Decision-Theoretic Refinement Planning using Inheritance Abstraction

Peter Haddawy * **Meliani Suwandi**
{*Haddawy, Suwandi*}*@cs.uwm.edu*

Department of Electrical Engineering and Computer Science
University of Wisconsin-Milwaukee
PO Box 784
Milwaukee, WI 53201

Introduction

Given a probabilistic model of the world and of available actions and a utility function representing the planner's objectives we wish to find the plan that maximizes expected utility. Finding the optimal plan requires comparing the expected utilities of all possible plans. Doing this explicitly would be computationally prohibitive in all but the smallest of domains. Thus we must find a way to compare partial plans in such a way that we can eliminate some partial plans before fully elaborating them. Such comparisons require a definition of partial plan that allows us to determine that all completions of one plan are less preferred than all completions of another. We achieve such a definition of partial plan by structuring actions into an abstraction hierarchy and by restricting the planner to using only refinement operators. A partial plan is then a plan in which some of the actions are abstract. An abstract plan's outcomes are sets of outcomes of more concrete plans. Since different probability and utility values may be associated with each specific outcome, in general a probability range and a utility range will be associated with each abstract outcome. Thus the expected utility of an abstract plan is represented by an interval, which includes the expected utilities of all possible instantiations of that abstract plan. Refining the plan, i.e., instantiating one of its actions, tends to narrow the interval. When the expected utility intervals of two plans do not overlap, the one with the lower interval can be eliminated.[1]

We present a method for abstracting probabilistic conditional actions and we show how to compute expected utility bounds for plans containing such abstract actions. We present a planning algorithm that searches through the space of possible plans by building abstract plans, comparing them, and refining only those that might be refinable to the optimal plan. The algorithm is guaranteed to find the optimal plan and with an appropriate abstraction hierarchy has a complexity which is exponentially better than that of exhaustive enumeration. The planning algorithm has been implemented as the DRIPS decision-theoretic refinement planning system. The system reasons with a probabilistic temporal world model. It maximizes expected utility relative to partially satisfiable deadline and maintenance goals, as well as the resources consumed in achieving them. It can reason about action effects involving both symbolic and quantitative attributes. Actions that change the state of the world and actions involving observations are treated in a uniform manner.

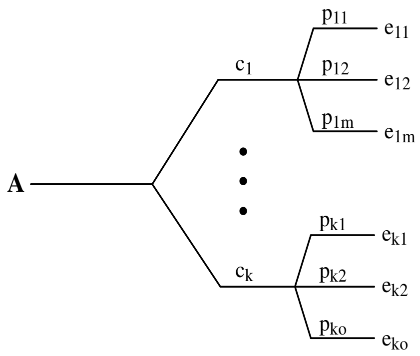

Figure 1: General form of an action description.

The Representation

Action Model Actions are both conditional and probabilistic: under certain conditions an action will have a given effect with a given probability. Action effects are represented in terms of conditional probabilities. An action is depicted with a tree structure as shown in figure 1, where the c_i are a set of mutually exclusive and exhaustive conditions, the p_{ij} are probabilities, and the e_{ij} are effects. If one of the conditions

---

*This work was inspired by discussions with Steve Hanks. Manonton Butarbutar performed the complexity analysis of the algorithm. This work was partially supported by NSF grant #IRI-9207262.

[1] If the upper bound on the expected utility of each abstract plan is tight, i.e., there is an instance of the abstract plan with that expected utility, then at each refinement step we can eliminate all abstract plans but the one with the highest upper bound.

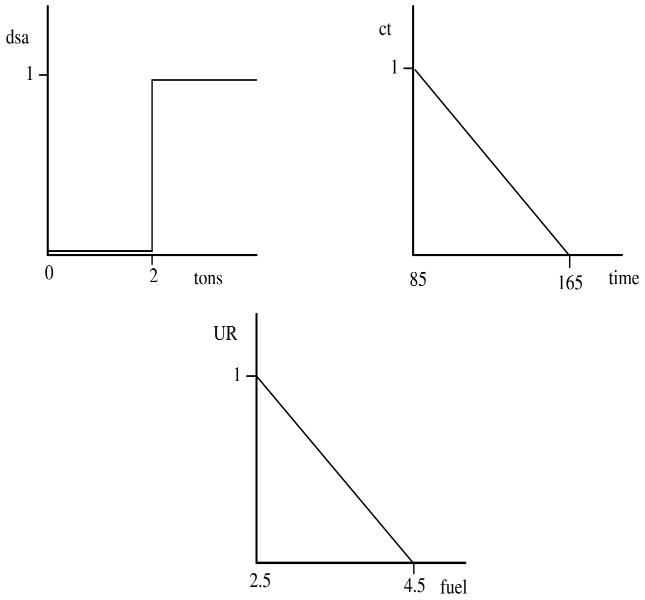

Figure 2: Specification of delivery utility function.

holds at the beginning time of the action then the effects on that branch are realized immediately following the action, with the specified probabilities. Each branch is a conditional probability statement. For example, the first branch means that $P(e_{11}|A \wedge c_1) = p_{11}$. This representation is similar to that used by Hanks (Hanks 1990).

World Model The state of the world is represented with a probability distribution over a set of attributes. Attributes may be symbolic or quantitative. For simplicity, we assume that all the attributes are probabilistically independent. So the joint distribution is just the product of the probabilities of the attributes. We assume that changes to the world are limited to those effects explicitly described in the agent's action descriptions, so we do not allow for exogenous events.

Utility Model We use the goal-directed utility model for deadline and maintenance goals described in (Haddawy & Hanks 1992; 1993). Due to space limitations, we will only discuss deadline goals in this paper. A deadline goal, such as delivering two tons of tomatoes to the warehouse within 85 minutes, is considered to consist of a temporal and an atemporal component. The atemporal component indicates what is to be achieved and the temporal component indicates when it is to be achieved. Our model allows goals to be partially satisfied. The example deadline goal above could be partially satisfied in two ways: we can partially satisfy the atemporal component by delivering less than two tons of tomatoes and we can partially satisfy the temporal component by making our delivery after the deadline. Of course, we can partially satisfy both components by making several small deliveries both before and after the deadline. A utility function relative to such a goal is specified in terms of a degree of satisfaction function for the atemporal component (DSA) and a temporal coefficient (CT). The DSA function indicates to what degree the atemporal component is satisfied at a time in a chronicle and the temporal coefficient is a discount factor that penalizes the planning agent for missing the deadline. The utility of a chronicle relative to a goal is computed by combining the two functions. This is done by summing the DSA at the deadline with the changes in DSA after the deadline, weighted by the temporal coefficient. We represent the costs associated with attempting to achieve a goal with a residual utility function UR. Assuming that the goal and residual utility are independent, the overall utility of a chronicle is the sum of the goal and residual utilities.

Figure 2 shows some possible utility functions for our tomato delivery example. If we only obtain benefit from having all tomatoes delivered, the DSA function would be a step function with value zero below two tons and value one above two tons. If we can obtain benefit from deliveries that miss the deadline we might represent the temporal coefficient as a linear function that has value one at 85 minutes and value zero at say 165 minutes. If our only cost is the amount of fuel consumed then the residual utility might be a linear function with value one at 2.5 gallons consumption and value zero at 4.5 gallons. If a unit of goal utility is worth .02 units of residual utility, we could represent the overall utility as $\mathbf{U}(c) = \mathsf{UG}(c) + (.02)\mathsf{UR}(c)$.

The Planning Task A planning problem is described in terms of an initial state distribution, a set of action descriptions, and a utility function. The action descriptions are organized into an abstraction/decomposition network, which is described later. A plan is a sequence of actions. So the planning task is to find the sequence of actions that maximizes expected utility relative to the given probability and utility models.

Abstracting Actions

We extend Tenenberg's (Tenenberg 1991) notion of inheritance abstraction for STRIPS operators to apply to conditional probabilistic actions. As Tenenberg explains it, "the intent of using inheritance abstraction is to formalize the notion that analogous action types can be structured together into an action class at the abstract level, characterized by the common features of all elements of the class." Thus we can plan with the abstract action and infer properties of a plan involving any of the instances of the abstract action.

We abstract actions by grouping their outcomes. If one action has an outcome e_1 and another action has an outcome e_2 then we can produce an abstract action

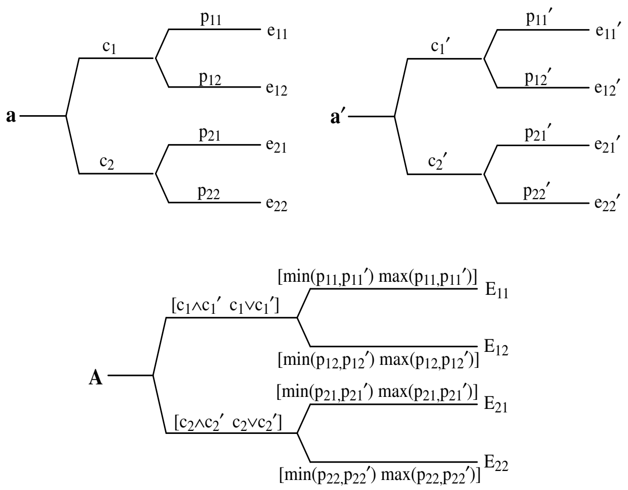

Figure 3: Abstracting probabilistic conditional actions.

with outcome E such that E is consistent with both e_1 and e_2. Consider the two actions $\mathbf{a}$ and $\mathbf{a}'$ shown in figure 3. Suppose we wish to produce an abstract action $\mathbf{A}$ by grouping topologically corresponding branches. Then under what conditions and with what probabilities will effect E_{11} be realized? First we consider the conditions under which it will be realized. It will certainly be realized with some probability in any world in which both condition c_1 and c_1' occur. It could be realized with some probability in any world in which c_1 or c_1' occurs. Thus the range of worlds in which E_{11} is realized with some probability is described by $[c_1 \wedge c_1' \quad c_1 \vee c_1']$. Notice that the set of models satisfying $c_1 \wedge c_1'$ is a strict subset of the set of models satisfying $c_1 \vee c_1'$, so the range is well defined. Now we must determine with what probability E_{11} is realized in such worlds. If action $\mathbf{a}$ is chosen as the instantiation, it is realized with probability p_{11} and if action $\mathbf{a}'$ is chosen it is realized with probability p_{11}'. So it is realized with a probability in the range $[p_{11} \quad p_{11}']$. Thus the probability that effect E_{11} is realized along the first branch is at least $P(c_1 \wedge c_1') \cdot min(p_{11}, p_{11}')$ and at most $P(c_1 \vee c_1') \cdot max(p_{11}, p_{11}')$.

Our method of representing abstract actions is similar to that of Chrisman (1992). He represents the uncertainty in the effects of an abstract action by using Dempster-Shafer intervals. He derives a closed-form projection rule that works by finding the convex hull of possible poststate probability distributions. Although his actions can include conditions, he does not show how to abstract the conditions.

Evaluating Plans

The outcome of a plan is a probability distribution over a set of chronicles. We need to compute this distribution from the initial state distribution and the action descriptions. We make the Markovian assumption that the conditional probabilities of an action's effects given the action and the conditions in its description are independent of all other conditions at the same time or earlier and all other previous actions. We also assume that the conditions determining the effects of an action are probabilistically independent of the action. Under these assumptions, we can compute the outcome distribution of an action by multiplying the conditional probabilities describing the action effects by the probabilities of the conditions. Projecting a plan produces a probability distribution over a future-branching tree of chronicles.

Projecting an abstract plan will result in a set of abstract chronicles characterized by duration ranges and attribute ranges after each action in the plan, as well as a range on the probability of each chronicle. In order to compute expected utility bounds for abstract plans, we need to be able to translate the attribute and duration ranges into a range for the utility of each abstract chronicle. The constraints our model places on the form of the utility function permit us to compute this utility range relatively easily. The decomposition of goal utility into temporal and atemporal components permits us to maximize utility by individually maximizing DSA and CT. Since CT is a nonincreasing function of time, we can maximize utility by minimizing the durations of all actions. So at each time point where the values of relevant attributes change we simply choose duration and attribute values that maximize the CT and DSA functions. Hence for our delivery example we maximize utility by choosing the earliest possible delivery times and largest possible delivery amounts. We'll assume that DSA and UR are monotonic, so we can compute the utility range by computing the utilities for the upper and lower bounds of the attribute ranges.

The Planner

We describe the DRIPS planning system by demonstrating how it handles an example problem. Suppose we wish to generate the best plan for delivering two tons of tomatoes from a farm to a warehouse within 85 minutes[2] and suppose we use the example utility function presented earlier. The delivery plan will consist of driving a truck from the depot to the farm, loading the truck, and driving the loaded truck to the warehouse. Planning is the process of generating this plan as well as choosing among the various ways this plan can be realized in such a way that expected utility is maximized. The descriptions of the available actions are shown in Figure 4. The leaves of the trees are labeled with the outcomes of the actions. Deterministic actions are labeled with a single outcome. Outcomes are described

[2]For simplicity of exposition our example includes only quantitative attributes but DRIPS is capable of reasoning about symbolic attributes as well.

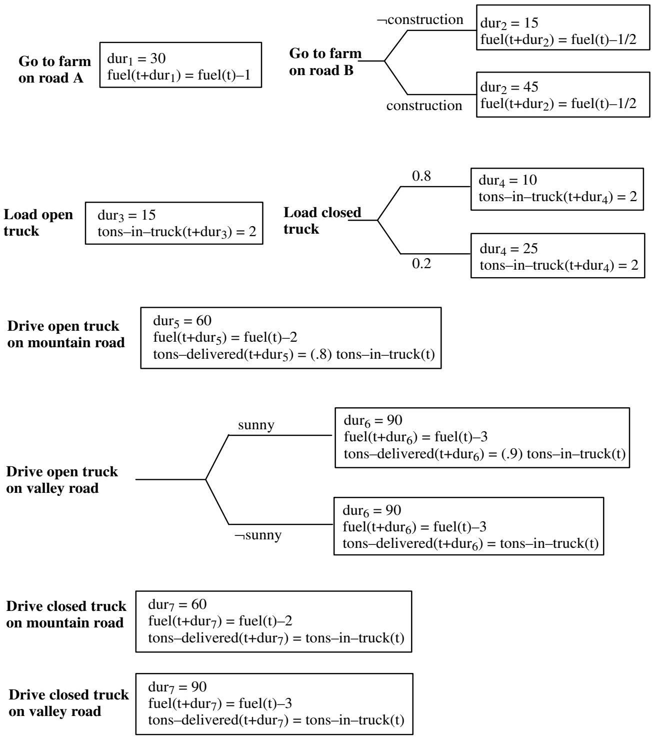

Figure 4: Action descriptions for the delivery example.

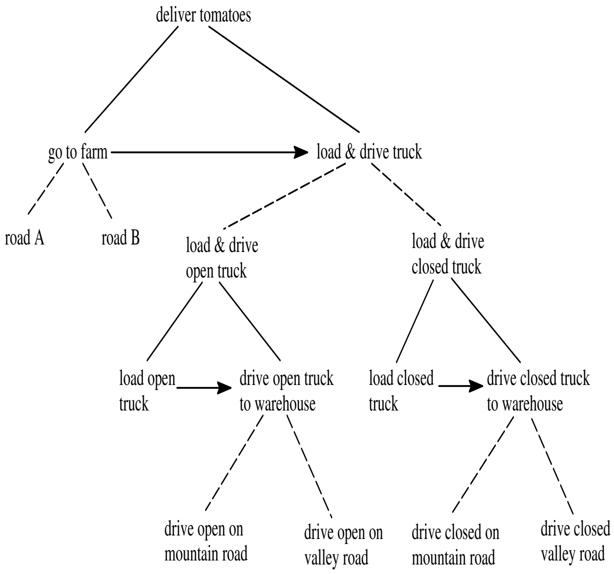

Figure 5: Abstraction/decomposition network.

in terms of a duration, as well as any changes in attribute values. Attributes are represented as functions of time. The variable t represents the beginning time of the action, making the action descriptions temporally indexical.

There are two possible routes we can take from the depot to the farm: road A and road B. Road A is longer but has no delays, while travel along road B might be delayed due to construction. The probability that construction is taking place is 0.2. These options are represented by the first two action descriptions in Figure 4.

Once at the farm we must load the truck. We have two trucks at the depot to choose from: an open truck and a closed, cushioned truck. The open truck is easy to load, while there is an 80% chance the closed truck can be loaded quickly and a 20% chance that loading it will take longer. The next two diagrams in Figure 4 depict these two actions.

Once the truck is loaded we must drive it to the warehouse. We have two routes to choose from: the mountain road and the valley road. The mountain road is shorter but bumpy. If we drive the open truck on the mountain road, the bottom 20% of the tomatoes will be crushed. If we drive the open truck on the valley road and the sun is shining, the top 10% of the tomatoes will be spoiled by the sun. This combination of options results in the last four action descriptions in Figure 4.

The DRIPS system searches for the optimal plan using an abstraction/decomposition network which describes the space of possible plans and their abstractions. The network for this problem is shown in Figure 5. Solid links show decompositions, while dashed links show possible refinements. For example the task "deliver tomatoes" is decomposed into the sequence of the two actions "go to farm" and "load & drive truck". The arrow between the actions signifies temporal succession. The abstract action "go to farm" can be realized by driving on road A or road B.

Descriptions of the abstract actions are shown in Figure 6. A set of actions is abstracted by grouping together their outcomes into abstract outcomes which represent the range of possible outcomes of the instantiations. Care must be taken to group together similar outcomes. For example, the "drive open truck" action is an abstraction of "drive open truck on mountain road" and "drive open truck on valley road." Since the single outcome of "drive open truck on mountain road" is more similar to the outcome of the upper branch of "drive open truck on valley road" than to the outcome of the lower branch, it is grouped with the former. The outcomes of the abstract action are now specified simply as the range of outcomes grouped together.

Given this representation of the planning problem, we evaluate plans at the abstract level, eliminate suboptimal plans, and refine remaining candidate plans further. The algorithm works as follows. A queue contains all actions that might need to be refined. A list called `plans` contains all candidate plans.

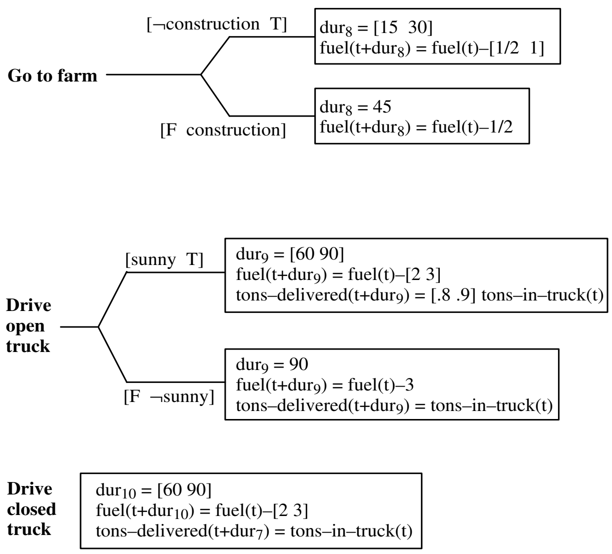

Figure 6: Abstract action descriptions.

Procedure:
Create a plan consisting of the single top-level action and put it in plans.
Put the action on the queue.
Until the queue is empty do:

Take the first action from the queue.
If the action does not appear in any plans or is a concrete action, do nothing.
Else it is an abstract action, so add the instantiations of the action to the queue and

Until no plans contain the action do

- Replace a plan in which the action appears with one plan for every possible instantiation of the action.
- Replace the instantiations with their decompositions.
- Compute the expected utility range of each new plan.
- Remove suboptimal plans from plans.

Return plans.

Eight possible plans are implicitly encoded in the abstraction/decomposition network, and we want to choose the one that maximizes expected utility. According to the network, the task of delivering tomatoes is first decomposed into going to the farm and loading, then driving the truck. The planner attempts to choose an instantiation of the "go to farm" action first. Because the utility function is a function over chronicles, the value of a particular action in a plan depends on when in the plan it occurs, so options can only be evaluated in the context of an overall plan. Consequently, we combine the "load & drive open truck" and the "load & drive closed truck" actions with the "go to farm" action to obtain a complete abstract plan that can be evaluated.

$\mathcal{P}_1$: Go to farm $\longrightarrow$ Load & drive open truck					
chron	time	fuel	tons	U(chron)	prob
c_1	[90 135]	[2.5 4]	[1.6 1.8]	[.005 .02]	[.56 1]
c_2	[120 135]	[3.5 4]	2	[.380 .5725]	[0 .3]
c_3	[120 150]	[2.5 3.5]	[1.6 1.8]	[.01 .02]	[0 .2]
c_4	150	3.5	2	.1975	[0 .06]
$\mathcal{P}_2$: Go to farm $\longrightarrow$ Load & drive closed truck					
chron	time	fuel	tons	U(chron)	prob
c_1	[85 130]	[2.5 4]	2	[.4425 1.02]	[.64 .8]
c_2	[100 145]	[2.5 4]	2	[.255 .8325]	[.16 .2]
c_3	[115 145]	[2.5 3.5]	2	[.255 .635]	[0 .16]
c_4	[130 160]	[2.5 3.5]	2	[.0675 .4475]	[0 .04]

Table 1: Highest level abstract plan outcomes.

Each abstract plan results in four chronicles since in each plan two of the actions have two possible outcomes. For example, for plan $\mathcal{P}_1$ we obtain these chronicles by concatenating the "go to farm," "load open truck," and "drive open truck" action descriptions. Rather than showing the complete chronicles, the relevant attributes of the resulting chronicles are summarized in Table 1, which shows for each chronicle the range of times, fuel consumption, and tons of tomatos delivered. The time refers to the time that the delivery is made. We assume that the plan begins execution at time zero and we take the beginning time of an action to be the beginning time of the previous action plus its duration.

Now we need to compute the utility range for each chronicle. For example, consider computing the utility range for chronicle c_2 of plan $\mathcal{P}_1$. Let UG_{min} and UG_{max} be the lower and upper bounds on the goal utility, respectively, and let UR_{min} and UR_{max} be the lower and upper bounds on residual utility, respectively. Then

$$\mathsf{UG}_{max}(c_2) = dsa(2) \cdot ct(120) = 0.5625$$
$$\mathsf{UG}_{min}(c_2) = dsa(2) \cdot ct(135) = 0.375$$
$$\mathsf{UR}_{min}(c_2) = 0.25, \;\; \mathsf{UR}_{max}(c_2) = 0.5$$

Since our global utility function is $\mathbf{U}(c) = \mathsf{UG}(c) + (0.02)\mathsf{UR}(c)$, we have $\mathbf{U}(c_2) = [0.380\ 0.5725]$.

Using the utility and probability information in the table we can compute the expected utility bounds for each plan. Computing a bound for a plan involves choosing the probabilities within the given ranges that minimize and maximize the expected utility. We can do so by solving a small linear programming problem in which the objective function is the expression for the expected utility and the constraints are the probability bounds. For plan $\mathcal{P}_1$ and $\mathcal{P}_2$ we obtain the expected utility ranges $\mathbf{EU}(\mathcal{P}_1) = [.005\ .1964]$ and $\mathbf{EU}(\mathcal{P}_2) = [.3673\ .9825]$. Since the lower bound for $\mathcal{P}_2$ is greater than the upper bound for $\mathcal{P}_1$, we can eliminate from consideration all possible refinements of $\mathcal{P}_1$ and concentrate on refining $\mathcal{P}_2$. So at this point we have chosen the option of using the closed truck. By making

$\mathcal{P}_{2.1}$: Go to farm $\longrightarrow$ Load closed $\longrightarrow$ Drive closed on mt.rd.					
chron	time	fuel	tons	U(chron)	prob
c_1	[85 100]	[2.5 3]	2	[.8275 1.02]	[.64 .8]
c_2	[100 115]	[2.5 3]	2	[.64 .8325]	[.16 .2]
c_3	115	2.5	2	.645	[0 .16]
c_4	130	2.5	2	.4575	[0 .04]
$\mathcal{P}_{2.2}$: Go to farm $\longrightarrow$ Load closed $\longrightarrow$ Drive closed on valley rd.					
chron	time	fuel	tons	U(chron)	prob
c_1	[115 130]	[3.5 4]	2	[.4425 .635]	[.64 .8]
c_2	[130 145]	[3.5 4]	2	[.255 .4475]	[.16 .2]
c_3	145	3.5	2	.26	[0 .16]
c_4	160	3.5	2	.0725	[0 .04]

Table 2: Intermediate level abstract plan outcomes.

this choice, we have pruned away the left-hand subnetwork underneath the "load & drive truck" node in Figure 5, resulting in pruning half the space of possible plans from consideration.

We are left with two more actions to refine: "go to farm" and "drive closed truck to warehouse." The planner chooses to refine the drive action. Again the instantiations involving the mountain road and the valley road must be evaluated in the context of a complete plan. So we compose the descriptions of the concrete actions "drive closed on mountain road" and "drive closed on valley road" with the descriptions of the concrete action "load closed truck" and the abstract action "go to farm." Table 2 summarizes the outcomes for the two alternative plans. We use this information to compute expected-utility bounds for the two alternatives: $\mathbf{EU}(\mathcal{P}_{2.1}) = [.7533\ .9825]$ and $\mathbf{EU}(\mathcal{P}_{2.2}) = [.3683\ .5975]$. Notice that the EU intervals for the two plans are contained in the EU interval for the abstract plan of which they are a refinement. Since the lower bound for plan $\mathcal{P}_{2.1}$ is greater than the upper bound for plan $\mathcal{P}_{2.2}$, we can eliminate $\mathcal{P}_{2.2}$ from consideration, pruning away two more possible concrete plans. By eliminating plan $\mathcal{P}_{2.2}$, we have chosen to take the mountain road.

Finally we refine plan $\mathcal{P}_{2.1}$. Our two options are taking either road A or road B to the farm. Since the plans now include only concrete actions, the attribute, probability, and utility values are now point values. The expected utilities of the two remaining plans are $\mathbf{EU}(\mathcal{P}_{2.1.1}) = .79$ $\mathbf{EU}(\mathcal{P}_{2.1.2}) = .9075$ so we choose plan $\mathcal{P}_{2.1.2}$, which is "go to farm on road B", "load closed truck", and "drive closed truck on mountain road". Since this is a complete concrete plan, we have generated the plan that maximizes expected utility and are finished.

Notice that under time constraints we could stop the planner before it runs to completion and choose any instantiation from the set of abstract plans not yet shown to be suboptimal. Since we have expected utility ranges for all abstract plans, we can bound the possible degree to which we are sacrificing optimality in choosing that instantiation.

Complexity Analysis

Suppose we have an abstraction/decomposition network with p actions in each decomposition, n possible instantiations of each abstract action, and k levels of abstraction. This network contains $n^{(p+p^2+p^3+\ldots+p^k)}$ possible concrete plans. Now suppose that x is the percentage of the plans remaining at each abstraction level after pruning. The number of plans (abstract and concrete) for which the DRIPS algorithm computes expected utility is

$$n + xn^2 + x^2n^3 + \ldots + x^{p+p^2+\ldots+p^k-1}n^{p+p^2+\ldots+p^k}.$$

With maximum pruning ($xn = 1$) the number of plans examined is only

$$n + xn \times n + (xn)^2 \times n + \ldots + (xn)^{p+p^2+\ldots+p^k-1} \times n = n(p + p^2 + \ldots + p^k),$$

a logarithmic reduction in complexity.

Conclusions

We have presented a decision-theoretic planner that reduces the complexity of planning by using inheritance abstraction. The planner finds the expected utility-maximizing plan by repeatedly instantiating abstract actions, computing expected utility, and pruning suboptimal classes of plans. The system reasons with a rich domain representation that includes time and both symbolic and numeric attributes. The planner finds plans to achieve partially satisfiable deadline and maintenance goals. Because the planning algorithm works by eliminating suboptimal plans, it can be stopped at any time to obtain the current set of possibly optimal plans.

References

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

Haddawy, P., and Hanks, S. 1992. Representations for decision-theoretic planning: Utility functions for deadline goals. In Nebel, B.; Rich, C.; and Swartout, W., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*. San Mateo, CA: Morgan Kaufmann. 71–82.

Haddawy, P., and Hanks, S. 1993. Utility models for goal-directed decision-theoretic planners. Technical Report 93-06-04, Department of Computer Science and Engineering, University of Washington. Available via anonymous FTP from `~ftp/pub/ai/` at `cs.washington.edu`.

Hanks, S. 1990. *Projecting Plans for Uncertain Worlds*. Ph.D. Dissertation, Yale University.

Tenenberg, J. 1991. *Reasoning About Plans*. San Mateo, CA: Morgan Kaufmann. 213–283.