Handling Contingency Selection Using Goal Values

Nilufer Onder
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
nilufer@cs.pitt.edu

Martha E. Pollack
Department of Computer Science
and Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
pollack@cs.pitt.edu

Abstract

A key question in conditional planning is: how many, and which of the possible execution failures should be planned for? One cannot, in general, plan for all the failures that can be anticipated: there are simply too many. But neither can one ignore all the possible failures, or one will fail to produce sufficiently flexible plans. We describe a planning system that attempts to identify the contingencies that contribute the most to a plan's overall value. Plan generation proceeds by extending the plan to include actions that will be taken in case the identified contingencies fail, iterating until either a given expected value threshold is reached or planning time is exhausted.

Introduction

Classical AI plan generation systems assume static environments and omniscient agents, and thus ignore the possibility that events may occur in unexpected ways—that contingencies might arise—during plan execution. A problem with classical planners is, of course, that things do not always go "according to plan." In contrast, universal planning systems and more recent MDP-based systems make no such assumption. They produce "plans" or "policies" that are functions from states to actions. However, the state space can be enormous, making it difficult or impossible to generate complete policies or truly universal plans[1].

Conditional planners take the middle road. They allow for conditional actions with multiple possible outcomes and for sensing actions that allow agents to determine the current state (Blythe & Veloso 1997; Draper, Hanks, & Weld 1994; Etzioni *et al.* 1992; Goldman & Boddy 1994; Peot & Smith 1992; Pryor & Collins 1993). A key question in conditional planning is: how many, and which of the possible execution failures should be planned for?

[1]Thus Dean *et al.* (1995) describe an algorithm to construct an initial policy for a restricted set of states, and incrementally increase the set of states covered.

In this paper, we describe Mahinur, a probabilistic partial-order planner that supports conditional planning with contingency selection based on probability of failure and goal values. We present an iterative refinement planning algorithm that attempts to identify the influence each contingency would have on the outcome of plan execution, and then gives priority to the contingencies whose failure would have the greatest negative impact. The algorithm first finds a skeletal plan to achieve the goals, and then during each iteration selects a contingency whose failure will have a maximal *disutility*. It then extends the plan to include actions to take in case the selected contingency fails. Iterations proceed until the expected value of the plan exceeds some specified threshold or planning time is exhausted.

Planning with Contingency Selection

We start with a basic idea: a plan is composed of many steps that produce effects to support the goals and subgoals, but not all of the effects contribute equally to the overall success of the plan. Of course, if plan success is binary—plans either succeed or fail—then in some sense all the effects contribute equally, since the failure to achieve any one results in the failure of the whole plan. However, as has been noted in the literature on decision-theoretic planning, plan success is not binary. In this paper, we focus on the fact that the goals that a plan is intended to achieve may be decomposable into subgoals, each of which has some associated value.

For example, consider a car in a service station and a plan involving two goals: installing the fallen front reflector and repairing the brakes. The value of achieving the former goal may be significantly less than the value of achieving the latter. Consequently, effects that support only the former goal (e.g., getting the reflector from the supplier) contribute less to the overall success of the plan than the effects that support only the latter (e.g., getting the calipers). Effects that support both

1. **Skeletal plan.**: Construct a skeletal plan.
2. **Plan refinement.**: While the plan is below the given expected value threshold:
   2a. Select the contingency with the highest expected disutility.
   2b. Extend the plan to include actions that will be taken in case the contingency fails.

Figure 1: The planning algorithm.

goals (e.g., knowing the phone number of the supplier) will have the greatest importance. We will say that an effect e *supports* a goal g if there is some path through the plan starting at e and ending at g.

The high level specification of a planning algorithm based on this idea is shown in Fig. 1. It first builds a *skeletal plan* which is one in which every subgoal is supported minimally, i.e., by exactly one causal or observation link. In our current implementation of the Mahinur system, the skeletal plan is constructed using Buridan with the restriction that each subgoal is supported by exactly one link.

This algorithm depends crucially on the notions of contingencies and their expected values. We define *contingencies* to be subsets of outcomes of probabilistic actions. For instance, if we repair the brakes, we may or may not succeed in having the brakes functioning properly (perhaps the new calipers are defective). Having the brakes functioning correctly is one contingent outcome of the repair action; not having them is another. A deterministic action is simply one with a single (contingent) outcome.

The importance of planning for the failure of any particular contingency can then be seen to be a factor of two things: the probability that the contingency will fail, and the degree to which the failure will effect the overall success of the plan. To compute the former, one needs to know both the probability that the action generating the particular contingency will be executed, and the conditional probability of the contingency given the action's occurrence. Computing the latter is more difficult. In this paper, however, we make two strong simplifying assumptions. First, we assume that the top-level goals for any planning problem can be decomposed into utility-independent[2] subgoals having fixed scalar values. Second, we assume that all failures are equally difficult to recover from at execution time.

With these two assumptions, we can compute the

[2]The utility of one subgoal does not depend on the degree to which other goals are satisfied (Haddawy & Hanks 1993).

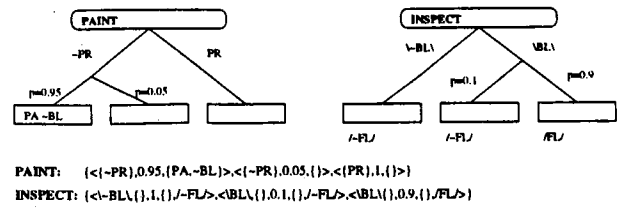

Figure 2: Two example actions.

expected disutility of a contingency's failure: it is the probability of its failure, multiplied by the sum of the values of all the top-level goals it supports. Planning for the failure of a contingency c means constructing a plan that does not depend on the effects produced by c, i.e., a plan that will succeed even if c fails to occur.

Action Representation

To formalize these notions, we build on the representation for probabilistic actions that was developed for (Kushmerick, Hanks, & Weld 1995, p. 247). *Causal actions* are those that alter the state of the world. For instance, the PAINT action depicted both graphically and textually in Fig. 2 has the effects of having the part painted (PA) and blemishes removed (~BL) 95% of the time when the part has not been processed (~PR).

Definition 1 A causal action N is defined by a set of *causal consequences*:

$$\mathtt{N}: \quad \{< t_1, \rho_{1,1}, e_{1,1} >, \ldots, < t_1, \rho_{1,l}, e_{1,l} >, \ldots, \\ < t_n, \rho_{n,1}, e_{n,1} >, \ldots, < t_n, \rho_{n,m}, e_{n,m} >\}.$$

For each i, j: t_i is an expression called the consequence's *trigger*, and $e_{i,j}$ is a set of literals called the *effects*. $\rho_{i,j}$ denotes the probability that the effects in $e_{i,j}$ will be produced given that the literals in t_i hold. The triggers are mutually exclusive and exhaustive. For each effect e of an action, $RESULT(e, st)$ is a new state that has the negated propositions in e deleted from, and the non-negated propositions added to st.

Conditional plans also require *observation actions* so that the executing agent will know which contingencies have occurred, and hence, which actions to perform. Observational consequences are different from causal consequences in that they record two additional pieces of information: the *label* (in //) shows *which proposition's* value is being reported on, the *subject* (in \\) shows *what the sensor looks at*. The subject is needed when an aspect of the world is not directly observable and the sensor instead observes another (correlated) aspect of the world. For instance, a robot INSPECTing a part might look at whether it is blemished (BL), to report whether it is flawed (/FL/) (Fig. 2). Note that

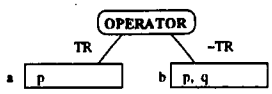

Figure 3: Multiple outcomes.

the label and subject are not arbitrary strings—they are truth-functional propositions.

Definition 2 An **observation action** N is a set of *observational consequences*:

$$\begin{aligned} \mathtt{N}: \quad & \{< sbj_1, t_1, \rho_{1,1}, e_{1,1}, l_{1,1} >, \\ & \ldots, < sbj_1, t_1, \rho_{1,l}, e_{1,l}, l_{1,l} >, \ldots, \\ & < sbj_n, t_n, \rho_{n,1}, e_{n,1}, l_{n,1} >, \\ & \ldots, < sbj_n, t_n, \rho_{n,m}, e_{n,m}, l_{n,m} >\}. \end{aligned}$$

For each i, j: sbj_i is the *subject* of the sensor reading, t_i is the *trigger*, $e_{i,j}$ is the set of any *effects* the action has, and $l_{i,j}$ is the label that shows the *sensor report*. $\rho_{i,j}$ is the probability of obtaining the effects and the sensor report. The subjects and triggers are mutually exclusive and exhaustive. The subject and trigger propositions cannot be identical, although the subject and label propositions can be.

This observation action representation is similar to C-Buridan's (Draper, Hanks, & Weld 1994). However, C-Buridan labels are arbitrary strings that do not relate to propositions, and it does not distinguish the subject of the sensor reading.

Contingencies

An important issue in the generation of probabilistic plans is the identification of equivalent consequences. Suppose that an action A is inserted to support a proposition p. All of the outcomes of A that produce p as an effect can then be treated as an equivalent *contingent outcome*, or *contingency*. The contingencies of a step are relative to the step's intended use. If the operator in Fig. 3 is used to support p, then {a,b} is its single contingency, but if it is used to support q, then {a} and {b} are alternative contingencies.

The Plan Structure

We define a planning problem in the usual way, as a set of initial conditions encoded as a start step (START: $\{< \emptyset, \rho_1, e_1 >, \ldots, < \emptyset, \rho_n, e_n >\}$), a goal (GOAL: $\{< \{g_1, \ldots, g_n\}, 1, \emptyset >\}$), and an action library. We assume that each top-level goal g_i has value $val(g_i)$. A plan in our framework has two sets of steps and links (causal and observational), along with binding and ordering constraints.

Definition 3 A **partially ordered plan** is a 6-tuple, $< T_c, T_o, O, L_c, L_o, S >$, where T_c and T_o are causal and observation steps, O is a set of temporal

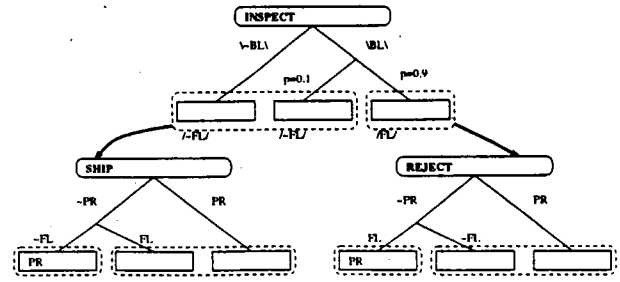

Figure 4: Observation links.

ordering constraints, L_c and L_o are sets of causal and observation links, and S is a set of open subgoals.

Causal links record the causal connections among actions, while observational links specify which steps will be taken depending on the report provided by an observation step. It is important to note that both types of links emanate from contingencies rather than individual outcomes. For instance, in Fig. 4, SHIP will be executed if INSPECT reports /FL/, and REJECT will be executed otherwise.

Definition 4 A **causal link** is a 5-tuple $< S_i, c, p, c_k, S_j >$. Contingency c of step S_i is the link's *producer*, contingency c_k of step S_j is the link's *consumer*, and p is the *supported proposition* ($S_i, S_j \in T_c \cup T_o$).

Definition 5 An **observation link** is a 4-tuple $< S_i, c, l, S_j >$. Contingency c is the collection of consequences of $S_i \in T_o$ that report l. An observation link means that $S_j \in T_c \cup T_o$ will be executed only when the sensor report provided by S_i is l.

Often the planner needs to reason about all the steps that are subsequent to a particular observation. We therefore define a *composite step*, which is a portion of the plan graph rooted at an observation action O. Each composite step is factored into *branches*, one branch for each contingency associated with O. For example, the composite step $(O :< \{BL, 1\}, \{BL, 0.1\} >< S_1, S_2 > , < \{BL, 0.9\} >< S_3 >)$, indicates that after some observation action O, steps S_1 and S_2 will be executed subsequent to one observation (which always occurs when BL is false, and occurs 10% of the time when BL is true), while step S_3 will be executed subsequent to the alternative observation (which occurs 90% of the time when BL is true).

Definition 6 A **composite step** includes an observation action and the entire plan graph below it:

$$\begin{aligned} (O: \quad & < \{t_{1,1}, \rho_{1,1}\}, \ldots, \{t_{1,l}, \rho_{1,l}\} >< S_{1,1}, \ldots, S_{1,m} >, \\ \ldots, \quad & < \{t_{p,1}, \rho_{p,1}\}, \ldots, \{t_{p,q}, \rho_{p,q}\} >< S_{p,1}, \ldots, S_{p,r} >). \end{aligned}$$

$< S_{i,1}, \ldots, S_{i,m} >$, called *branch* i, includes the steps that will be executed if contingency i of the observation

step is realized. $< \{t_{i,1}, \rho_{i,1}\}, \ldots, \{t_{i,l}, \rho_{i,j}\} >$ denote the *condition* for contingency i.

Although composite steps are an important part of our overall framework, for brevity in this paper, we provide only base definitions, not involving composite steps. (See (Onder & Pollack 1996) for the general definitions).

Expected Value

The main idea in computing the expected value of a contingency c is to find out which top-level goal(s) will fail if the step fails to produce c. For example, suppose that c supports top-level goal g_i in two ways: along one path with probability 0.7, and along another with probability 0.8. If c fails, the most that will be lost is the support for g_i with probability 0.8, and thus the expected value of c is $0.8 \times val(g_i)$. Therefore, while propagating the values to a contingency, we take the maximum support it provides for each top-level goal.

Definition 7 Expected value of a contingency: Suppose that contingency c has outgoing causal links to consumer contingencies $c_1, \ldots, c_n$ and possibly an outgoing observation link to step S. Assume that k_i is the probability that c_i supports goal g, and k_{n+1} is the probability that S supports g. Then, the expected value of c with respect to g is:

$$evg(c,g) = \begin{cases} val(g), \text{if } c \text{ directly supports } g, \\ max(k_1, \ldots, k_{n+1}) \times val(g), \text{otherwise.} \end{cases}$$

For z top-level goals, the expected value of c is

$$EV(c) = \sum_{i=1}^{z} evg(c, g_i).$$

We next consider the computation of expected disutility of a contingency's failure. To begin, we need to compute the probability that a given state will occur after the execution of a sequence of actions.

Definition 8 The probability of a state after a (possibly null) sequence of steps is executed:

$$P[st'|st, <>] = \{1, \text{if } st' = st; 0, \text{otherwise.}\},$$

$$P[st'|st, < S >] = \sum_{<t_i, \rho_{i,j}, e_{i,j}> \in S} \rho_{i,j} P[t_i|st] P[st'|RESULT(e_{i,j}, st)],$$

$$P[st'|st, < S_1, \ldots, S_n >] = \sum_u P[u|st, < S_1 >] P[st'|u, < S_2, \ldots, S_n >].$$

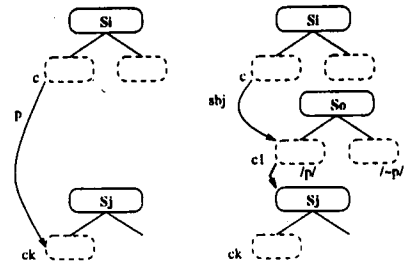

Figure 5: Inserting a new observation action.

where S_1 is a causal step[3]. The action sequence is a total ordering consistent with the partially ordered plan.

The expected disutility of a contingency c is the product of the probability that the action generating c will be executed, the conditional probability of c given the action's occurrence, and the value of c.

Definition 9 Expected disutility of the failure of a contingency: Let S be a step and c be the ith contingency of S. If the condition for c is $< \{t_{i,1}, \rho_{i,1}\}, \ldots, \{t_{i,j}, \rho_{i,j}\} >$, then the expected disutility of c is defined as

$$P[S \text{ is executed}] \times (1 - (\textstyle\sum_{k=1}^{m} \rho_{i,k} P[t_{i,k}| < S_1, \ldots, S_{i-1} >]) \times EV(c).$$

The Planning Algorithm

We can now see how the algorithm in Fig. 1 works. After forming a skeletal plan, it selects a contingency c whose failure has maximal expected disutility. Suppose that c has an outgoing causal link $< S_i, c, p, c_k, S_j >$ (Fig 5, left). Because the proposition supported by c is p, the planner selects an observation action, S_o, that reports the status of p and inserts it ordered to come after S_i. Two contingencies for S_o are formed: c_1 contains all of S_o's outcomes that produce the label $/p/$, and c_2 contains all the other outcomes. The causal link $< S_i, c, p, c_k, S_j >$ is removed and an observation link $< S_o, c_1, /p/, S_j >$ is inserted to denote that S_j will be executed when the sensor report is $/p/$. In addition, a causal link, $< S_i, c, sbj, c_1, S_o >$ is inserted (Fig. 5, right). This link prevents the insertion of an action that would perturb the correlation between the propositions coded in the subject and the label.

Conditional planning then proceeds in the CNLP style: the goal is duplicated and labeled so that it cannot receive support from the steps that depend on contingency c_1 of the observation step.

Plan iteration stops when a given expected value threshold is reached or planning time is exhausted.

[3]The probability of an expression e with respect to a state st is defined as: $P[e|st] = \{1, \text{if } e \subseteq st; 0, \text{otherwise.}\}$.

Definition 10 The **expected value of a totally ordered plan** is the sum of the product of the probability that each of the z goal propositions will be true after S_n is executed, and the value of the goal.

$$\sum_{i=1}^{z} \sum_{u} P[g_i|u] \times P[u| < S_1, \ldots, S_n >] \times val(g_i).$$

where u ranges over all possible states.

We currently use an arbitrary total ordering of the plan for this computation.

Example

The following example is based on one solved by C-Buridan. The goal is to process (PR) and paint (PA) parts. Initially, 30% of the parts are flawed (FL) and blemished (BL), the rest are in good condition. The "correct" way of processing a part is to reject it if it is flawed, and to ship it otherwise. We assign 100 as the value of processing a part, and 560 as the value of painting. The action library contains the SHIP, REJECT, INSPECT and PAINT actions of Draper *et al.* (1994).

The Mahinur planning system starts by constructing a skeletal plan, and nondeterministically chooses the SHIP action to process the part. Two contingencies of SHIP are constructed (shown in Fig. 6 in dashed boxes): the first one produces PR, and the second does not. A causal link that emanates from the first contingency is established for PR.

The triggers for the first contingency are adopted as subgoals (¯PR,¯FL), and both are supported by START. For START, two different sets of contingencies are constructed: one with respect to ¯PR, and one with respect to ¯FL. When the PAINT action is inserted to support the PA goal, and the ¯PR trigger of the first contingency is supported by START, the skeletal plan is complete.

The skeletal plan contains three contingencies whose disutilities of failure must be calculated (the contingency for ¯PR of START has no alternatives). The expected disutilities for the first contingencies of START, PAINT, and SHIP are $0.3 \times 100 = 30$ (for ¯FL), $0.05 \times 560 = 28$ (for PA), and $0.3 \times 100 = 30$ (for PR), respectively.

Mahinur chooses to handle the first contingency with the higher expected disutility, namely the first contingency of START. Because ¯FL is the proposition supported by that contingency, an action that inspects the value of FL is inserted before SHIP (Fig. 7). The causal link supporting ¯FL is removed and an observation link from the ¯FL report is inserted. The INSPECT step looks at BL to report about FL. Therefore, the first contingency of START is linked to INSPECT to ensure that an action that alters the value of BL cannot be inserted between START and INSPECT (BL and FL are correlated).

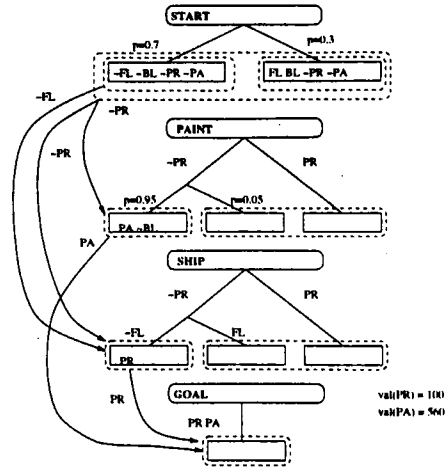

Figure 6: The skeletal plan.

The new branch is then completed. For the duplicated goal, a new REJECT action is inserted to support PR, and the existing PAINT step is used to support PA yielding the plan shown in Fig. 7. If the success threshold has not been met, Mahinur will plan for additional contingencies. Note that existing conditional planners can solve this problem, but the plan generated will be the same regardless of the value assigned to the top-level goals. Within our framework, the planner is sensitive to the values assigned to each goal and is able to focus on different parts of the plan based on the disutility of contingencies.

Mahinur was implemented using Common Lisp on a PC running Linux. We conducted preliminary experiments on two sets of problems. In the first set, we used synthetic problems which yielded plans containing 1, 2, or 3 possible sources of failure. Table 1 shows the total run time for solving the problem and amount of time used for disutility calculation. As expected, runtime increases exponentially with the size of the problem[4]; disutility calculation, however, is insignificant.

For the second set of experiments, we implemented the C-Buridan algorithm. C-Buridan constructs branches for alternative contingencies in a somewhat indirect fashion. When it discovers that there are two incompatible actions, say, $A1$ and $A2$, each of which can achieve some condition C, it introduces an observation action O with two contingent outcomes, and then "splits" the plan into two branches, associating each outcome with one of the actions. We used the same support functions for conditional planning (context propagation, checking context compatibility), the same ranking function and the same base planning sys-

[4]The problems marked with a * terminated with a LISP memory error.

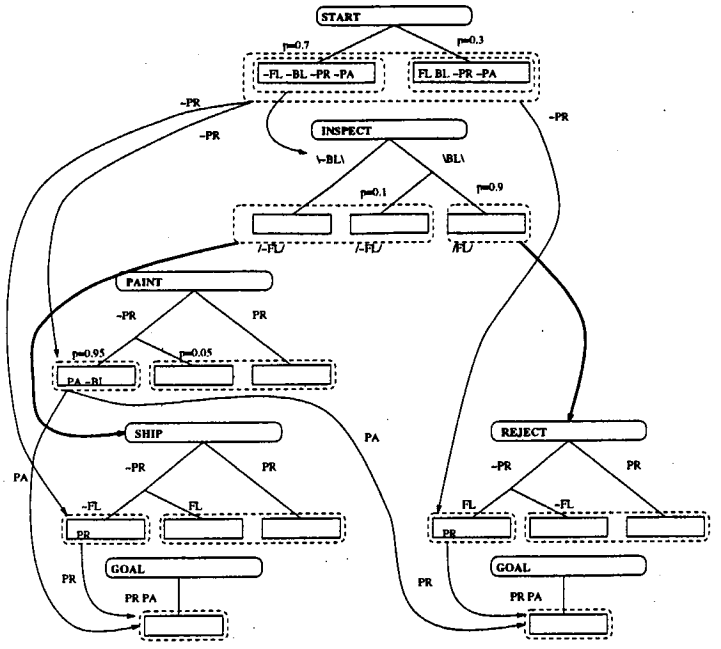

Figure 7: The complete plan.

	1 poss. fail.		2 poss. fail.		3 poss. fail.	
#	Run	Disu.	Run	Disu	Run	Disu.
0	0.01	0.01	0.03	0.01	0.08	0.01
1	0.14	0.01	0.56	0.01	3.92	0.02
2	0.42	0.01	11.28	0.02	335.21	0.03
3	1.67	0.02	229.22	0.03	4368.43	0.05
4	9.11	0.03	3558.87	0.04	*	-

Table 1: Planning time and disutility calculation time for iterations on three problems.

tem. To demonstrate the advantage of directly planning for contingencies, we used a domain which contains a single PAINT operator. Painting a part repeatedly increases the probability of success by 0.5^n, where n is the number of repetitions. It is an error to paint an already painted part, thus the plan has to contain an observe action to check whether the part has been painted and repeat the PAINT step only if it has not been successful previously. Table 2 compares the run times for C-Buridan and Mahinur as the probability threshold is increased. Mahinur expands significantly less number of nodes, because the planning process concentrates on planning for contingencies, as opposed to improving the support for the (sub)goals.

Summary and Related Work

We presented an approach to decision-theoretic plan refinement that directly reasons about the value of planning for the failure of alternative contingencies. To do this, we adopted a different strategy towards con-

Probability	C-Buridan	Mahinur
0.75	22	27
0.875	1317	106
0.9375	*	336
0.96875	*	566

Table 2: Number of nodes visited as the probability threshold is increased.

ditional planning than that taken in some earlier systems, notably C-Buridan which does not reason about whether contingent outcomes are worth planning for.

The PLINTH system (Goldman & Boddy 1994) and the Weaver system (Blythe & Veloso 1997) skip certain contexts during the construction of probabilistic plans. Our approach differs from theirs in two aspects. First, we are focusing on partial-ordering, where they use a total-order approach. Second, we consider the impact of failure in addition to the probability of failure.

Other relevant work includes the recent decision-theoretic planning literature, e.g., the PYRRHUS system (Williamson & Hanks 1994) which prunes the search space by using domain-specific heuristic knowledge, and the DRIPS system (Haddawy & Suwandi 1994) which uses an abstraction hierarchy. Kambhampati (1994) describes planning algorithms with multi-contributor causal links. Our algorithm maintains multiple contributors at the action outcome level in a probabilistic setting.

Acknowledgments

This work has been supported by a scholarship from the Scientific and Technical Research Council of Turkey, by the Air Force Office of Scientific Research (Contract F49620-92-J-0422), by Rome Laboratory and the Defense Advanced Research Projects Agency (Contract F30602-93-C-0038), and by an NSF Young Investigator's Award (IRI-9258392).

References

Blythe, J., and Veloso, M. 1997. Analogical replay for efficient conditional planning. In *Proceedings of the 14th National Conference on Artificial Intelligence.*

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76:35–74.

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, 31–36.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to plan-

ning with incomplete information. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 115–125.

Goldman, R. P., and Boddy, M. S. 1994. Epsilon-safe planning. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, 253–261.

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.

Haddawy, P., and Suwandi, M. 1994. Decision-theoretic refinement planning using inheritance abstraction. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, 266–271.

Kambhampati, S. 1994. Multi-contributor causal structures for planning: a formalization and evaluation. *Artificial Intelligence* 69:235–278.

Kushmerick, N.; Hanks, S.; and Weld, D. S. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76:239–286.

Onder, N., and Pollack, M. E. 1996. Contingency selection in plan generation. In *1996 AAAI Fall Symposium on Plan Execution: Problems and Issues*, 102–108.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, 189–197.

Pryor, L., and Collins, G. 1993. Cassandra: Planning for contingencies. Technical Report 41, The Institute for the Learning Sciences, Northwestern University.

Williamson, M., and Hanks, S. 1994. Optimal planning with a goal-directed utility model. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, 176–181.