a shows an example action description in which the amount of goods delivered varies, depending on $fuel$ being greater or less than 3. An assignment such as $ton = ton + [2,3]$ states that ton after executing the action will be increased by an unknown amount between 2 and 3. Since metric effects such as $fuel = fuel + [2,3]$ can map a state ($fuel = 2$) into a set of states ($fuel = [4,5]$) we define each effect E_{ij} to be a mapping from states into *sets* of states $e_{ij} : \Omega \rightarrow 2^{\Omega}$. Abusing notation, for a set $B \subseteq \Omega$ we will use $E_{ij}(B)$ to denote the set $\bigcup_{b \in B} E_{ij}(b)$.

We can say that each condition c_i is associated with a IMA M_i whose branches are $\langle I_{ij}, E_{ij} \rangle$; but to be more precise, M_i should be called a *functional* IMA because its focal elements, effects E_{ij}, are functions instead of sets. For a state b and a condition c_i we write M_{ib} to denote the instantiated IMA computed from M_i by

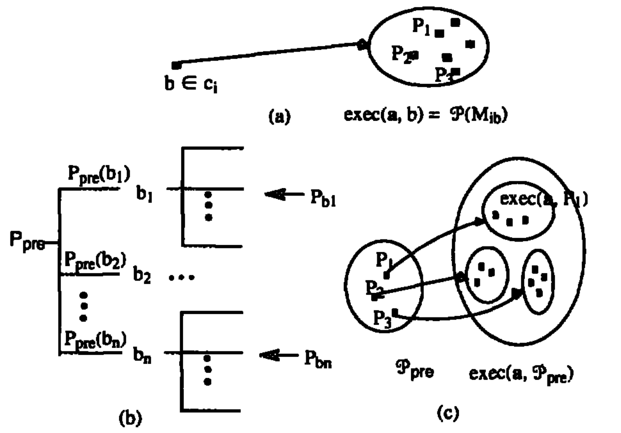

Figure 5: Illustrations on action semantics: pre-execution and post-execution sets.

replacing each focal element E_{ij} in M_i with the set of states $E_{ij}(b)$. Note the similarity between the definition of M_{ib} and that of P_{ib} in the section on the BURIDAN representation. For a set $B \subseteq \Omega$ we also define M_{iB} similar to M_{ib} but with each E_{ij} being replaced by the set of states $E_{ij}(B)$. Figure 4.b shows the computed IMAs M_{1b} and M_{1B} for the state b, set B, and the condition $c_1 = (fuel > 3)$ of the action a depicted in Figure 4.a.

Action Semantics In contrast to the SPD model, in which executing an action a on a state $b \in c_i$ yields the probability distribution P_{ib}, in our model executing an action a on a state $b \in c_i$ yields a set of probability distributions represented by the IMA M_{ib}. This means that the true post-execution probability distribution is in the set $\wp(M_{ib})$. Denote the set of all possible probability distributions over Ω that can result from executing a on b as $exec(\mathsf{a}, b)$; we have $exec(\mathsf{a}, b) = \wp(M_{ib})$ (Figure 5.a). We define $exec(\mathsf{a}, P)$ and $exec(\mathsf{a}, \wp)$, where P is a probability distribution and $\wp$ is a set of probability distributions, in a similar manner below.

Based on the above semantics the set of all possible probability distributions over Ω that can result from executing action a on a probability distribution P_{pre} can be computed as

$$exec(\mathsf{a}, P_{pre}) = \{P_{post} | \\ \forall A \subseteq \Omega \;\; P_{post}(A) = \textstyle\sum_b P_{pre}(b) \cdot P_b(A); P_b \in exec(\mathsf{a}, b)\}$$

where $P_{post}(A)$ and $P_b(A)$ are the probabilities of the set A in the probability distributions P_{post} and P_b, respectively. This is illustrated in Figure 5.b; for a state b, if $P_b \in exec(\mathsf{a}, b)$ is the true probability distribution resulting from executing a on b then the probability of set A along the branch of b is $P_{pre}(b) \cdot P_b(A)$; $P_{post}(A)$, the total probability of set A in the post-execution probability distribution P_{post}, is the sum of the probabilities of A along all the branches. The distribution P_{post} is a possible distribution resulting from executing a on P_{pre}; the set $exec(\mathsf{a}, P_{pre})$ thus is the set of all such P_{post}.

When the world is represented with a set of probability distributions $\wp_{pre}$, executing action a on this world yields the set

$$exec(\mathsf{a}, \wp_{pre}) = \bigcup_{P_{pre} \in \wp_{pre}} exec(\mathsf{a}, P_{pre}) \qquad (2)$$

which is illustrated in Figure 5.c

Projecting Actions When an action a is executed on the world represented by an IMA M_{pre} the set of all possible post-execution distributions will be $exec(\mathsf{a}, \wp(M_{pre}))$. Projecting action a on M_{pre} by computing this set explicitly based on action semantics is computationally infeasible in all but small domains. Instead, we will develop a computationally efficient projection rule that when given M_{pre} and action a will return an IMA M_{post} such that $\wp(M_{post}) \supseteq exec(\mathsf{a}, \wp(M_{pre}))$. The IMA M_{post} will be our best estimation of the post-execution set $exec(\mathsf{a}, \wp(M_{pre}))$.

An intuitive way of developing the projection rule is to generalize the rule used in the SPD model and shown in (1). One of the advantages of such a rule will be a forward projection algorithm which works in a manner similar to the forward projection algorithm of the BURIDAN representation. In Equation 1, $P(A|\mathsf{a}, P_{pre})$ can be replaced by $M_{post}(A)$, $P_{pre}(b)$ by $M_{pre}(B)$, $P(c_i|b)$ by $P(c_i|B)$, and $P_{ib}(A)$ by $M_{i(B\cap c_i)}(A)$; the set B runs over all the subsets of Ω. There is a problem with this generalization, however, because $P(c_i|B)$ is the sum of the probability masses (taken from the probability mass of B) of those elements in B which satisfy c_i, and this sum cannot be determined uniquely because it may vary depending on how the probability mass of B is distributed among its elements. Our solution is to bound $P(c_i|B)$ with a probability interval $P_I(c_i|B)$. Observe that $P(c_i|B)$ is zero if $c_i \cap B = \emptyset$, 1 if $c_i \cap B = B$, and a value between 0 and 1 otherwise. So we can define the "conditional probability interval of c_i given B" $P_I(c_i|B)$ as follows: $P_I(c_i|B)$ is 0 if $c_i \cap B = \emptyset$, 1 if $c_i \cap B = B$, and [0,1] otherwise. We have the following projection rule

Projection Rule 1 *Given an action* a *and an IMA* M_{pre}, M_{post} *is calculated such that for all* $A \in 2^{\Omega}$

$$M_{post}(A) = \\ \sum_{B \subseteq \Omega} M_{pre}(B) \cdot \left(\sum_i P_I(c_i|B) \cdot M_{i(B\cap c_i)}(A) \right). \qquad (3)$$

Note the similarity between this projection rule and Equation 1. Intuitively, Projection Rule 1 states that the probability of any set $A \subseteq \Omega$ after "executing" action a is the sum over conditions c_i of the probability of A given that c_i is realized; for any set $B \subseteq \Omega$, the probability that condition c_i is realized is $P(c_i|B)$, which is bounded by $P_I(c_i|B)$. Figure 4.c shows an example of how the rule is used to project the action a depicted in Figure 4.a on an IMA M_{pre}. Here the forward projection algorithm is almost exactly the same as that of the BURIDAN model (compare the two projection trees of Figures 2 and 4.c). Figure 4.d shows the post-projection IMA M_{post}, the focal elements of which are the leaves of the projection tree in Figure 4.c and the probability intervals of which are obtained by multiplying the intervals along the paths.

The following theorem states that the set of probability distributions represented by the post-projection IMA M_{post} subsumes the set of distributions resulting from executing the action on the pre-projection IMA M_{pre}, thus proving the correctness of Projection Rule 1.

Theorem 1 *Given $P_{pre} \in \wp(M_{pre})$, action* a, *and $P_{post} \in exec(\mathrm{a}, P_{pre})$, there exists a mass assignment $m_{post} \in M_{post}$, where M_{post} is calculated using (3), such that $P_{post} \in \wp(m_{post})$. This, together with (2), implies that $exec(\mathrm{a}, \wp(M_{pre})) \subseteq \wp(M_{post})$.*

For a proof of this theorem, see (Doan 1996).

Plan Model

Projecting Plans We define a *plan* to be a finite sequence of actions. Semantically, when a plan $p = \mathrm{a}_1\mathrm{a}_2\cdots\mathrm{a}_n$ (the a_i are actions) is executed on a world represented by a set of probability distributions $\wp_{pre}$ the result will be a world represented by a set of distributions $\wp_{post}$ and can be computed as $\wp_{post} = exec(\mathrm{a}_n, exec(\mathrm{a}_{n-1}, \cdots, exec(\mathrm{a}_1, \wp_{pre})\cdots))$. This set will be denoted as $exec(p, \wp_{pre})$.

Since the set $exec(p, \wp_{pre})$ cannot be computed directly, we approximate it by projecting actions in plan p sequentially using Projection Rule 1. Let M_{post} be the post-projection IMA resulting from projecting plan p on the initial world M_{pre}; we can easily prove that $\wp(M_{post}) \supseteq exec(p, \wp(M_{pre}))$. So Projection Rule 1 is also correct with respect to projecting plans.

Computing a Plan's Expected Utility Given a utility function $U : \Omega \rightarrow R$, which assigns to each state a number indicating its relative desirability, the expected utility of a probability distribution P over Ω is computed as the sum of the utility of each state, weighted by the state's probability. So for the set of probability distributions $\wp_{post}$ representing the result of executing a plan p we have a corresponding set of expected utility values. Determining this set is usually infeasible in practice, so we shall approximate the set by computing an interval that includes all values in the set. Observe that by projecting plan p using Projection Rule 1 we obtain a IMA M_{post} which satisfies $\wp(M_{post}) \supseteq \wp_{post}$. So if we denote the maximum (minimum) of the expected utilities of the distributions in the set $\wp(M_{post})$ with $EU_{max}(M_{post})$ ($EU_{min}(M_{post})$) then the interval $[EU_{min}(M_{post}), EU_{max}(M_{post})]$ includes all utility values in the set corresponding to $\wp_{post}$; this interval is therefore an approximation of the expected utility of plan p.

The following theorem shows that it is easy to compute EU_{max} and EU_{min} for IMAs, using the fractional knapsack greedy algorithm (Cormen, Leiserson, & Rivest 1990).

Theorem 2 *For an IMA M with n branches $\langle I_i, B_i \rangle$ let $U_{max,i} = \max_{b \in B_i} U(b)$ and $U_{min,i} = \min_{b \in B_i} U(b)$ $i = 1, 2, \cdots, n$. We have*

$$EU_{max}(M) = \max_{\sum_i p_i = 1, p_i \in I_i} \sum_i p_i \cdot U_{max,i} \quad (4)$$

$$EU_{min}(M) = \min_{\sum_i p_i = 1, p_i \in I_i} \sum_i p_i \cdot U_{min,i}. \quad (5)$$

According to (Cormen, Leiserson, & Rivest 1990), the greedy algorithm to compute $EU_{max}(M)$ and $EU_{min}(M)$ runs in $O(n \log n)$ time.

An Implementation

Projection Rule 1 provides an algorithm to project plans, but if it is to be used efficiently in practice, computing the terms $M_{i(B \cap c_i)}$ in (3) must be done in an effective way. Similarly, efficient computing of a plan's expected utility requires that computing the terms $U_{max,i}$ and $U_{min,i}$ in Theorem 2 be done efficiently. This section describes an implementation where these requirements are met without overly restricting the representation language. It briefly discusses how the action model is implemented in DRIPS, a decision-theoretic planner that works with metric and temporal domains (Haddawy, Doan, & Goodwin 1995).

In DRIPS, a planning domain is modeled with a finite set of features called *attributes*. Each attribute takes a value from a totally ordered set called its *domain*. A state is represented with an assignment of a value to every attribute. Since worlds are represented with IMAs, we need a language to represent the focal elements, that is, sets of states. In DRIPS a set of states is represented by a set of constraints of the form $attr_i = V_i$, where V_i is a value interval in the domain of the attribute $attr_i$. The set $\{fuel = [4, 6], time = [3, 5]\}$ thus represent the set of all states in which $fuel$ takes a value in the interval [4,6], and $time$ takes a value in the interval [3,5]. Figure 3.d shows a world as represented in DRIPS.

Action conditions are logical expressions such as $(and\ (fuel > 5)\ (time = 6))$, $(fuel = time)$, etc. Each action effect E_{ij} is a set of assignments in the form $attr_i = f_i$, where each f_i is a function involving domain attributes, such as $f_{fuel} = fuel + [2, 3]$, $f_{cost} = price \cdot ton$, $f_{block-dry} = 1$, etc. Figure 4.a shows an action description with action effects as represented in DRIPS.

Efficient computing of the term $M_{i(B \cap c_i)}$ in (3) requires that for any set of states C (represented by a set of constraints of the form $attr_i = V_i$) the sets $E_{ij}(C)$

defined as $\bigcup_{b \in C} E_{ij}(b)$ be computed in an effective way. This requirement in turn is satisfied if for any function f_i and set of states C as defined above the set $f_i(C) = \bigcup_{b \in C} f_i(b)$ can be computed or approximated efficiently. Observe that this is satisfied if f_i changes its monotonicity at a small finite number of points[4] with respect to any attribute appearing in f_i. For example, assume $f_i = fuel + 2$ and C contains the constraint $fuel = [4, 7]$, then since f_i is monotonically increasing with respect to $fuel$, the bounds on the values of $fuel$ in $f_i(C)$ will be the $f_i(b)$-s with b-s being the minimal and maximal values of $fuel$ in C. We have, therefore, $f_i(C) \subseteq [4+2, 7+2] = [6, 9]$. All functions f_i implemented so far in DRIPS have countable changes in their monotonicity.

The utility function U (see the section on plan model) is implemented as a procedure that takes as input a state and provides as output the state's utility. The requirement imposed on the utility function is that computing the minimum and maximum of the utilities of the states in a set be done efficiently. Since attributes in such a set take on value intervals, if the utility function changes its monotonicity at a small number of points with respect to every attribute appearing in the function, then computing the utility bounds of a set of states does not pose a practical problem. All domains implemented so far in DRIPS (see the next section) have utility functions with the above property.

Related Work and Discussion

Our framework is similar to Chrisman's (Chrisman 1992). He used mass assignments to represent uncertainty about the state of the world. A rule for projecting actions was presented, but its correctness in projecting plans and its run-time complexity were not discussed. The action model was not implemented.

Chrisman's projection rule returns the convex hull of the post-execution probability distributions. Since mass assignments represent convex sets of probability distributions, and the post-projection result must be represented with a mass assignment, the projection rule seems to be as tight as one can hope for. Unfortunately, the projection rule is not closed with respect to the mass assignment representation: there are cases where the post-projection set of probability distributions cannot be represented with a mass assignment. Recently the author has communicated (Chrisman) that if lower envelopes, an instance of lower probabilities and a generalization of mass assignments, are used instead of mass assignments to represent the uncertainty about the world then the projection rule is valid and in special cases it produces tight bounds. However, even when this is the case, the rule requires iterating over all subsets of the state space to compute their mass assignments. With a state space of cardinality n the rule would require considering 2^n cases, and thus is infeasible in practice in all but small domains.

Most of the work in extending the restrictiveness of the single probability distribution have been placed in the context of belief network and influence diagram (Breese & Fertig 1991; Ramoni 1995). Some deal directly with building action models and rules for projecting actions (Chrisman 1992). The restrictiveness of these models is that they all assume a finite state space; the inference algorithm (marginalization in belief net and forward projection in action model) takes advantage of this fact and usually requires some sort of iteration over all the states, thus rendering the techniques unapplicable for problems with large or infinite state space.

In our framework, Projection Rule 1 (Equation 3) sanctions a forward projection procedure similar to that in the BURIDAN framework. The complexity of our projection procedure does not depend on the cardinality of the state space, but grows exponentially with the number of actions in the plan. The previous section shows that computing terms in (3) can be assumed to take constant time. So the complexity of the projection rule is linear with the size of the projection tree. Assume that the action to be projected has p conditions and the IMA associated with each condition has q branches. Assume further that in projecting the action only x conditions yield the "conditional probability interval" other than zero. Then projecting the action on a set B would yield a tree with qx branches. For a plan of n actions the final projection tree has $(qx)^n$ branches. In the best case the projection tree has q^n branches and the projection algorithm has the same complexity as that of forward projection in the BURIDAN framework; in the worst case the tree will have $(qp)^n$ branches.

While the complexity of our projection rule can be bounded and shown to be manageable, the precision of the rule is difficult to evaluate analytically. When the factor x in the above paragraph is 1, the rule can be shown to return the tightest possible bounds; in the remaining cases we have been unable to obtain analytical results. On the first inspection, the projection rule in (3) seems to generate a "diluted" probability model over time because on each projection the probability intervals of a number of paths in the projection tree are multiplied with the vacuous interval $[0, 1]$. So the probability intervals of the paths in the projection tree seem to approach $[0, 1]$. This, however, is not the case. When a probability interval is multiplied with $[0, 1]$, only its lower bound is "widened" to zero, its upper bound remains the same. This shows that in applying our projection rule, only the lower bounds of the probability intervals might be "ruined" (being equal zero). In the case the lower bounds are zero, we can still obtain fairly tight expected utility bounds because the intact upper probability bounds and the constraint that the probabilities of the paths sum to one eliminate a large number of "illegitimate" probability distributions. Our experiments with the DRIPS planner, which

[4] From the point of view of asymptotic complexity, finiteness is sufficient.

are reported below, support this hypothesis and thus suggest that the projection rule is effective in practice.

Conceptually, DRIPS uses a divide-and-conquer technique to do planning. It constructs abstract actions from the provided actions, then uses abstract actions to builds abstract plans. Each abstract plan represents a set of concrete plans; the abstract plan's expected utility interval can be computed by projecting the plan using Projection Rule 1 and then evaluating the plan using Theorem 2; this interval subsumes all utility intervals of the concrete plans in the set. So when the utility intervals of two abstract plans do not overlap, no concrete plan in the set represented by the abstract plan with the "lower" utility interval can be the optimal plan, thus that abstract plan and all its instances can be eliminated from further consideration. Since the concrete plans in the set being eliminated are not projected and their utility values computed, we realize computational savings.

We have applied DRIPS to two real world medical planning problems (Doan 1996) These problems involve choosing the best plan (combination of test, treat, and other options) from a set of plans according to a utility function modeling the physician's and the patient's preferences. In (Haddawy, Doan, & Goodwin 1995) we showed that the branch-and-bound technique frequently used in decision tree analysis cannot handle one of these problems efficiently, while DRIPS yielded very good running results, taking in some cases only 15% as much running time as the branch-and-bound algorithm. In one problem with the plan space of 6,206 plans DRIPS eliminates 5,551 plans without explicitly computing their expected utilities, yielding a pruning rate of 89%. Such a good pruning rate indicates not only that the abstractions provide useful heuristic information but also that the projection rule yields expected utility bounds tight enough to allow the elimination of those abstract plans that represent a large number of concrete plans. In the second problem the plan space contains 258 plans. We spent very little efforts building good abstractions, and still obtained a high pruning rate of 48%. The projection rule thus appears to provide a good balance in trade-off between projection complexity and precision. We are currently testing DRIPS and thus the action model on more real-world planning problems, developing techniques to estimate the loss in expected utility due to using our projection rule, and developing alternative projection rules.

Acknowledgments. This work was partially supported by NSF grant IRI-9509165, by an AEG award from Sun Microsystems, and by a UWM graduate school fellowship. We thank Peter Haddawy, Richard Goodwin, Lonnie Chrisman, and Vu Ha for their valuable comments on the topics of this paper.

References

Breese, J., and Fertig, K. 1991. Decision making with interval influence diagram. *Uncertainty in Atrificial Intelligence 6*. P.P Bonissone, M. Henrion, L.N. Kanal and J.F. Lemmer (Editors).

Chrisman, L. Personal communications.

Chrisman, L. 1992. Abstract probabilistic modeling of action. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, 28–36.

Cormen, T.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. Cambridge, MA: The MIT Press.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 574–579.

DeRobertis, L., and Hartigan, J. 1981. Bayesian inference using intervals of measures. *The Annals of Statistics* 9(2):235–244.

Doan, A. 1996. An abstraction-based approach to decision-theoretic planning for partially observable metric domains. Technical report, Dept. of Elect. Eng. & Computer Science, University of Wisconsin-Milwaukee. Masters Thesis.

Frisch, A., and Haddawy, P. 1994. Anytime deduction for probabilistic logic. *Artificial Intelligence* 69(1-2):93–122.

Goldman, R., and Boddy, M. 1994. Epsilon-safe planning. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 253–261.

Haddawy, P., and Doan, A. 1994. Abstracting probabilistic actions. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 270–277.

Haddawy, P.; Doan, A.; and Goodwin, R. 1995. Efficient decision-theoretic planning: Techniques and empirical analysis. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 229–236.

Kushmerick, N.; Hanks, S.; and Weld, D. 1993. An algorithm for probabilistic planning. Technical Report 93-06-03, Dept. of Computer Science and Engineering, University of Washington. (To appear in *Artificial Intelligence*).

Nilsson, N. 1986. Probabilistic logic. *Artificial Intelligence* 28:71–87.

Ramoni, M. 1995. Ignorant influence diagrams. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1869–1875.

Shafer, G. 1981. Constructive probability. *Synthese* 48:1–60.

Snow, P. 1986. Bayesian inference without point estimates. In *National Conference on Artificial Intelligence*, 233–237.

Strat, T. M. 1990. Decision analysis using belief functions. *International Journal of Approximate Reasoning* 4:319–417.