Decison-Theoretic Refinement Planning using Inheritance Abstraction

Peter Haddawy * **Meliani Suwandi**

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

1 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 on 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.

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 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.

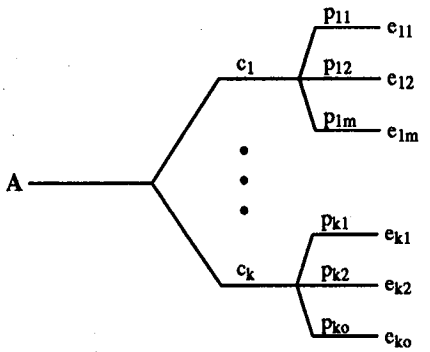

Figure 1: General form of an action description.

2 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. Each branch is a conditional probability statement. For example, the first branch means that $P(e_{11} \mid A \wedge c_1) = p_{11}$. This representation is similar to that used by Hanks [Hanks, 1990b]. We assume that an action's effects are realized at the instant following the action.

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

*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.

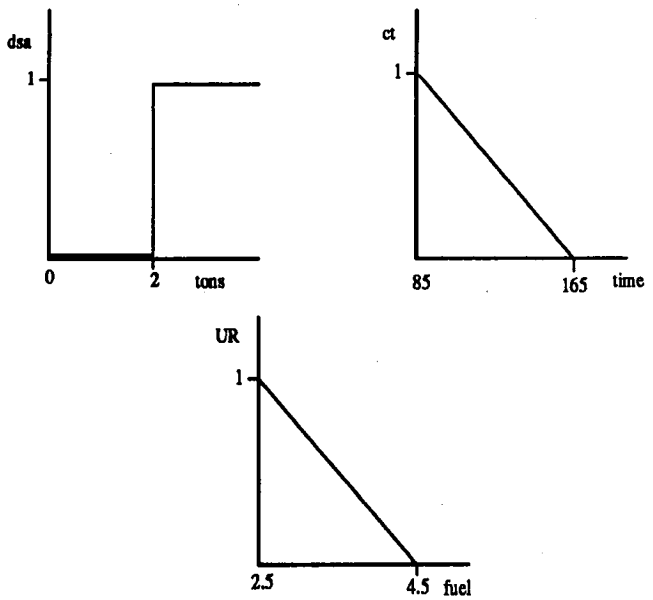

Figure 2: Specification of delivery utility function.

that changes to the world are limited to those effects explicitly described in the agent's action descriptions. We do not allow for exogenous events.

Utility Model We use the goal-directed utility model for deadline goals described in [Haddawy and Hanks, 1992, Haddawy and Hanks, 1993]. 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 example deadline goal 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

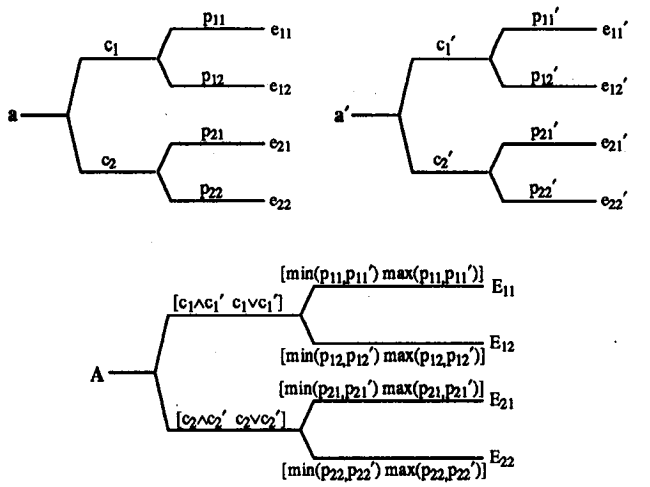

Figure 3: Abstracting probabilistic conditional actions.

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 in section 5. 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.

3 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 with outcome E such that E is consistent with both e_1 and e_2. Consider the two actions **a** and **a'** shown in figure 3. Suppose we wish to produce an abstract action **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 $[c_1 \wedge c_1' \;\; c_1 \vee c_1']$. 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 **a** is chosen as the instantiation, it is realized with probability p_{11} and if action **a'** is chosen it is realized with probability p_{11}'. So it is realized with a probability in the range $[p_{11} \;\; p_{11}']$. 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.

4 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. In practice, we compute the set of outcome chronicles of a plan by starting with a set of worlds with probabilities assigned to them and by projecting the action in each of those worlds. In each world the action produces the resulting distribution associated with the conditions that are true in that world. Thus if actions have probabilistic effects, as we project a plan forward the set of worlds increases, i.e. we generate a future-branching set of chronicles. We need to distinguish the separate chronicles since utility is a function of attribute values over time.

Projecting an abstract plan will result in a set of 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 the chronicle. In order to do this we must make some assumptions concerning the form of the DSA function. For quantitative attributes we assume that DSA is a monotonic function of the quantity. So for each outcome, either the upper bound or the lower bound of the values corresponds to the upper bound of the utility of the chronicle, and the other one corresponds to the lower bound of the utility of the chronicle. For deadline goals, achieving the goal at the earliest time is preferred, and so has the highest utility. Thus the maximum value of the utility corresponds to the lower bound of the range of time, and the minimum value to the upper bound.

5 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[1] and suppose we use the example utility function from section 2. 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 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 the 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 the 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 the 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

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

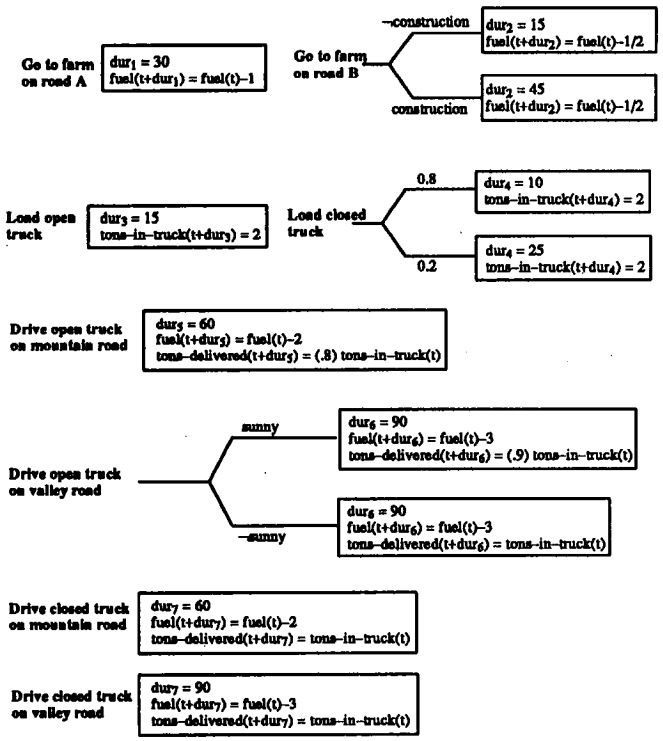

Figure 4: Action descriptions for the delivery example.

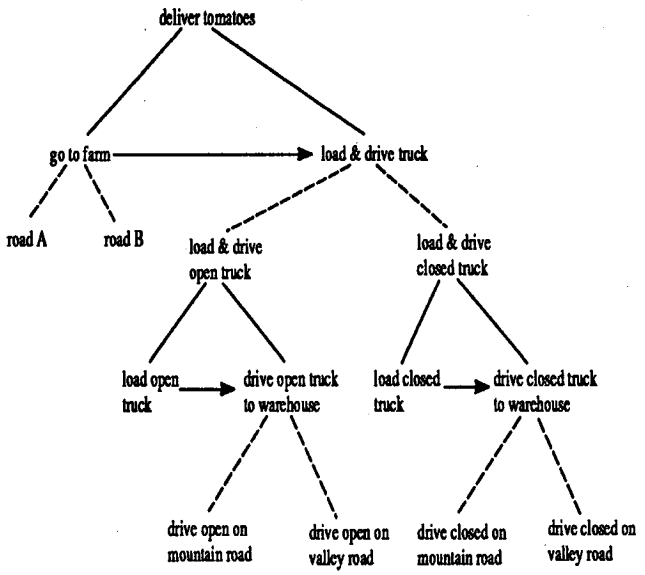

Figure 5: Abstraction/decomposition network.

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. Since both elements of a decomposition must be realized, decomposition links are AND links and since either element of an instantiation may be used, instantiation links are OR links. So this network forms an AND/OR tree.

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. While similarity is fairly clear in this case, determining similarity of outcomes with multiple attributes may not be straightforward in general. (In fact, one may wish to produce more than one abstraction based on different similarity measures and try each one on a given problem. One would use the hierarchy that resulted in the most pruning for the given problem.) 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. The algorithm uses two primary data structures. A queue contains all actions that might need to be refined. A list called plans contains all candidate plans.

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

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

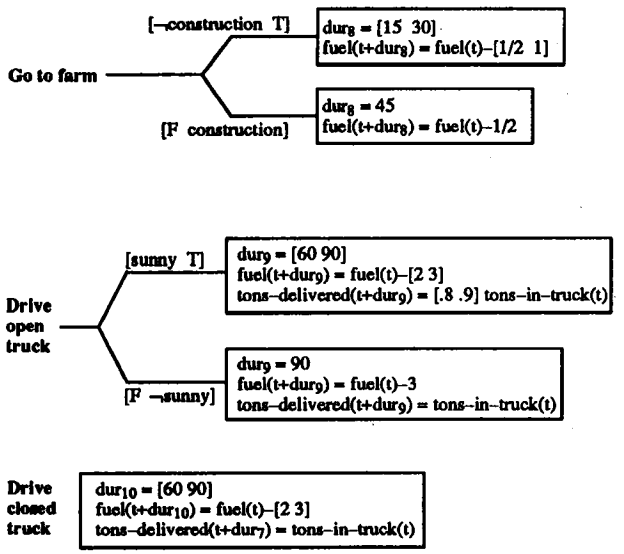

Figure 6: Abstract action descriptions.

$\mathcal{P}_1$: Go to farm ⟶ 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 ⟶ 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.

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.

Each abstract plan results in four chronicles (Table 1). 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. Doing so results in four chronicles, since "go to farm" and "drive open truck" each have two branches. 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. So for example the fuel consumption for chronicle c_1 is computed according to

$$fuel(dur_8) = fuel(0) - [.5\ 1]$$
$$fuel(dur_8 + dur_9) = fuel(dur_8) - [2\ 3]$$

so

$$fuel(dur_8 + dur_9) = fuel(0) - [2.5\ 4]$$

We illustrate how the utility range is computed by showing the computation for chronicle c_2 of plan $\mathcal{P}_1$. The computations for the other chronicles are similar. 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. We minimize the utility of the abstract chronicle by choosing the latest delivery times and the smallest delivery amounts and maximize the utility by choosing the earliest delivery times and the largest amounts. So

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

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 example, the upper bound for plan $\mathcal{P}_1$ can be computed by maximizing the objective function

$$.02p_1 + .5725p_2 + .02p_3 + .1975p_4$$

subject to the constraints

$$.56 \leq p_1 \leq 1,\ 0 \leq p_2 \leq .3,$$
$$0 \leq p_3 \leq .2,\ 0 \leq p_4 \leq .06,$$
$$p_1 + p_2 + p_3 + p_4 = 1$$

The maximizing probabilities are

$$p_1 = .56\ \ p_2 = .3\ \ p_3 = .08\ \ p_4 = .06$$

So the upper bound on expected utility is

$$(.56)(.02)+(.3)(.5725)+(.08)(.02)+(.06)(.1975) = .1964$$

So 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 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.

$\mathcal{P}_{2.1}$: Go to farm ⟶ Load closed ⟶ 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 ⟶ Load closed ⟶ 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.

$\mathcal{P}_{2.1.1}$: Go on A ⟶ Load closed ⟶ Drive closed on mt.rd.					
chron	time	fuel	tons	U(chron)	prob
c_1	100	3	2	.8275	.8
c_2	115	3	2	.64	.2
$\mathcal{P}_{2.1.2}$: Go on B ⟶ Load closed ⟶ Drive closed on mt. rd.					
chron	time	fuel	tons	U(chron)	prob
c_1	85	2.5	2	1.02	.64
c_2	100	2.5	2	.8325	.16
c_3	115	2.5	2	.645	.16
c_4	130	2.5	2	.4575	.04

Table 3: Concrete plan outcomes.

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. The outcomes of the plans incorporating these options are summarized in table 3. 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.

6 Complexity Analysis

The complexity of decision-theoretic refinement planning is a function of two factors: the number of plans and the length of plans. The DRIPS approach reduces complexity as a function of the number of plans. 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, which is a huge number even for small values of n, p, and k. For example if we have only three instantiations of each abstract action, four actions in each decomposition, and three levels of abstraction then the network contains 3^{84} possible concrete plans. Now suppose that x is the percentage of the plans remaining at each abstraction level after pruning. For a poorly chosen hierarchy, the value of x is 100%, i.e., none of the plans are pruned, and for an ideally chosen hierarchy, the value of x times the number of current plans is 1, i.e., at each abstraction level all abstract plans are pruned except one. 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. With no pruning ($xn = n$) the number of plans examined is

$$n + n^2 + n^3 + \ldots + n^{p+p^2+\ldots+p^k}$$

which is more than the number examined by simply exhaustively enumerating all plans since we examine the abstract plans as well as the concrete ones. But an asymptotic analysis shows that this number is not significantly greater than the total number of concrete plans when the number of plans is very large. If u is the number of possible concrete plans represented in the network then with maximum pruning the total number of plans examined by the DRIPS algorithm is $O(\log(u))$. With no pruning the total number of plans examined is $O(u)$. This is to say that using abstraction we reduce the complexity logarithmically for the best case, while we retain the same complexity for the worst case.

7 Conclusions and Future Research

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 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.

DRIPS is intended to be a practical planning system for solving large real-world planning problems. To test its usefulness, we are applying it to the problem of planning the brewing of beer. Our current domain model contains approximately 2^{30} possible plans and we anticipate that the final model will be significantly larger.

In addition to applying DRIPS to large problems, we are interested in extending the scope of problems that DRIPS can solve, as well as making it easier to use. We are currently adding the capability to plan for partially satisfiable maintenance goals [Haddawy and Hanks, 1993]. A maintenance goal specifies that a condition should be maintained over a period of time. Adding this capability only requires adding the ability to compute expected utility ranges for the new utility functions.

The efficiency of DRIPS depends on how well the abstraction hierarchy is suited to the utility function for which a plan is being generated. In the worst case, the algorithm performs more work than that involved in exhaustively enumerating all possible plans. We are examining ways in which the abstraction hierarchy can be dynamically tailored to each given utility function by indexing off of certain characteristics of the utility function.

The DRIPS algorithm reduces planning complexity as a function of the number of plans by using abstraction. Abstraction can also be used to reduce complexity as a function of plan length. The complexity of planning under uncertainty increases exponentially as a function of plan length because each action may have several possible outcomes and the total number of outcomes of a plan is the product of the number of outcomes of each of its actions. This complexity can be reduced by using a form of abstraction in which some of the outcomes of an action are bundled together, thus reducing the branching factor [Hanks, 1990a]. Incorporating this kind of abstraction into the DRIPS framework raises several interesting questions. The planner must be able to represent both abstraction of outcomes and abstraction into action classes at the same time. The planner must decide when to refine the bundled outcomes of an action versus when to refine abstract action classes.

The DRIPS planner finds the optimal plan by searching through a static abstraction/decomposition network. The need to completely specify the network places a large burden on the user. Generation of task decompositions is the problem addressed by traditional nonlinear planners. A nonlinear planner could be used to generate the decompositions.

References

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

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

[Haddawy and Hanks, 1993] P. Haddawy and S. Hanks. Utility models for goal-directed decision-theoretic planners. Technical Report 93-06-04, Department of Computer Science and Engineering, University of Washington, June 1993. (Also submitted to *Artificial Intelligence*).

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

[Hanks, 1990b] S. Hanks. *Projecting Plans for Uncertain Worlds.* PhD thesis, Yale University, January 1990.

[Tenenberg, 1991] J.D. Tenenberg. *Reasoning About Plans*, pages 213–283. Morgan Kaufmann, San Mateo, CA, 1991.