A Modular Structured Approach to Conditional Decision-Theoretic Planning

Liem Ngo   Peter Haddawy   Hien Nguyen
Decision Systems and Artificial Intelligence Lab
Dept. of EE&CS, University of Wisconsin-Milwaukee
{*liem,haddawy,hien*}*@lombok.cs.uwm.edu*

Abstract

A realistic system for planning with uncertain information in partially observable domains must be able to reason about sensing actions and to condition its further actions on the sensed information. Among implemented planning systems, we can distinguish two approaches to contingent decision-theoretic planning. The first is characterized by a highly unconstrained plan space, while the second is characterized by a constrained and inflexible specification of plan space. In this paper, we take a middle ground between these two approaches that we consider to be more practical. We permit the user to specify the structure of the space of possible plans to be considered but to do so in a flexible manner. This flexibility is obtained through the use of a modular representation. We separate the representation of actions from the representation of domain relations and we separate those from the representation of the plan space. Actions and domain relations are represented with schematic Bayes net fragments and plan space is represented using programming language constructs. We present a planning system that can find optimal plans given this representation.

Introduction

A realistic system for planning with uncertain information in partially observable domains must be able to reason about sensing actions and to condition its further actions on the sensed information. This is necessary in order to reduce uncertainty about the world and increase the likelihood of plan success. Among implemented planning systems, we can distinguish two approaches to contingent decision-theoretic planning. The first is characterized by the C-Buridan planner (Draper, Hanks, & Weld 1994). C-Buridan is a probabilistic nonlinear planner that first generates a noncontingent plan and then considers places where sensing actions and contingencies may be added in order to increase the probability of success. By doing this,

the planner considers and must search through the infinite space of all possible contingent plans and so the approach tends to be rather inefficient. The second approach is characterized by the DRIPS planner (Haddawy, Doan, & Goodwin 1995). DRIPS considers only a finite space of possible plans structured into an abstraction hierarchy. Sensing actions and contingent actions are prespecified. Actions in DRIPS have conditional effects and this feature is used to implement contingent actions. A contingent action is created manually by the user by folding the various contingent actions into a single action description and using the action conditions to choose the effects that represent the contingent choice. A plan is simply a sequence of contingent actions. While DRIPS is highly efficient when provided with a good abstraction hierarchy, its representation of contingent plans has two disadvantages. First, once the contingent action descriptions are created, modifying the contingencies is difficult. Second, representing a sequence of actions that is contingent on some outcome is extremely cumbersome.

In this paper, we take a middle ground between these two approaches that we consider to be more practical. Our approach is inspired by practice in medical decision making. To find optimal treatment policies, researchers in medical decision analysis (Erkel *et al.* 1996) typically utilize the following procedure: determine the probabilistic models of actions and domain relationships, specify the constraints on the possible plan space, construct the decision trees satisfying the constraints, manually compute the branching probabilities of the decision trees, and solve the decision trees to find the optimal policies. We find appealing the idea that the user of a planning system be permitted to provide whatever domain knowledge he has in the form of constraints on the plan space. So we would like to support this general methodology while relieving the user of the burden of constructing decision trees and computing branch probabilities. Our approach permits the user to specify the structure of the space of possible plans to be considered but to do so in a flexible manner. This flexibility is obtained through the use of a modular representation. We separate the repre-

sentation of actions from the representation of domain relations and we separate these from the representation of the plan space. Actions and domain relations are described with sets of rules in a knowledge base representing schematic Bayesian network fragments. The plan space is represented by using programming language constructs. Contingent execution of actions is represented in the plan space specification, so it is separated from the action descriptions. The language for specifying the plan space allows us not just to represent contingent actions but to represent contingent plan sub-spaces. For example, after performing a sensing action, the set of reasonable further courses of action to consider may be dependent on the outcome of that sensing action. Our representation can capture such structure. Specifically, we represent plan space using *plan schemes*, which are programs in which steps are non-deterministic choices among conditional plans and sequencing is controlled by conditionals

We present an implemented planning system that effectively searches through the space defined by the above representation to find the optimal plan. Our system gains efficiency by using a combination of techniques for creating compact Bayesian network structures to evaluate plans. In a previous paper (Ngo, Haddawy, & Helwig 1995), we presented a framework for representing actions and domain knowledge with a knowledge base of context-sensitive temporal probability logic sentences. Plans were evaluated by constructing and evaluating a Bayesian network tailored to represent each plan. The plan evaluation algorithm presented in this paper builds upon and extends that framework in two ways. First, we present procedures for *knowledge-based construction of compact* Bayesian networks for evaluation of contingent plans.

Second, we present a method of effectively using discrete Bayesian networks to evaluate actions and domain relations involving functional effects and variables with potentially infinite state spaces. Our system automatically abstracts the state space of such variables so that only states of non-zero probability are reasoned about. The algorithm does this by symbolically propagating state information.

Planning Example

Consider the following planning problem which will be used as a running example throughout this paper. It is winter time and you wish to drive across the mountains. The main problem you need to deal with is snow on the mountain pass. You have various actions you can perform, which all take time and during that time snow may be falling. You can find out whether it is currently snowing or not by calling the weather report. The report is very accurate and the call takes only 10 minutes and costs nothing. The mountain passes are controlled by the highway patrol and if the snow is beyond a certain depth, they will not allow cars to proceed without chains on the tires. If the snow is

too deep they will simply stop all traffic. You do not currently own chains but you can go to the store and buy them before leaving town. That takes one hour and costs $40.00. You have two routes you can take through the mountains: (RoadA, Pass1, RoadB) or (RoadC, Pass2, RoadD). RoadA is slower than RoadC but Pass2 has a more restrictive controls than Pass1. On Pass1, if the snow is less than 2 inches all cars can proceed. If it is between 2 and 6 inches only cars with chains can proceed. If it is greater than 6 inches, the pass is closed. On Pass2, the cutoff depths are 1 and 5 inches, respectively. While you are performing your various actions, snow may be falling. If too much snow falls, the passes may be closed. The question we would like to answer is: What is your optimal plan if you wish to reach your destination as soon as possible?

Notice the issues that must be addressed in order to solve this problem:

- We must be able to represent the process of the snow falling, which is independent of the agent's actions.

- The snow level is a function of the rate of snowfall and the amount of time that has passed, so we must be able to represent functional effects on random variables.

- We must be able to represent sensing actions like obtaining the weather report and actions or plans that are performed contingent on the sensed information.

- We must be able to represent the space of possible plans to be considered.

Modular Representation of Action and Domain Models

To represent a planning problem, we must represent the state of the world and how it evolves with time, as well as the actions available to the planning agent. We describe the state of the world with a set of random variables, which we represent as predicates. We require that each predicate have at least one attribute representing the *value* of the corresponding RV. By convention we take this to be the last attribute. For example, the RV *snow-level* can be represented by a two-position predicate $snowlevel(T, V)$, where T is the time point, and V is the real numbered value representing the snow level at T.

We describe action effects and domain relations with probabilistic sentences[2]. A **probabilistic sentence** (p-sentence) in our language has the form $Pr(A_0|A_1, \ldots, A_n) = \alpha \leftarrow B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_k$, where the A_i, B_j and C_r are atoms and α is a number in the $[0, 1]$ interval. The meaning of such a sentence is "in the context that B_j are true, and none of C_k is shown to be true, $Pr(A_0|A_1, \ldots, A_n) = \alpha$". The context serves to select the appropriate probabilistic relation between the RVs. In this paper, action

---

[2]For a detailed formal description of the representation language see (Ngo, Haddawy, & Helwig 1995).

$Pr(time(T, X + 120)|time(T - 1, X)) = 1$
$\leftarrow DriveOnRoadA(T - 1)$

$Pr(hasChains(T, yes)) = 1 \leftarrow BuyChains(T - 1)$
$Pr(time(T, X + 60)|time(T - 1, X)) = 1 \leftarrow BuyChains(T - 1)$
$Pr(cost(T, X + 40)|cost(T - 1, X)) = 1 \leftarrow BuyChains(T - 1)$

$Pr(weatherReport(T, X)|snowfall(T - 1, X)) = 1$
$\leftarrow GetWeatherReport(T - 1)$
$Pr(time(T, X + 10)|time(T - 1, X)) = 1$
$\leftarrow GetWeatherReport(T - 1)$

$Pr(snowlevel(T, X1)|snowfall(T - 1, none),$
$snowlevel(T - 1, X1), time(T, X2), time(T - 1, X3)) = 1$
$Pr(snowlevel(T, X1 + 0.01 * (X2 - X3))|$
$snowfall(T - 1, moderate), snowlevel(T - 1, X1),$
$time(T, X2), time(T - 1, X3)) = 1$
$Pr(snowlevel(T, X1 + 0.05 * (X2 - X3))|$
$snowfall(T - 1, heavy), snowlevel(T - 1, X1),$
$time(T, X2), time(T - 1, X3)) = 1$

$Pr(time(X + 100, T)|time(X, T - 1)) \leftarrow Failure(T - 1)$
$Pr(hasChains(T, X)|hasChains(T - 1, X)) = 1$
$\leftarrow \neg BuyChains(T - 1)$

$Pr(snowfall(0, none)) = .25; Pr(snowfall(0, moderate)) = .35;$
$Pr(snowfall(0, heavy)) = .4 Pr(hasChains(0, no)) = 1$

Figure 1: A portion of the knowledge base modelling the driving domain.

descriptions are always represented by p-sentences in which the context is the predicate representing the action. Domain relations are represented by p-sentences with no context. So given a plan represented by a set of actions, the model construction algorithm uses the context information to select the relations among RVs that hold within the context of that plan.

We represent time by using discrete time points, as well as a RV *time* to indicate the metric time at each time point. For example, the following rule says that if you choose to drive on road A then it will certainly take 120 time units.

$Pr(time(T, X + 120)|time(T - 1, X)) = 1 \leftarrow$
$DriveOnRoadA(T - 1).$

Not all predicates need include a time point as a parameter. Predicates without a time point refer to facts that are independent of time.

In addition to the action descriptions, we need to model two kinds of domain knowledge: intrinsic causal or correlate relationships and persistence rules. In our example, snow-fall is a process independent of our actions. As a result, snow-level depends solely on the snow-fall and the passing time. The basic assumption of persistence is that if the state of a variable is not known to be affected by actions or other events over a period of time, it will tend to remain unchanged over that period. In our example, the predicate *hasChains* persists unless the action *BuyChains* occurs.

Figure 1 shows some of the p-sentences for modelling our example planning problem. For example, the second set of sentences describes the effect of buying chains: the duration is 60 minutes, the cost is
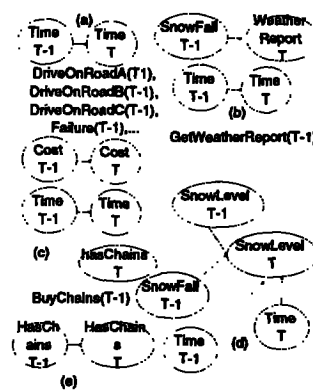

Figure 2: The BN model of (a) DriveOnRoadA,...; (b) GetWeatherReport; (c) BuyChains actions. (d) The domain model of snow level. (e) The persistence rule for hasChains.

$40, and the agent certainly has chains. The fourth set of sentences describes how the snow level is functionally related to the time and the rate of snowfall. The last set of sentences describes the initial state of the world. The sentences describing the actions and domain relations are diagrammatically depicted as Bayesian network fragments in Figure 2. The actions DriveRoadB, DriveRoadC, DriveRoadD, PutOnChains, TakeOffChains, Pass1WithoutChains, Pass2WithoutChains,Pass1WithChains, Pass2WithChains are modelled in the same way as DriveOnRoadA. If the snow level at one pass exceeds a certain limit, the through traffic is blocked. We consider such a situation a failure of the plan and interpret it as "it will take a long time to reach the destination". To simplify the example, we model failure by the action *Failure* which has 100 unit duration.

The Combining Rules

When two or more actions or events influence the state of a RV, we need to know the probability distribution of that RV given all possible combinations of states of the actions and events. For example, if B and C influence A we need to know $P(A|B,C)$. In such situations we need some way of inferring the combined influence from the individual influences. Combining rules such as generalized noisy-OR and noisy-AND are commonly used to construct such combined influences in Bayesian networks. A combining rule takes as input *a set of p-sentences* which have the same RV in their consequents and produces the combined effect of the antecedents on the common consequent.

The combining rules are generally domain-dependent. One plausible rule is that actions take precedence over other causes. For example, if the *BuyChains* action is chosen to be performed at time 5, then the status of *hasChains* at time 6 does not depend on its status at the previous time points. In our current implementation, we provide such a dominance

rule for temporal reasoning: If the user specifies that a probabilistic sentence R1 dominates another probabilistic sentence R2 then when the context of R1 is satisfied, R1 is used and R2 is eliminated from consideration in that context.

Contingent Plans

We represent a **contingent plan (CP)** using a programming language in which the primitive statements are the actions and there are only two control structures: sequential and conditional (by using the CASE construct). Each sequential step is labelled. CASE structures can be nested. The conditions of CASE can refer to the values of RVs in previous time slices. We always assume that in a CASE construct the different branching conditions are mutually exclusive.

Example 1 *In the following CP, the sequence of actions (including possible failure) after DriveOnRoadA is contingent on the snow level at the pass and whether the agent is carrying chains.*

```
s1: DriveOnRoadA
CASE after s1 snowlevel < 2:
         s11: Pass1WithoutChains; s12: DriveOnRoadB
    after s1 snowlevel ≥ 2 AND after s1 snowlevel
    ≤ 6 AND before s1 hasChains is yes:
         s13: PutOnChains; s14: Pass1withChains;
         s15: TakeOffChains; s16: DriveOnRoadB
    after s1 snowlevel > 6 OR
    (after s1 snowlevel ≥ 2 AND after s1 snowlevel
    ≤ 6 AND before s1 hasChains is no):
         s17: failure
END CASE
```

In a CP, we call a sequence of consecutive actions in the plan which does not contain the CASE or ENDCASE keywords a (sequential) plan fragment. A CP can be represented by a graph of its maximal plan fragments as shown in Figure 3. In the figure, the names starting with F denote maximal plan fragments. In the graph representation, each horizontal bar represents a maximal plan fragment and is annotated with the corresponding name of the fragment. The lines connecting the horizontal bars represent the diverging (corresponding to the CASE keyword) and converging (corresponding to the ENDCASE keyword) links. The diverging links are annotated with the corresponding conditions in the program.

The Goal of Plan Projection

In this paper, we are interested in evaluating the probability distribution of some RVs, which are called *goal RVs*, at the end of the performance of a CP. Notice that such a plan has several branching possibilities, each with a specific time length and a specific probability of occurrence. For example, one branch of the CP in Figure 3 is $\langle F0, F1, F11, F6 \rangle$. Suppose the CP P has n possible branches $\mathcal{F}_i, i = 1, \ldots, n$, the probability of occurrence of branch $\mathcal{F}_i$ (or probability of the conjunction of conditions on $\mathcal{F}_i$) is $Pr(\mathcal{F}_i)$, the length
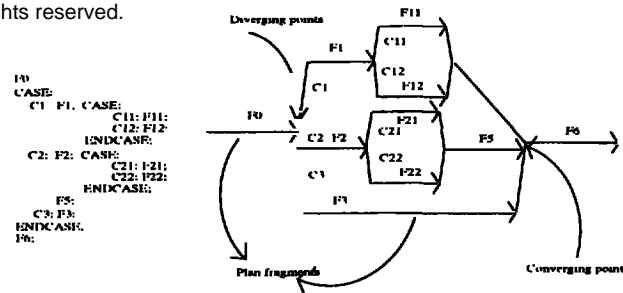

Figure 3: The graph model of a contingent plan.

(duration) of branch $\mathcal{F}_i$ is n_i and we want to evaluate the probability that an (atemporal) RV X achieving the value x. The desired probability is given by the following formula:

$$Pr(X = x|P) = \sum_{i=1}^{n} (Pr(A_i|\mathcal{F}_i) \times Pr(\mathcal{F}_i))$$

where A_i is the ground atom in our language representing the fact that the RV X achieves the value x at time n_i and $Pr(A_i|\mathcal{F}_i)$ is the probability of A_i when $\mathcal{F}_i$ is actually performed.

For example, if the CP in the example 1 is performed at time point 0 then the probability distribution of its duration is determined by:

$Pr(time(3.X)|\mathcal{F}_1) + Pr(time(4.X)|\mathcal{F}_2) \times$
$Pr(\mathcal{F}_2) + Pr(time(2.X)|\mathcal{F}_3) \times Pr(\mathcal{F}_3)$
where $\mathcal{F}_1$ denotes
$(DriveonRoadA(0), Pass1WithoutChains(1).$
$DriveonRoadB(2). snowlevel(1.Y).Y < 2)$. $\mathcal{F}_2$ denotes
$(DriveonRoadA(0).Putonchains(1). Pass1withchains(2).$
$Takeoffchains(3), DriveonRoadB(4), snowlevel(1.Y).$
$Y \geq 2, Y \leq 6. hasChains(0, yes))$ and $\mathcal{F}_3$ denotes
$(DriveonRoadA(0). Failure(1), snowlevel(1,Y).$
$(Y > 6$ OR $(2 \leq Y \leq 6, hasChains(0. no))))$

We compute such probability distributions by constructing and evaluating BN models.

Plan Fragments and Their BN Models

A (sequential) plan fragment is a sequence of actions. If F is the plan fragment $\langle A^{(1)}, \ldots, A^{(n)} \rangle$ and t is a fixed time point, we use F_t to denote the "concrete" plan $\langle A_t^{(1)}, \ldots, A_{t+n-1}^{(n)} \rangle$, where $A_r^{(i)}$ means "the action $A^{(1)}$ is performed at time point r". We say t is the *starting time* and $t + n$ is the *ending time* of F_t.

In order to build a BN model to evaluate a CP, we construct the BN model for each maximal plan fragment (a maximal sequence of actions with CASE construct). The BN model of the CP will be a graph of those component BNs. BN-graphs will be introduced in the next section. Assume we want to construct a BN to evaluate the effect of a plan fragment F on a set of RVs. We assume that F is performed at a time point T, where T is a variable, and try to construct a parameterized BN for F which will be instantiated to a concrete BN at concrete time points when we combine
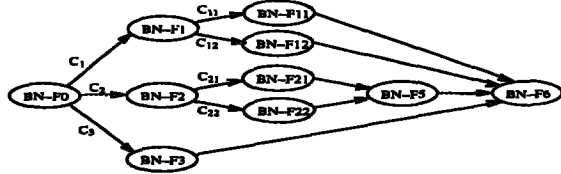
Figure 4: A BN-graph model of a contingent plan.

the fragments. We formulate the problem as evaluating the effect of the "concrete" plan F_T on the goal RVs at time $T + n$.

We adapt the procedure BUILD-NET which was presented in (Ngo, Haddawy, & Helwig 1995) to the current framework to construct *parameterized and conditional BNs*. The concept of conditional Bayesian networks has been used by several researchers, including (Davidson & Fehling 1994). We call a DAG a conditional BN if we can construct from it a BN by assigning prior probability distributions to some of its nodes which do not have incoming arcs. We call such nodes *input RVs*. In our planning settings the input RVs are RVs at the starting time point of plan fragments. For example, the BN-fragments other than BN-$F0$ in Figure 4 are conditional: the prior probability distribution of their starting state RVs is provided by the sequence of preceeding BN-fragments. To emphasize that normal BNs have no missing link matrices, we call them *unconditional BNs*.

We denote the BN constructed by BUILD-NET when the input consists of the plan fragment F, the starting time T, ending time $T + n$ and the set of goal RVs G by $BN(F, T, T + n, G)$. As an example, Figure 5.(a) shows the conditional BN $BN(\langle Pass1WithoutChains(T), DriveOnRoadB(T + 1)\rangle, T, T+2, \{time(T+2)\})$ constructed by our procedure to evaluate the plan fragment $\langle Pass1WithoutChains(T), DriveOnRoadB(T + 1)\rangle$ with respect to the goal RV *time*. Figure 5.(b) shows the conditional BN to evaluate the plan fragment $\langle DriveOnRoadA(T)\rangle$ with respect to the goal *snowLevel* and *time*. Compared to the action model of *DriveOnRoadA* in Figure 2, the additional nodes and links are created by using domain information on *snowLevel*.

Bayesian Network-Graphs

A CP can be represented as a graph in which each link is annotated with an optional condition and each node contains a plan fragment F (see Figure 3). In the previous section we presented the concept of $BN(F, T, T + n, G)$, the conditional BN for evaluating a plan fragment F. In order to evaluate the whole CP, we connect those BNs into a BN-graph.

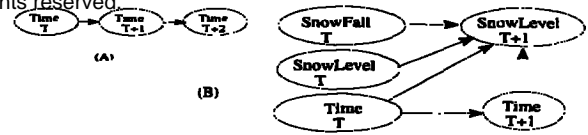**Example 2** *If the CP is represented as a graph of maximal plan fragments in Figure 3 and we have a*

Figure 5: (a)
$BN(\langle Pass1WithoutChains(T), DriveOnRoadB(T + 1)\rangle, T, T + 2, \{time(T + 2)\})$;
(b) $BN(\langle DriveOnRoadA(T)\rangle, T, T + 1, \{time(T + 1), snowLevel(T + 1)\})$.

(conditional or unconditional) BN for each maximal plan fragment then we can connect those BNs into a graph form shown in Figure 4. In Figure 4, each BN-$F_i, i > 0$, is the conditional BN of F_i. BN-F_0 is the unconditional BN of F_0.

In general terms, a BN-graph is an acyclic graph of (conditional or unconditional) BN-fragments which satisfies (1) there is a root node from which every node can be reached by a directed path; (2) if there are more than one arc departing from a node and one of the arcs is annotated with a condition then each such arc must be annotated with a condition; (3) the conditions annotated to arcs departing from a common node are mutually exclusive and covering; (4) the conjunction of the conditions along any directed path must be consistent; and (5) when we connect the BN-fragments on a maximal directed path the result is an unconditional BN without repeating nodes.

Let X be the set of all RVs in a BN-graph G and x be one value assignment of X. Then, there is one and only one maximal directed path b in G such that the conjunction of the conditions on b is consistent with x. We define the probability of x induced by the G as $Pr_G(x) = Pr_b(x)$, where Pr_b is the probability function induced by the BN formed from b.

Figure 5 shows two conditional BNs constructed by our procedure for two plan fragments. The purpose is constructing the BNs relevant to the evaluation of the final goal RV *time* after performing plan fragments $\langle DriveOnRoadA\rangle$ and $\langle Pass1WithoutChains, DriveOnRoadB\rangle$ of the first branch of the CP in example 1. The process starts with the final goal RV and the action *DriveOnRoadB*. The input RVs of $BN(\langle Pass1WithoutChains(T), DriveOnRoadB(T+1)\rangle, T, T+2, \{time(T+2)\})$; and the observable RV $snowLevel(T)$ become the goal RVs of $BN(\langle DriveOnRoadA(T)\rangle, T, T+1, \{time(T+1), snowLevel(T + 1)\})$. *snowLevel* is considered because it appears in the conditions of the given CP.

Notice that we do not generate the entire BN-graph but only the portion relevant to the goal RVs. For example, the snow level is not considered while driving through the pass and road B. This strategy makes the procedure more efficient.

```
Step 2: CASE
     after s1 report is no-snow:
Step 2.1: CHOICE s11: DriveonRoadA:
          s12: Pass1WithoutChains; s13: DriveonRoadB
     OR s14: DriveonRoadC:
          s15: Pass2WithoutChains: s16: DriveonRoadD
     after s1 report is moderate or heavy snow:
Step 2.2: CHOICE s21: BuyChains OR s22: noact
Step 2.3: CHOICE s31: DriveonRoadA
     CASE after s31 snow-level $j$ 2:
          s32: Pass1WithoutChains;s33: DriveonRoadB
          (after s31 snow-level is between 2 and 6)
          and (after s31 hasChains is yes):
          s34: PutonChains; s35: Pass1WithChains;
          s36: TakeoffChains; s37: DriveonRoadB
          (after s31 snow-level $j$8 6) or
          (after s31 snow-level is between 2 and 6)
          and (after s31 hasChains is no):
          s38: failure
     END CASE
     OR s41: DriveonRoadC
     CASE after s41 snow-level $j$ 1:
          s42: Pass2WithoutChains;s43: DriveonRoadD
          (after s41 snow-level is between 1 and 5)
          and (after s41 hasChains is yes):
          s44: PutonChains; s45: Pass2WithChains;
          s46: TakeoffChains; s47: DriveonRoadD
          (after s41 snow-level $j$8 5) or
          (after s41 snow-level is between 1 and 5)
          and (after s41 hasChains is no):
          s48: failure
     END CASE
END CASE
```

Figure 6: The example plan scheme.

A Decision-Theoretic Planner

Decision-Theoretic planners search for an optimal or near-optimal plan in some specified space of possible plans. Many planning systems assume that the space is simply the infinite space defined by all possible sequences of actions from some set, e.g. (Draper, Hanks, & Weld 1994). But for many practical planning problems, constraints can be placed on the space of possible plans. In our example, no reasonable plan should recommend the action GetWeatherReport or BuyChains after going through the pass. We specify the constraints on the plan space by using programming language control constructs which are used to specify CPs.

We represent plan space by using *plan schemes*. To represent a non-deterministic choice we use the construct: *CHOICE plan-scheme1 OR plan-scheme2 OR ....* A plan scheme is an extended form of a CP with CHOICE constructs. A *plan step* in a plan scheme is either a CHOICE construct, a CASE construct, or a maximal sequence of actions without embedded CHOICE or CASE construct.

The plan scheme in Figure 6 has five plan steps. GetWeatherReport is always performed first. In the next step BuyChains may be chosen if the weather report is there is snow. Depending on the result of the weather report, different sequences of actions may be selected. If the report is no-snow, we can choose between Pass 1 and Pass 2. If the report is snow, we can still choose between Pass 1 and Pass 2 but the plans must have contingencies.

Given the plan scheme in Figure 6, we want to find the CP, in the plan space specified by the plan scheme, that minimizes the expected value of execution time. We can represent an arbitrary utility function by a special node with functional effect whose parents are input variables of the function. In our example, the utility function is represented by the *time* variable.

Generalized BN-graphs

One BN-graph encodes the probabilistic structure of one CP. We use a generalized form of BN-graph to store all probabilistic information necessary for the evaluation of all CPs implied by a plan scheme.

A **generalized BN-graph** is a structure similar to BN-graph. The only difference is that generalized BN-graphs contain also CHOICE nodes which specify different alternatives.

The procedure to generate a generalized BN-graph is a simple extension of the BN-graph generation procedure. It is a backward chaining procedure which starts with the input utility variable and the last plan steps in the input plan scheme and generates the corresponding BN-fragments of relevant variables. The input variables of these BN-fragments are used to construct the BN-fragments for the preceeding plan steps.

Figure 7 shows the generalized BN-graph of the plan scheme in Figure 6. In the figure, *BN-BuyChains*, for example, is the conditional BN for the alternative *BuyChains*.

Evaluation Procedures

Given a KB, a plan scheme and a utility function, the system needs to produce a plan(s) in the plan space specified by the plan scheme that maximizes expected utility. We use a two-step process which is commonly used in Influence Diagram evaluation. In the first step, a generalized BN-graph and a decision tree are built. Because a generalized BN-graph of a given plan scheme contains also its branching structure, we just simply unfold the constructed generalized BN-graph to get the corresponding decision tree. The main part of the next step involves the computation of probability values, which is performed on the constructed generalized BN-graph. After that we use a simple algorithm named *sum-and-fold-back* which is similar to *average-out-and-fold-back* method (Raiffa 1968) to find the optimal plan.

Implementation Issues

Our system is written in Common LISP. Probabilistic sentences which share a common consequence and context are bundled in a rule. For example, the three p-sentences specifying the prior probability of snow-fall in the example 1 is represented by the following rule:

```
((Snow-fall (?t)) () (.25 .35 .4)
 (none moderate heavy) R6)
```
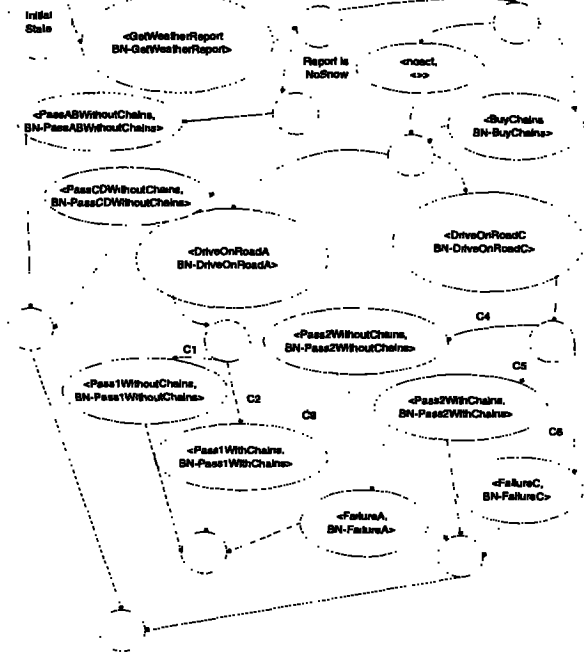
Figure 7: The generalized BN-graph for the example plan scheme.

Figure 8: The decision tree for the example plan scheme.

In the current implementation, we provide three combining rules: dominance, Noisy-OR and Noisy-AND rules. For each predicate, a combining strategy needs be specified. A combining strategy is represented as an expression with dominance, Noisy-OR and Noisy-AND as operators and rule labels as operands. In a given context and with a given RV, our system determines the applicable rules and uses the given combining strategy to form the corresponding link matrix.

Our system allows variables with very large or infinite state space. For example, the state space of snow level is potentially infinite. We use the observation that in concrete situations these variables may achieve only values from finite sets to construct BNs with discrete state variables. For example, snow level is computed from the starting snow level, snow fall and time lapse (see Figure 1). If the set of possible values of snow level at the starting state is finite then the set of possible values of snow level after the performance of a finite sequence of actions is also finite.

For the running example, when the failure is penalized with 100 time units, our system returns the optimal plan without the *BuyChains* action. If we raise the penalty to 400 time units, the returned optimal plan contains *BuyChains*. In both cases, pass 2 is chosen because road C is fast enough to avoid snow build-up. The optimal plans were returned by the planner in approximately 1 minute on a Pentium 150 Mhz.

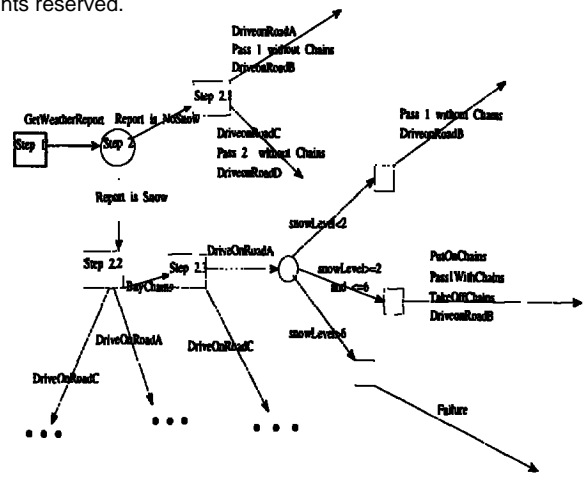We have applied our planner in a real problem, the diagnosis of Suspected Pulmonary Embolism ((Erkel *et*

al. 1996)). After defining the knowledge base, different plan schemes incorporating expert's knowledge were given to the planner, which returned the corresponding optimal plans in less than 1 minute.

Discussion and Related Work

Two-Stage Bayesian Networks

One currently active approach to Markov Decision Processes (e.g. (Boutilier, Dearden, & Goldszmidt 1995)) represents each action by a fixed two-stage BN that combines action model, domain model, and the execution context of the action. For example, the action $DriveOnRoadA(T)$ would be represented by the BN in Figure 5.(b) in the two-stage BN approach. In general, an action model in the two-stage BN approach needs to include all possibly related domain variables. In our framework, $DriveOnRoadA(T)$ is represented by the two-stage BN in Figure 2.(a). The relationships with $SnowFall$ and $SnowLevel$ are constructed as needed using domain models and combining rules. Hence, we offer a more modular and flexible representation, and produce smaller BNs.

Relationship to Influence Diagrams

Our proposed framework is an extension of the influence diagram (ID) formalism. The asymmetric structure of BN link matrices has been recognized by several authors, e.g.(Boutilier, Dearden, & Goldszmidt 1995). In that work, link matrices are represented in the form of decision trees and an evaluation procedures exploiting asymmetric structure is investigated. BN-graphs offer another alternative for representing and utilizing asymmetric structures. Also, generalized-BN-graphs are an asymmetrized form of IDs.

Markov Decision Processes

One active approach to decision-theoretic planning is based on Markov decision processes (Dean *et al.* 1993). While the traditional Markov decision process approach uses simple state transition and unstructured value function representations, more recent work in this area, e.g (Boutilier, Dearden, & Goldszmidt 1995), exploits the structure of factored state spaces, action spaces, and value funtions to gain computational leverage. In this paper, we have further explored the structure of action models and have gained computational leverage primarily by structuring the plan space. Rather than choosing from the entire set of actions at every stage, certain choices of actions or sub-plans often constrain our later choices of actions or sub-plans. This observation helps to significantly reduce the size of the plan spaces, and hence, improve the efficiency of optimal plan generation.

In (Blythe 1996), Blythe describes a technique to automatically abstract a Markov chain to efficiently answer specific queries.

C-BURIDAN

The main differences between C-BURIDAN and our approach lie in the expressivenss of the representation and the structuring of plan space. Our action descriptions and domain models can include quantified variables. We allow *derived effects* of actions through domain relationships. We allow functional effects. C-BURIDAN searches through an infinite space of plans. Without good heuristic guidance, that can incur high computational cost. There is also no guarantee that the planner will eventually find a satisficing plan when one exists. In contrast, our approach requires a user to provide the structure of plan space. It always returns the optimal plan(s) in the given plan space.

DRIPS

Our approach offers the following advantages over DRIPS:

- The DRIPS representation does not include quantified variables or derived effects.

- DRIPS has not facility to support representation of contingent actions.

- We detach action and domain models from the contingencies and plan structure. This modularity makes the modification of the components of a planning problem simple.

- Our planner constructs BNs for plan evaluation. These BNs can be utilized for analysis and explanation tasks.

Future Research

CPs can be generalized to contain loops and parallel control structures. Another future research topic is the efficient reasoning on BN-graphs. Currently, we use the graph structure to compactly store a set of BNs. Yet the inference is still performed on individual component nets. We are looking for more efficient procedures which work directly on BN-graphs.

We plan to incorporate other forms of constraints on the plan space that can be used in conjunction with our plan schemes. We currently investigate more efficient procedures for finding optimal plan(s). We plan to incorporate DRIPS action abstraction techniques (Haddawy, Doan, & Goodwin 1995) to guide the search.

References

Blythe, J. 1996. Event-based decompositions for reasoning about external change in planners. In *Proceedings of the Third International Conference on AI in Planning Systems*.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1995. Exploiting structure in policy construction. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Davidson, R., and Fehling, M. 1994. A structured, probabilistic representation of action. In *Proceedings of The Tenth Conference on Uncertainty in AI*, 154 161.

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

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, 31 36.

Erkel, A.; van Rossum, A. B.; Bleom, J. L.; Kievit, J.; and Pattynama, P. M. T. 1996. Spiral ct angiography for suspected pulmonary embolism: A cost-effective analysis. *Radiology* 201:29 36.

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

Ngo, L.; Haddawy, P.; and Helwig, J. 1995. A theoretical framework for temporal probability model construction with application to plan projection. In Besnard, P., and Hanks, S., eds., *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 419 426.

Raiffa, H. 1968. *Decision Analysis*. Reading,MA: Addison-Wesley.