From: AAAI Technical Report SS-94-06. Compilation copyright © 1994, AAAI (www.aaai.org). All rights reserved.

Decision-Theoretic Subgoaling for Planning with External Events

Jim Blythe
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jblythe@cs.cmu.edu

Abstract

I describe a planning methodology for domains with uncertainty in the form of external events that are not completely predictable. Under certain conditions, these events can be modelled as continuous-time Markov chains whose states are characterised by the planner's domain predicates. Planning is goal-directed, but the subgoals are suggested by analysing the utility of the partial plan rather than being simply the open conditions of the operators in the plan, a technique I call "decision-theoretic subgoaling". Other planners for uncertain domains can be viewed as performing decision-theoretic subgoaling, which I argue is a useful way to combine AI-based planning and decision theory[1].

1 Introduction

One of the central assumptions of classical planning is that the state resulting at some time after performing an action can be predicted completely and with certainty. This assumption permits a style of planning in which a goal, represented by a sentence in first-order logic, is achieved exactly by a plan, represented as a partially ordered set of actions. The plan need include no sensing or branching because of the assumption.

More realistic planners allow for uncertainty in the results of actions. Specifically, uncertainty in the domain is typically represented in one or more of 3 ways:

1. non-deterministic effects of operators, possibly with probability distributions,
2. uncertainty in the initial conditions of the domain, and
3. uncertainty about future states due to unpredictable external events in the domain.

This classification is not intended to be exhaustive or exclusive. For instance, non-deterministic effects of operators in the Cassandra planner are represented by conditional effects dependent on unknown (and unknowable) initial conditions [Pryor and Collins, 1993].

In a domain that includes uncertainty, plans must account for the different possible outcomes of actions. Thus plans take on the form of a Markov chain, rather than a partially ordered set of actions [Dean *et al.*, 1993, Thiebaux *et al.*, 1993]. Each node of the chain represents a set of states and the action that the plan associates with this set. The arcs, with associated probabilities, represent the different possible outcomes of the action. Plans may also have to include explicit sensing, to determine which alternative was reached.

Plans produced in uncertain domains may not achieve their goals with certainty. Planners dealing with uncertainty have typically considered the probability of plan success as a measure of goodness of the plan [Kushmerick *et al.*, 1993], which is decision-theoretic under the assumption that the utility of an event history is completely and positively determined by the binary property of goal satisfaction [Haddawy and Hanks, 1994]. When there are no time constraints on the planning process, one might continue planning until some threshold probability of success for the plan is reached [Kushmerick *et al.*, 1993], or attempt to cover every possible outcome [Pryor and Collins, 1993]. For cases when the time available for planning might preclude this, "anytime algorithms" that increase the expected utility of the plan over time have been proposed [Dean *et al.*, 1993, Koenig, 1992].

Most planners that deal with uncertainty deal with uncertain action effects or uncertainty in the initial domain, the first two types mentioned above. In this paper I describe planning under uncertainty due to external events which are not completely predictable. I will assume that some knowledge about the distribution of these events is available to the planner in the form of continuous-time Markov processes [Ross, 1980]. In this representation, given a state, we assume an exponential distribution for the amount of time before an external event changes the state, and store the mean time and a probability distribution over the new states that the events can create. Other planning formalisms are able to represent events in more general ways [Dean and Kanazawa, 1989,

[1]This research is sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Material Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. Views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of Wright Laboratory or the United States Government.

Haddawy, 1991], but they do not specify planning mechanisms that make use of the information directly.

I describe a method of plan repair for dealing with this type of uncertainty that uses goal-directed search, where the goals are suggested by decision-theoretic criteria. I argue in section 4 that this is a promising way to combine the approaches of decision theory and AI-based plan synthesis. In the next section I describe the representation for external events in more detail, and in section 3 I illustrate the plan repair technique, which is implemented on top of Prodigy 4.0 [Carbonell *et al.*, 1992], with a worked example.

2 Domain model

In order to simplify the description, I choose a domain whose only source of uncertainty is external events. All the operators are deterministic and the initial state is known with certainty. In addition I ignore the cost of obtaining information: the state is always known with certainty during plan execution.

Consider the following simple transportation problem. A number of packages are to be moved between locations, using combinations of taxis and airplanes. The taxis are used to move objects around within a city, and airplanes to move objects between airports of different cities. The predicates of the domain, with the types they expect, are "in(package, vehicle)", "at(object, location)", "lost(package)" and "have-key(locker)". "Object" is a super-type of "package", "vehicle".

There are seven operators, which have zero duration unless stated:

load takes a vehicle and an object as parameters. The precondition is that the two have the same location and the effect is that the package is in the vehicle.

unload takes a vehicle and an object. The precondition is that the object is in the vehicle and the effect is that the object's new location is the same as that of the vehicle.

drive takes a taxi and a destination. The precondition is that the destination be in the same city as the taxi and the effect is to make the destination the location of the taxi. This action has duration 1 hour, unless the destination is the same as the taxi's location in which case it has no duration.

fly takes an airplane and a destination. This is like "drive", except the locations must both be airports. The duration is 5 hours if the destination is not the current location.

open-locker takes a locker. It provides the key to the locker so it can be used. This operator consumes \$1 in resources.

store takes a package and a locker. The package must be in the same airport as the locker and the planner must be in possession of the key. The effect is to put the package in the locker (a kind of location).

unstore takes a package and a locker. The package must be in the locker, and the planner must have the key. The effect is to remove the package from the locker, placing it at the airport, and swallowing the key.

Event:	`lose-package-from-airport`
Preconditions:	`(at ?package ?airport)`
Deletes:	`((at ?package ?airport))`
Adds:	`((lost ?package))`
Expected time:	$1/\alpha$

Figure 1: The event schema "lose-package-from-airport"

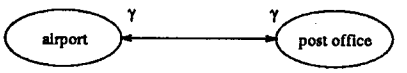

Figure 2: Markov model for taxis

In addition to the operators, we provide resource function $U_r(h)$ that maps an event history to a measure of the resources consumed. While only the "open-locker" operator has an explicit cost attached, there is a charge per hour per package for the use of a taxi or an aircraft. The charge is \$1 per hour for a taxi and \$10 per hour for an airplane, the cost being accrued from the time the package is loaded to the time it is unloaded.

Lastly we provide an explicit model for the probability distributions of external effects. Many researchers have investigated explicit models of external events, e.g. [McDermott, 1982, Haddawy, 1991, Kanazawa, 1992]. Here I take a very simplistic view and model external events in a similar way to operators, with preconditions, add and delete lists, and with an expected time to occur. The intended meaning is that if the preconditions are satisfied, the operator may occur, and we model the probability of its occurrence as exponentially distributed with the given expected time, as long as the preconditions remain satisfied. For example, the event schema `lose-package-from-airport` is shown in figure 1. It can be interpreted as meaning that if a package is left at an airport at time t_0, and no other events occur, then the probability the package is lost at time $t_0 + t$ is $1 - e^{-\alpha t}$. In what follows, I will sometimes refer to the inverse of the expected time of an event as its *volatility*.

In general, reasoning about the outcomes of plans in the presence of external events is very difficult. In this domain, however, the events act on disjoint parts of the representation and hence the projection problem can be decomposed, much as the planning problem can be decomposed for serializable subgoals [Korf, 1987]. Thus the state description can be factored into independent parts, each of which can be modelled by a continuous-time stationary Markov chain.

First we model the uncertainty of the location of a taxi. A taxi can move from one location in its city to any other with equal probability, and the expected time for it to remain in its current location is $1/\gamma$ for each location. This is modelled by the event `taxi-moves`, which in turn induces the Markov chain on taxi locations shown in figure 2, because each taxi's location is independent of the other taxis and the other events of the domain, and because there are only two valid locations for each

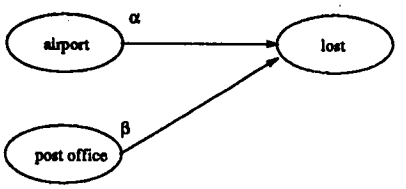

Figure 3: Markov model for packages

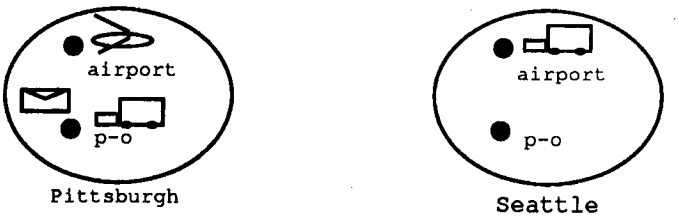

Figure 4: Initial state for the example problem

taxi.

Since there are only two valid locations for each taxi we can easily calculate the probability distribution for the location of a taxi at some future point, given its current location, using the formulae: $P_{\text{same}}(t) = \frac{1}{2}(1 + e^{-2\gamma t})$, and $P_{\text{diff}}(t) = \frac{1}{2}(1 - e^{-2\gamma t})$ [Ross, 1980], where $P_{\text{same}}(t)$ is the probability that the taxi is in the same state at time $t_0 + t$ as at time t_0, and $P_{\text{diff}}(t)$ is the probability that it is in a different state.

Second we model the probability of a package becoming "lost" from certain locations. When a package is lost it cannot be loaded by any vehicle. This is modelled by the two event schemas **`lose-package-from-airport`** and **`lose-package-from-post-office`**. These are similar event schemas, but they have different volatilities. Since no other events affect the locations of packages, they induce the simple Markov chain shown in figure 3 and this in turn allows the location of a package to be modelled with two exponential distributions, with an expected time for the package to be lost from an airport $(1/\alpha)$ and a post office $(1/\beta)$.

Again, it is simple to calculate the probability that a package will be in a location at some future time $t_0 + t$, given that it is there at the current time t_0. This is $e^{-\alpha t}$ for an airport.

3 Planning with external events

We are now ready to consider a simple planning problem, which is illustrated in figure 4. We have a package at the post office in Pittsburgh, to be delivered to the post office in Seattle. For this example I will model the utility of an event history, h, as

$$U(h) = U_g(h) - U_r(h)$$

where $U_g(h)$ is \$200 if the event history h achieves the given goal, and $U_r(h)$ is the resource function, as described in the domain model. This is consistent with a goal-directed agent as defined in [Haddawy and Hanks, 1994]. Initially there is a taxi at the Pittsburgh post office, an airplane at the Pittsburgh airport ready to whisk

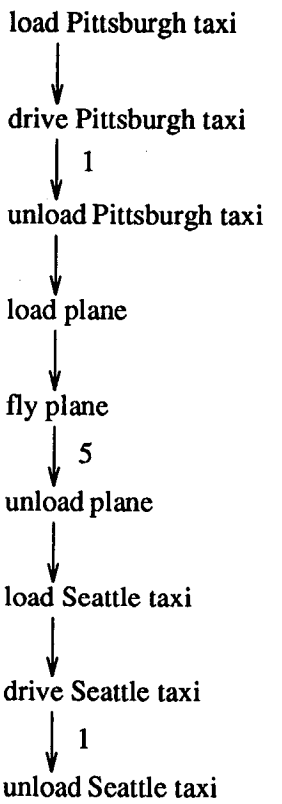

Figure 5: Initial plan (durations noted under operators)

the package to Seattle, and a taxi waiting at Seattle airport.

The planner, as currently implemented, operates in a loop as follows:

1. Generate an initial plan P, to achieve a state satisfying the goal description.
2. If P satisfies the termination criterion, based on expected utility and/or planning resources, return P. Otherwise
   (a) Find a set of potential failures F. Each "failure" consists of a step in the plan, a precondition of that step that may not be satisfied, and a sequence of events which would lead to the precondition not being satisfied. Each event in the sequence is related to a set of steps in the plan during which it may take place.
   (b) Choose a failure to work on.
   (c) Choose one of the following repairs:
   - Create a plan to achieve a goal state from the state produced by the sequence of actions, and add this to P as a conditional branch, should the event sequence take place.
   - Pick an event in the chain of events, and attempt to protect P by adding steps to negate the preconditions of the event at the time it can be applied
   - Reduce the probability of the chain of events by moving the steps of the plan along

a time line in order to reduce the time over which the events can occur.

(d) Go to step 2.

In order to find the best plan, the planner must backtrack over all possible ways to fix a failure.

Since there is no uncertainty in the initial state or in the effects of the operators, a classical planner could be used to obtain the initial plan shown in figure 5. Next we calculate the expected utility of the plan. This is complicated in general [Kushmerick *et al.*, 1993], but the task is much simplified in this case because of the restrictions on the events discussed in the previous section. We use an execution model under which execution ceases as soon as a step of the plan is attempted that cannot be applied, regardless of whether a subset of the plan could still be executed to achieve some benefit. Any conditional execution must be explicitly represented as branching in the plan. Thus we can compute the expected utility by considering the probability that each action's preconditions are satisfied at the time the action is to be applied, and the expected benefit of the state that results from the plan. In general, this is a lower bound on the probability of success because we have not taken into account external events that might produce some utility irrespective of steps in the plan being applied. In this case, however, our model does not contain any such events.

The planner derives the set of possible plan failures, F, as a precursor to calculating the expected utility of the plan. For the initial plan there is only one element in F, which says that the step to load the taxi in Seattle may fail, because the taxi has moved to the post office, either while the taxi in Pittsburgh was moving or while the plane was in flight. The probability of this event occurring over this time frame is $P_{\text{diff}}(6) = \frac{1}{2}(1-e^{-12\gamma})$, where γ is our measure of the volatility of taxis in Seattle. Taxis being what they are, we can assume $\gamma > 2$, so this probability is just slightly less than $\frac{1}{2}$. The utility of the first part of the plan, delivering the package to Seattle airport, is -\$51 (\$50 for the airplane and \$1 for the taxi). The utility of the second part is \$199 if the taxi is present, and is 0 otherwise, giving an estimate for the expected utility of $-51 + \frac{1}{2}(199) = \$48.50$.

Only one of the three techniques available to fix the problem in the initial plan will work: re-planning from the outcome when the event takes place. So the planner adds a conditional branch to call the taxi in the outcome where it is not waiting at the airport. We assume that the taxi will take 1 hour to arrive[2] and go to step 2, calculating the expected utility of the new plan. While the Seattle taxi is arriving, the package can be lost with probability $1 - e^{-\alpha}$, in which case no action can be made to rescue the plan. Taking $\alpha = 1$, we estimate the expected utility of this plan to be $-51 + \frac{1}{2}(199) + \frac{1}{2}e(198) \approx \$85$. Note that this estimate can be calculated incrementally as the plan evolves.

Again there is only one failure predicted for the plan, that of the package being lost, so the planner picks a method to fix it. It is not possible to increase the utility of the outcome if the event takes place, since no operators are then applicable, so instead the system tries to decrease its probability. This can either be done by shortening the time that the system is in the state from which the package can be lost, or by negating the preconditions of the event `lose-package-from-airport`.

[2]The Markov representation is an approximation that ignores the probability that the taxi is in transit.

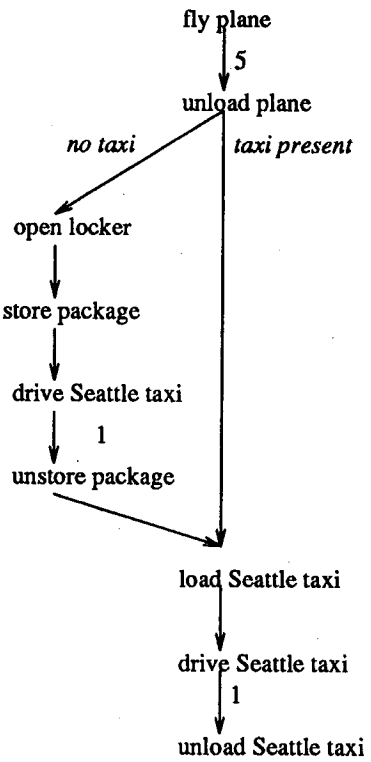

Figure 6: Final portion of the final plan

One option for the planner is to shorten the time in the state during which the event is applicable by entering the state later or leaving it earlier. One way to enter the state later is to delay unloading the package from the airplane until after the taxi has driven to Seattle airport. The package is, after all, safe in the airplane. This repair to the plan can be found by posting constraints on the time intervals over which the steps are applied. This is an attractive repair technique, because no extra steps have to be added.

Since this approach increases the cost of the plan, the planner also tries to negate the preconditions of the event. To do this, it subgoals on the goal of moving the package from the airport as early as possible once it is unloaded from the plane. The two-step plan to open a locker and store the package there is found, with much lower cost than delaying the plane. The cheap extra step of removing the package again once the taxi has arrived is added to keep the plan consistent. The tail end of the final plan is shown in figure 6. Its estimated expected utility is \$147.

4 Discussion and related work

In the last two sections I described a representation for knowledge about the distribution of external events that

can affect a plan, and a planning method that makes use of this knowledge by subgoaling to improve the expected utility of the plan. The plan with the highest expected utility was found by reasoning explicitly about the causes of external events, and subgoaling in order to increase the expected utility of the plan. The planner essentially finds a point in the plan such that most event histories emanating from the point have relatively high probabilities and relatively low utilities. The different techniques used for plan repair either try to increase the utility of event histories from this point, or decrease the probability of reaching the point.

The use of decision theory to suggest goals to work on allows the planner to cope with more richly specified problems than an AI-based planner can typically deal with, including different degrees of plan satisfaction and operator costs. The use of goal-directed search to achieve these subgoals brings with it a focus of attention that allows the planner to cope with problems with higher branching factor than otherwise would be possible. I refer to this combination as "decision-theoretic subgoaling". Planners for dealing with uncertainty in the state and in the effects of operators, such as BURIDAN [Kushmerick *et al.*, 1993], Cassandra [Pryor and Collins, 1993] also make use of decision-theoretic subgoaling to an extent, although they do not have an explicit utility function. By planning for contingencies, and adding extra links to achieve preconditions, they are subgoaling to increase the probability of success of the plan.

I believe that the use of goal-directed search will be necessary for any planner that deals with uncertainty in a reasonably complex domain. To illustrate this point, consider how a planner based on policy iteration, such as used in [Dean *et al.*, 1993][3], would approach this problem. Policy iteration [Howard, 1960] computes the optimal action for each state, given the set of all states and a mapping from an action and a state to a probability distribution of successor states. In order to make policy iteration tractable, Dean et al. restrict attention to an *envelope* of the states contained in plans generated by an initial planner. Various schemes for incrementally increasing the envelope are considered, all based on adding a subset of the *fringe*, or the set of states that can be reached with some probability by executing the current plan. None of these schemes would be able to solve the worked example of this paper, since states with open lockers never appear on the fringe. In addition this path only shows greater utility when the two-step plan of opening the locker and storing the package is considered. The search space for considering all n-step plans from some envelope would quickly become prohibitive, making goal-directed reasoning in some form necessary to generate envelopes for policy iteration in a moderately complex domain.

5 Conclusion and future work

Decision-theoretic subgoaling combines goal-directed search with a richer representation for plan utility. In this paper, I illustrated and motivated the technique with a simple planning problem involving external events with known probability distributions. The explicit knowledge of both the causes and probabilities of the events allow the system to formulate subgoals that will improve the utility of the current plan.

The assumption that the events interact in such a way as to allow the domain to be modelled as a group of small, independent Markov chains is crucial in this paper to allow the expected utility of a plan to be calculated quickly. However it is a very restrictive assumption in general, and one that I want to relax. I have implemented a Monte Carlo simulation technique that works for a much wider class of problems, and converges on the correct value for the expected utility. I will begin a series of experiments to investigate the tradeoffs between the Monte Carlo simulation and the use of Markov models as an approximation to the domain. Since convergence of the simulation can be slow, however, I am also interested in techniques which exploit knowledge of independence in the form of influence diagrams [Pearl, 1988], and more systematic modelling techniques.

Finally, this paper ignored the effect of the time taken to plan on the utility of the resulting plan, concentrating on off-line planning. However, the analysis of dynamic planning domains and the increased planning complexity both point to a need to take this into account. The use of Monte Carlo simulation also lends itself well to techniques such as anytime planning [Dean and Boddy, 1988]. I intend to relate the work described here with previous work in that area [Blythe and Reilly, 1993].

Acknowledgements

Jaime Carbonell provided many useful suggestions for this work, and discussions with the Prodigy group were very helpful. The paper was improved by comments on earlier drafts from Rujith De Silva, Rich Goodwin, Scott Reilly and Bryan Loyall.

References

[Blythe and Reilly, 1993] Jim Blythe and W. Scott Reilly. Integrating reactive and deliberative planning for agents. Technical Report CMU-CS-93-135, Carnegie Mellon University, May 1993.

[Carbonell *et al.*, 1992] Jaime G. Carbonell, and the Prodigy Research Group. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University, June 1992.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. An analysis of time dependent planning. In *Proceedings AAAI-88*, pages 49–54. AAAI, 1988.

[Dean and Kanazawa, 1989] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[Dean *et al.*, 1993] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *National Conference on Artificial Intelligence*, Washington, DC, 1993.

[3] suitably generalised to reason about external events

[Haddawy and Hanks, 1994] Peter Haddawy and Steve Hanks. Utility models for goal-directed decision-theoretic planners. *Artificial Intelligence*, to appear, 1994.

[Haddawy, 1991] Peter Haddawy. *Representing Plans Under Uncertainty: a Logic of Time, Chance and Action.* PhD thesis, University of Illinois at Urbana-Champaign, December 1991.

[Howard, 1960] Ronald A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, 1960.

[Kanazawa, 1992] Keiji Kanazawa. *Reasoning About Time and Probability.* PhD thesis, Brown University, May 1992.

[Koenig, 1992] Sven Koenig. Optimal planning using markovian decision theory. Technical report, UCB, 1992.

[Korf, 1987] Richard E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[Kushmerick *et al.*, 1993] Nicholas Kushmerick, Steve Hanks, and Daniel Weld. An algorithm for probabilistic planning. Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington, June 1993.

[McDermott, 1982] Drew V McDermott. A temporal logic for reasoning about processes. *Cognitive Science*, 6:101–155, 1982.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[Pryor and Collins, 1993] Louise Pryor and Gregg Collins. Cassandra: Planning for contingencies. Technical Report 41, The Institute for the Learning Sciences, June 1993.

[Ross, 1980] Sheldon Ross. *Introduction to Probability Theory.* Academic Press, 1980.

[Thiebaux *et al.*, 1993] Sylvie Thiebaux, Joachim Hertzberg, William Shoaff, and Moti Schneider. A stochastic model of actions and plans for anytime planning under uncertainty. Technical Report TR-93-027, The International Computer Science Institute, May 1993.