Modeling Probabilistic Actions for Practical Decision-Theoretic Planning

AnHai Doan*
Department of EE & CS
University of Wisconsin-Milwaukee
PO Box 784
Milwaukee, WI 53201
anhai@cs.uwm.edu
414 229-2722

Abstract

Most existing decision-theoretic planners represent uncertainty in 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 action model with a well-founded semantics based on probability intervals. In particular, sets of probability distributions representing uncertainty about the world are represented with mass assignments, the representation of which is slightly modified to suit practical considerations. We present a projection rule and prove it correct. Complexity results and empirical evidence are also provided 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 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 (0; 0; 0; 0) and the generally accepted and traditional solution is to use probability to model domain uncertainty (0; 0; 0). A representative of this approach is the BURIDAN planner (0). 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. Projecting a plan thus does not result in a single final state, but a probability distribution over the state space. To make the representation computationally tractable, the probability distributions involved take non-zero probabilities on only a finite number of states.

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 and decision-theoretic planning systems (0; 0; 0). 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 (0; 0). 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 (0; 0; 0). 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 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 (0; 0; 0). Some approaches use abstract worlds, each of which represents a set of worlds at the base level and thus are necessarily represented with constraints characterizing a set of probability distributions (0; 0). Such abstract worlds cannot be represented in the SPD framework.

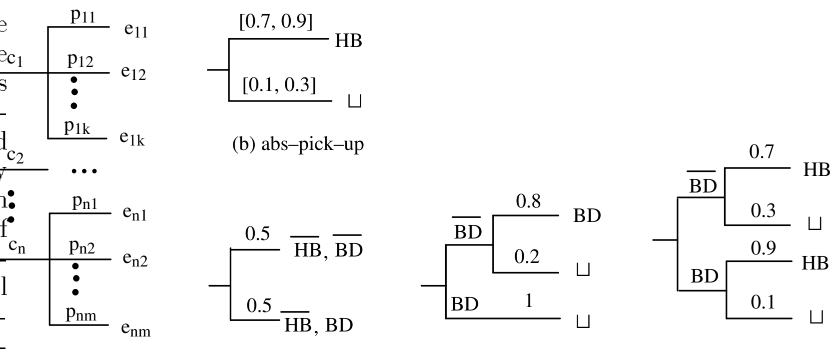
Most of the work on computing with probability intervals has been done in the context of probabilistic logic (0; 0; 0) or Bayesian networks and influence diagrams (0; 0; 0). There has been little work dealing directly with the restrictiveness of the single probability distribution model in planning. Chrisman (0) 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 Section). In a previous paper (0), 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 a single probability distribution 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 tech-

*This work was partially supported by NSF grant IRI-9509165, by an AEG award from Sun Microsystems, and by a UWM graduate school fellowship.

niques to create abstraction hierarchies.

This paper makes the following contributions: We define general world and action models which have well-founded semantics based on probability intervals and also provide computational tractability. We propose representing the uncertainty about the world with a set of probability distributions characterized by a mass assignment. Our representation differs from those proposed earlier in that the representation of mass assignments is modified to suit practical considerations, thus contributing to creating an action model which is computationally tractable. The new representation also proves crucial in accommodating abstraction techniques which are reported on in related papers (0; 0). We 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. Sections 3 and 4 then introduce our world and action model, together with a projection rule and proof of correctness. Section 5 shows 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. Section 6 briefly shows how the DRIPS decision-theoretic planner (0; 0), which uses abstraction techniques, implements the action model and imposes practical requirements to improve the efficiency of computing with the model. We discuss related work and practical applicability issues in Section 7.



(a) general action description (b) abs–pick–up (c) Initial world (d) dry–block (e) pick–up–block

Figure 1: Action and world descriptions.

The BURIDAN Representation

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.c 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.

Actions are context-dependent. 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, $P(e_{ik}(b)|\mathsf{a}, b) = p_{ik}$[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.d and 1.e show two slightly modified sample action descriptions taken from (0). 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 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 $P(c_i|b)$ just acts an an indicator function that is 1 just if $b \in c_i$, 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

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

Projecting Plans A plan is a finite sequence of actions. A planning problem cast in terms of decision-

[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.

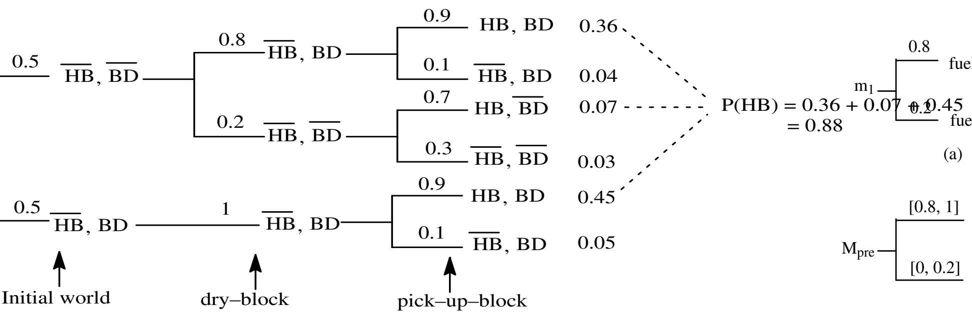


Figure 2: Projecting the plan "dry-block" "pick-up-block".

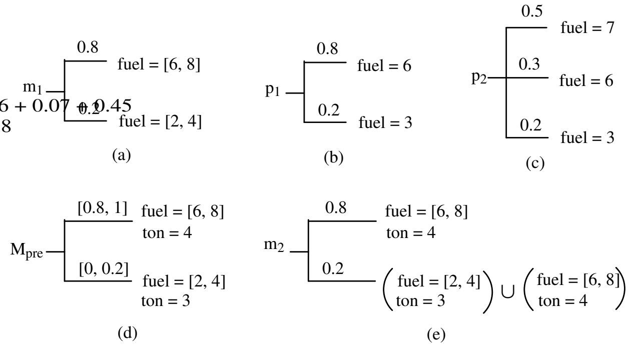


Figure 3: (a)-(c) A mass assignment and two probability distributions it represents (d)-(e) A general mass assignment and a mass assignment it represents.

theoretic planning provides as input a set of plans (or a set of actions together with instructions on how to construct plans), a utility function, and an initial world in which the plan is to be executed; and requires as output the optimal plan, that is, the one with the highest expected utility. 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.c. 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 (0; 0; 0). 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 Section 1 suggest that uncertainty about the world be represented with a set of probability distributions instead of an exactly specified distribution. Lower probability has been found to be a convenient representation for sets of probability distributions (0; 0; 0). In particular, belief functions (0), which constitute a subclass of lower probabilities, are very appealing because when represented as mass assignments (obtained from the Möbius transform of lower probability (0)) they have the intuitive interpretation of assigning probability masses to sets of states. Since this mass assignment interpretation facilitates the derivation of our projection rule in Section , and it is not clear how much additional expressive power can be gained by choosing a more general representation, we adopt the mass assignment framework to model actions and the world, but modify its standard representation slightly to suit practical considerations.

It is worth repeating at this point that we use belief functions as a convenient notation for talking about constraints on probability distributions. They should not be interpreted as representing evidence (0) and thus are not combined using Dempster's rule of combination (0).

Let Ω denote the state space; a mass assignment $m\colon 2^{\Omega} \rightarrow [0,1]$ 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 $P(b)$ (0). 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 (0). 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 introduce the concept of *general mass assignment*. Denote the set of all intervals in $[0,1]$ as I, a *general mass assignment (GMA)* $M: 2^{\Omega} \rightarrow I$ 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 designate the set of all probability distributions consistent with M as $\wp(M) = \bigcup_{m \in M} \wp(m)$. The concepts of focal elements and branches for general mass assignments are straightforward generalizations of the definitions for simple mass assignments.

Theoretically, a GMA is just a mass assignment with a slightly modified representation; it can be proven that any GMA can be converted into a mass assignment which represents the same set of probability distributions. Figures 3.d and 3.e show, for example, a GMA and the equivalent mass assignment, respectively. Here ton refers to the amount of goods delivered, and the second focal element of the mass assignment is the union of the focal elements of the GMA.

Adopting the GMA rather than the mass assignment representation to model the world, however, offers significant practical advantages. During the planning process probability intervals often arise due to several factors [3]. First, the planner may not be able to supply the exact probability value for each subset of the state space. For example, the GMA shown in Figure 3.d could perfectly model the case where the planner knows only that the probability of the world being in one of the states represented by the first focal element is not less than 0.8. Second, abstraction approaches frequently induce probability intervals. For example, if we are to abstract away the conditions in the action "pick-up-block" shown in Figure 1.e and group together similar outcomes, we would end up with the abstract action "abs-pick-up" shown in Figure 1.b, whose branches are characterized by probability intervals. Third, some practical projection rules, such as the one we shall introduce in Section , also work with probability intervals. For all these reasons, if we were to represent the world and action effects with mass assignments, we would be forced to frequently perform the conversion from GMA to mass assignment.

Besides high overhead cost, limits on the expressiveness of the action representation language may not allow such conversion. For example, in converting the general mass assignment M shown in Figure 3.d, the union of its focal elements is taken to be the second focal element of the resulting mass assignment m_2. In the language of the DRIPS planner (see Section) this union can not be represented directly and must be replaced by the set $\{fuel = [2,8], ton = [3,4]\}$, which clearly contains more states than the original union. The resulting mass assignment now contains more probability distributions than the original GMA.

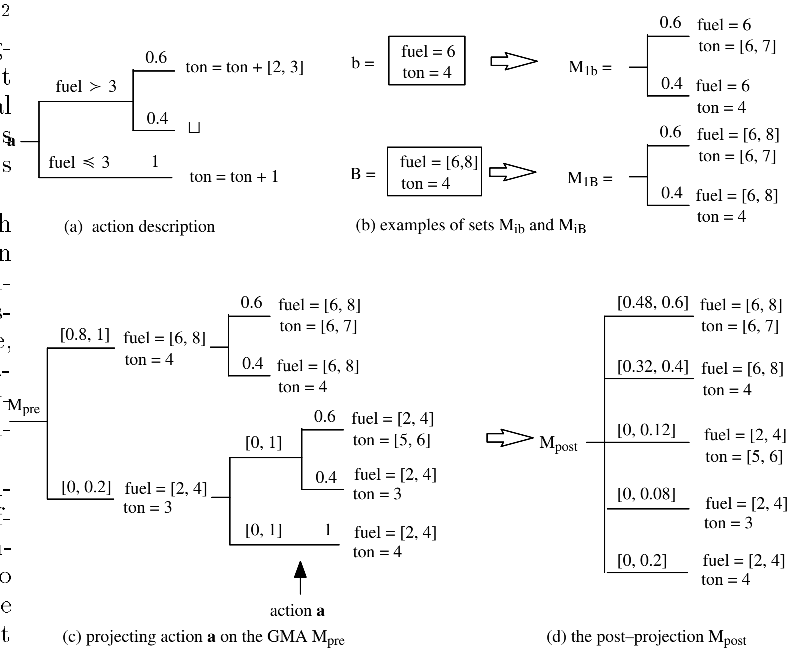


Figure 4: examples of action description, sets M_{ib} and M_{iB}, action projection, and the post-projection GMA.

For all the above reasons, we avoid the conversion by adopting the GMA representation to model uncertainty in the state of the world. A world is modeled with a GMA; the true probability distribution representing the world is among those represented by the GMA.

Action Model

Action Syntax We will be talking about action descriptions, but for brevity will refer to them simply as actions. 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. We retain the action syntax discussed in Section 2 and depicted in Figure 1.a, but replace the p_{ij} with probability intervals I_{ij}. Since metric effects such as $fuel = fuel + [2,3]$ can map a state ($fuel = 2$) into a set of states ($fuel = [4,5]$) we shall 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 GMA M_i whose branches are $\langle I_{ij}, e_{ij} \rangle$; but to be more precise, M_i should be called a functional GMA because its focal elements, effects e_{ij}, are functions in-

[2] Actually a GMA 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] "Probability intervals" here refers to those appearing in the syntax of GMAs and should not be confused with probability intervals induced by a belief function.

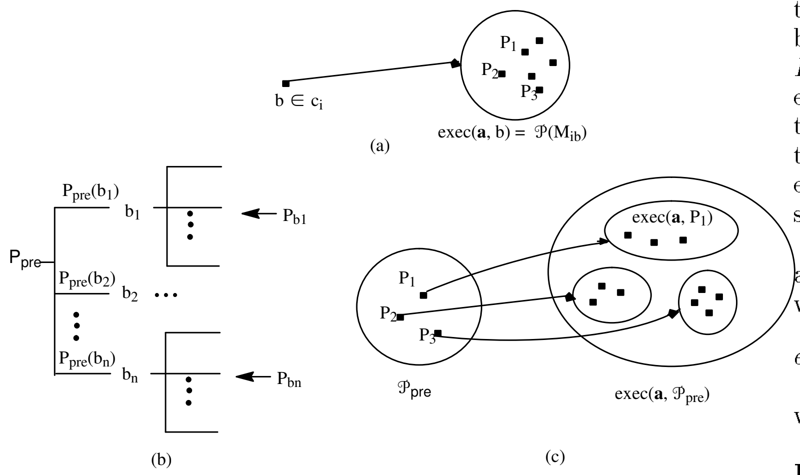


Figure 5: Illustrations on action semantics: pre-execution and post-execution sets.

stead of sets. For a state b and a condition c_i we write M_{ib} to denote the instantiated GMA computed from M_i as follows:

$$\text{for } A \subseteq \Omega \;\; M_{ib}(A) = \begin{cases} I_{ij} & \text{if A} = e_{ij}(\text{b}), \text{ j} = 1, 2, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note the similarity between the definition of M_{ib} and that of P_{ib} in Section 2. For a set $B \subseteq \Omega$ we also define M_{iB} similar to M_{ib} but with b replaced by B in (2). Figure 4.b shows the computed GMAs 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 single probability model, in which executing an action a on a state $b \in c_i$ yields the probability distribution P_{ib} (see Section 2), in our model executing an action a on a state $b \in c_i$ yields a set of probability distributions represented by the GMA 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

$$exec(\mathsf{a}, \wp_{pre}) = \{P_{post} | P_{post} \in exec(\mathsf{a}, P_{pre});\, P_{pre} \in \wp_{pre}\}, \tag{3}$$

which is illustrated in Figure 5.c

Projecting Actions Since a GMA represents a set of probability distributions, if the action a is executed on the world represented by a GMA M_{pre} the set of all possible post-execution distributions will be the set $exec(\mathsf{a}, \wp(M_{pre}))$. 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 a GMA M_{post} such that $\wp(M_{post}) \supseteq exec(\mathsf{a}, \wp(M_{pre}))$. The GMA 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 $M_{iB}(A)$; 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 probabilities 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 a GMA* M_{pre}*,* M_{post} *is calculated as*

$$\forall A \in 2^{\Omega} \;\; M_{post}(A) = \sum_{B \subseteq \Omega} M_{pre}(B) \cdot \left(\sum_i P_I(c_i|B) \cdot M_{iB}(A) \right). \tag{4}$$

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 the GMA M_{pre} depicted in Figure 3.d. 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 GMA 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 GMA M_{post} subsumes the set of distributions resulting from executing the action on the pre-projection GMA M_{pre}, thus proving the correctness of Projection Rule 1.

Theorem 1 *Given $P_{pre} \in \wp(M_{pre})$, action* a*, and $P_{post} \in exec(\mathsf{a}, P_{pre})$, there exists $m_{post} \in M_{post}$, where M_{post} is calculated using (4), such that $P_{post} \in \wp(m_{post})$. This, together with (3), implies that $exec(a, \wp(M_{pre})) \subseteq \wp(M_{post})$.*

For a proof of this theorem, see (0).

Plan Model

Projecting Plans We define a *plan* to be a finite sequence of actions. Semantically, when a plan $p = \mathsf{a}_1\mathsf{a}_2 \cdots \mathsf{a}_n$ (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 $\wp_{post} = exec(\mathsf{a}_n, exec(\mathsf{a}_{n-1}, \cdots, exec(\mathsf{a}_1, \wp_{pre}) \cdots))$. 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 GMA 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 We assume the existence of 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 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 GMA 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 $U_{max}(M_{post})$ ($U_{min}(M_{post})$) then the interval $[U_{min}(M_{post}), U_{max}(M_{post})]$ 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 U_{max} and U_{min} for general mass assignments, using the fractional knapsack greedy algorithm (0).

Theorem 2 *For a general mass assignment M with n branches $\langle I_i, B_i \rangle$ let $U_{max,i} = \max_{b \in B_i} U(b)$ and $U_{min,i} = \min_{b \in B_i} U(b)$ $i = 1, 2, \cdots, n$ then*

$$U_{max}(M) = \max_{\sum_i p_i = 1, p_i \in I_i} \sum_i p_i \cdot U_{max,i} \quad (5)$$

$$U_{min}(M) = \min_{\sum_i p_i = 1, p_i \in I_i} \sum_i p_i \cdot U_{min,i}. \quad (6)$$

According to (0), the greedy algorithm to compute $U_{max}(M)$ and $U_{min}(M)$ 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 $P_I(c_i|B)$ and M_{iB} in (4) must also be done in an effective way. Similarly Equations 5 and 6 in Theorem 2 suggest how a plan's expected utility can be computed efficiently, provided computing the minimum and maximum of the utilities of the states in a set can also 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 (0; 0).

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 GMAs, 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. In projecting a plan computing the probability interval $P_I(c_i|B)$ is a frequent operation. In DRIPS this is done as follows: A checking procedure takes as input the condition c_i and a set of states B, and outputs *yes* if there exists at least a state in B which satisfies c_i; otherwise it outputs *no*. If the checking program returns *no* then clearly $P_I(c_i|B) = 0$. If the checking program returns *yes* then the planner uses the checking procedure to check all other conditions on B; the presence of any more *yes* outputs indicates that not all states in B satisfy c_i, in which case $P_I(c_i|B) = [0, 1]$, otherwise $P_I(c_i|B) = 1$.

Action effects as defined in Section are functions $e_{ij} : \Omega \rightarrow 2^{\Omega}$. In DRIPS 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_{iB} in (4) requires that for any set of states B (represented by a set of constraints of the form $attr_i = V_i$) the sets $e_{ij}(B)$ defined as $\bigcup_{b \in B} e_{ij}(b)$ (Section) be computed in an effective way. This requirement in turn is satisfied if for any function f_i and set of states B as defined above the set $f_i(B) = \bigcup_{b \in B} f_i(b)$ 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 B 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 $f_i(B)$ will be the $f_i(b)$-s with b-s being the minimal and maximal values of $fuel$ in B. We have, therefore, $f_i(B) \subseteq [4 + 2, 7 + 2] = [6, 9]$. All functions f_i implemented so far in DRIPS have countable changes in their monotonicity.

The utility function U (Section) 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 have utility functions with the above property.

Related Work and Discussion

Our framework is most similar to Chrisman's (0). He used mass assignments in their standard form to represent uncertainty about the world. A rule for projecting actions was presented, but its correctness in projecting a plan and its run-time complexity was not discussed. The action model was not implemented.

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

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 (0) 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 (0; 0; 0). Some deal directly with building action models and rules for projecting actions (0). 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, the complexity of the projection rule does not depend on the cardinality of the state space, but grows exponentially with the number of actions in the plan. Section shows that computing terms in (4) 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 GMA 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. Experiments with the DRIPS planner, however, show that the rule is efficient 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. These problems of interest for the medical community 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 (0; 0). It was shown that the branch-and-bound technique frequently used in decision tree analysis cannot handle 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 (0; 0). In one problem with the plan space of 6,206 plans (0; 0) 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 (0). We spent very little efforts building good abstractions, and still obtained a high pruning rate of 48% [5]. The projection rule thus appears to provide a good balance in trade-off between projection complexity and precision.

Experiments in DRIPS also show that the action model provides enough representational power to support abstraction techniques. A theoretical framework on action and plan abstraction in our model has been developed and reported in (0; 0).

Acknowledgments We thank Peter Haddawy, Richard Goodwin, and Lonnie Chrisman for their valuable comments on the topics of this paper.

References

J.S. Blythe. Planning with external events. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, July 1994.

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

L. Chrisman. Personal communications.

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

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

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

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

A. Doan and P. Haddawy. A theoretical framework for abstracting probabilistic actions. In *The Third International Conference on AI Planning Systems*, 1996. To appear.

A.H. Doan and P. Haddawy. Decision-theoretic refinement planning: Principles and application. Technical Report TR-95-01-01, Dept. of Elect. Eng. & Computer Science, University of Wisconsin-Milwaukee, January 1995. Available via anonymous FTP from `pub/tech_reports` at `ftp.cs.uwm.edu`.

Eckman et. al. Foot infections in diabetic patients. *JAMA*, 273(9):712–720, 1995.

K. W. Fertig and J. S. Breese. Probability intervals over influence diagrams. Technical report, Rockwell International Science Center, October 1991.

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

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

I. J. Good. Subjective probability as the measure of a non-measurable set. *Logic, Methodology, and Phylosophy of Science*, pages 319–329, 1962.

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

P. Haddawy, A. Doan, and C.E. Kahn. Decision-theoretic refinement planning in medical decision making: Management of acute deep venous thrombosis. Submitted to Journal of Medical Decision Making. Available via www at http://www.cs.uwm.edu/faculty/haddawy.

[5]The domain code and a report of the experiments can be obtained by sending request to the author.

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

P. Haddawy and M. Suwandi. Decision-theoretic refinement planning using inheritance abstraction. In *Proceedings, Second International Conference on AI Planning Systems*, pages 266–271, June 1994.

J.Y. Halpern. Two views of belief: Belief as generalized probability and belief as evidence. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 112–119, Boston, July 1990.

S. Hanks. Practical temporal projection. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 158–163, Boston, July 1990.

Hillner BE, Philbrick JT, Becker DM. Optimal management of suspected lower-extremity deep vein thrombosis: an evaluation with cost assessment of 24 management strategies. *Arch Intern Med*, 152:165–175, 1992.

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

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

M. Pittarelli. Anytime decision making with imprecise probabilities. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 470–477, Seattle, July 1994.

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

Glenn Shafer. A mathematical theory of evidence. 1976.

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

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

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

B. Tessem. Interval probability propagation. *International Journal of Approximate Reasoning*, 7:95–120, 1992.

S. Thiebaux, J. Hertzberg, W. Shoaff, and M. Schneider. A stochastic model of actions and plans for anytime planning under uncertainty. *International Journal of Intelligent Systems*, 1994. To appear.