From: AAAI Technical Report SS-95-07. Compilation copyright © 1995, AAAI (www.aaai.org). All rights reserved.

AI Planning in Dynamic, Uncertain Domains

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

Abstract

A major reason for the success of the STRIPS planner and its derivatives was the use of a representation that succinctly captured the changes in the world due to different actions. As planning systems are extended to deal with uncertainty, we seek ways to model the world that maintain this advantage. In this paper I examine four somewhat representative planning systems designed for uncertain domains that can change independently of the actions performed by the planning agent. These are Buridan, anytime synthetic projection, Weaver and XFRM. I compare their positions on several design issues including the power of the languages used to represent plans and actions, and the use to which it is put in both plan synthesis and plan evaluation. I briefly explore the tradeoff that exists between the expressiveness of the language used to reason about the effects of action in the world and the extent to which the planning system can take advantage of the language to build plans efficiently.

Introduction

AI planning systems gain much leverage from the combination of means-ends reasoning and subgoaling. As these systems are applied to more complex domains we must question whether this approach will scale to richer action, goal and plan representations. There are encouraging indications for uncertain action outcomes, eg (Pryor & Collins 1993, Kushmerick, Hanks, & Weld 1993), we would like to push the envelope to include external events and richer temporal models for goals, plans and actions.

In this paper I briefly describe and compare four planners that deal with uncertainty in initial conditions and the effects of actions, three of which reason about events explicitly. The systems have made different sets of design decisions, and after presenting them I will briefly discuss what we might learn from them about applying planning techniques to more involved domains.

The systems I describe are as follows:

- Buridan and C-Buridan (Draper, Hanks, & Weld 1994, Kushmerick, Hanks, & Weld 1993) use a partial-order causal-link planner designed for an action representation that models different possible outcomes of an action probabilistically.
- *Anytime synthetic projection* (Drummond & Bresina 1990) uses a model of both actions and exogenous events, building up a conditional plan by forward chaining, and reasoning about the planner's actions and external events in the same step.
- Weaver (Blythe 1994) uses a state-based planner (Carbonell *et al.* 1992) to improve a plan, guided by a critic that alters the planner's goals, initial state and action descriptions based on a probabilistic analysis of the plan.
- XFRM (McDermott 1992) uses a transformational planner to improve plans built from a library. Plan transformation is achieved by pairs of critics and repair routines, some of which are general while some are domain-dependent.

In the next section I describe the languages and algorithms used by each of these systems in more detail. Where possible I have used a uniform language to describe each system to make the comparison easier. This has not been done for XFRM, however, which can use arbitrary lisp code as part of its action definition.

Although the descriptions of the systems are too brief to do them justice, I mention some broad issues in this approach to planning under uncertainty in the third section.

The example systems

I begin this section with a description of the language I use to discuss the various planning systems.

A probabilistic extension to ADL

Buridan and Weaver both have action representations that are simplifications of Pednault's ADL (Pednault 1986). This language has a syntax similar to STRIPS (Fikes & Nilsson 1971) but is equivalent in representational power to the situation calculus (McCarthy & Hayes 1969). In Pednault's formalism, a state is represented as an algebraic structure, specifying a domain, a set of relations and functions over the domain, a set of distinguished objects of the domain and a mapping from a set of variable names to elements of the domain. Since neither Buridan nor Weaver use variables in their plans and all reachable states are constrained to have the same domain, I simplify the description here by allowing states to be simply symbols, elements of the set S. Each planning domain specifies in addition to S a set of objects O and a set of relations R. Each relation $r \in R$ has some arity n_r, and for each state $s \in S$ and relation r a

subset of O^{n_r} is defined. For example if O is the set { truck1, package3, london, pittsburgh } and in is a binary relation in R, then the state s might map in to the set { (truck1, london), (package3, pittsburgh) }, called its "interpretation" in s, written $I(\mathsf{in}, s)$. Note that while each state must associate in with such a set, not every subset of O^2 need correspond to a state. In addition I require that no two states have the same interpretation for each relation.

Actions in ADL are partial mappings from individual state to individual state. The domain of an action $\mathcal{A}$, the subset of S for which the mapping is defined, must be specifiable by a logical expression in the relations R, called the precondition of $\mathcal{A}$. The mapping itself is specified through the interpretations of the relations, which are in turn specified through add and delete lists for each relation $r \in R$. Thus if $\mathcal{A}$ has add and delete lists α_r and δ_r respectively for r, and applying $\mathcal{A}$ in state s leads to state t, then $I(r,t) = (I(r,s) - \delta_r) \cup \alpha_r$. We define the set of adds and deletes specified for every relation in R as the *effect set* ϵ of $\mathcal{A}$. The state t defined as above is denoted result(s, ϵ).

The planning systems described here model actions as producing a probability distribution over S when performed in a state in S. To do this they use probability distributions over finite sets of effect sets, called *effect distributions*. These can be written as a set of pairs, $\{(p_1, \epsilon_1), \ldots (p_n, \epsilon_n)\}$ for some finite n, where each $p_i > 0$ and $\sum_i p_i = 1$. If an action is specified with this effect distribution and is performed in some state s, then the probability distribution for the resulting state is given by the set $\{(p_1, \text{result}(s, \epsilon_1)), \ldots, (p_n, \text{result}(s, \epsilon_n))\}$. Finally, the actions are generalised to have a binary tree of effect distributions, whose internal nodes are relations and appropriate n-tuples of O, for example "in(truck1, pittsburgh)" and whose leaves are effect distributions. When these actions are performed, the tree is traversed, taking at each internal node the "true" or "false" branch depending on whether the n-tuple is in the relation's interpretation for the current state, and the effect distribution found at the leaf is used to build the probability distribution for the resulting state. I will refer to this binary tree as an action's *effect distribution tree*, and will write the probability distribution over S that results from performing action $\mathcal{A}$ in state s as $P(.|s, \mathcal{A})$.

C-Buridan

Buridan and C-Buridan (Draper, Hanks, & Weld 1994, Kushmerick, Hanks, & Weld 1993) are partial-order planners based on SNLP. C-Buridan is an extension of Buridan that allows information-gathering operators and conditional plans. They use the action representation described above, with two simplifications. First, each action's precondition is "true", so the action is applicable to every state in s, with results depending on the tree of effect distributions. Second, the planners use a propositional logic, so all the relations in R have arity 0. C-Buridan in addition models information-gathering actions that have observation labels as well as effects.

The planners are given an initial probability distribution over S, a goal description G and a threshold probability p_t. They seek plans that will reach a state satisfying G, from one drawn from the initial probability distribution, with probability at least p_t. Their algorithm is based on SNLP, which is described in detail in (McAllester & Rosenblitt 1991).

The systems begin by defining a "start" operator, whose effect distribution tree encodes the initial state probability distribution, and a "finish" operator whose effect distribution tree has only one useful leaf that adds the proposition "goal" with certainty, and whose path through the tree encodes the goal statement. An initial plan consisting of these two operators is put on the list of open plans. The planning algorithm used repeatedly selects the first plan in this list, and estimates its probability of success. If this is shown to be above the threshold, it is returned and the plan terminates. Otherwise, the plan is modified in several ways, the modified plans are evaluated according to a heuristic that returns a number and are placed in the list of open plans, sorted according to this ranking number.

The first way a plan may be modified is by adding a causal link from a particular outcome (effect set) of a new or existing action to a goal, an internal node on a path to an outcome that already has a causal link. Second, an ordering constraint can be added between two actions. This option is considered if there is a "threat", a possibility that one action may stop a causal link from working if it is executed between an establishing outcome and the goal. The action may be ordered either before the outcome or after the goal if this is consistent with the other orderings and causal links. Third, causal links can be added with the aim of reducing the probability of a threatening outcome of the threat action taking place, essentially by taking as a subgoal the negation of some step on the path in the effect distribution tree that leads to the bad outcome. C-Buridan can also modify plans by branching, resolving a threat by incrementing the context of steps subsequent to an information-gathering step with observation labels from that step, and ensuring that the threatening step has a different context from the threatened link — in other words, ensures that it would be in a different branch of the plan.

The stage of estimating the probability of success of a plan in Buridan can be done in one of four different ways, described in (Kushmerick, Hanks, & Weld 1993). None of these uses information that is not present in the plan or the domain representation. The default method finds a lower bound for the probability by considering the causal link structure of the plan.

Some important points to note about Buridan and C-Buridan from the point of view of this paper are these. The plan evaluator and the planner work with the same language for actions and plans. The planners include two steps that are not found in their deterministic counterpart: confrontation and branching. Confrontation is essentially a new way of determining subgoals, which are treated as before. The language used allows a probabilistic representation of uncertainty in the initial state and in action outcomes, but not in exogenous events.

Anytime Synthetic Projection

Anytime synthetic projection (ASP) (Drummond & Bresina 1990) models both actions and exogenous events, specified as probabilistic state transitions. It represents its plans as a collection of situated control rules that map descriptions of a state to chosen actions, and attempts to maximise the probability that the rules will result in behaviour that satisfies its goals, described below, although no guarantee of maximum probability is given.

Goals are given to the planner in terms of "behavioural constraints", which can specify predicates that are to be true over certain temporal conditions. The goal is represented as a "behavioural constraint strategy" which is a partially ordered set of behavioural constraints. Specifically, behavioural constraints are constructed according to the following grammar:

$\beta \rightarrow$ (and $\beta_1 \ldots \beta_n$) | (or $\beta_1 \ldots \beta_n$)
$\beta \rightarrow$ (maintain $\psi \tau_1 \tau_2$) | (prevent $\psi \tau_1 \tau_2$)
$\beta \rightarrow$ (maintain $\psi \phi_1 \phi_2$) | (prevent $\psi \phi_1 \phi_2$)
$\psi \rightarrow$ (and $\psi_1 \ldots \psi_n$) | (and $\psi_1 \ldots \psi_n$) | predicate

Here, the symbol β represents a behavioural constraint, ϕ a time variable, τ a time constant and ψ a formula. Time points are natural numbers.

The algorithm searches for combinations of actions and events that satisfy a behavioural constraint strategy using a forward-chaining filtered beam search (Ow & Morton 1986). First, the current state is defined as the initial state. A behavioural constraint β that is not preceded in the partial order is chosen to work on. From the current state, all possible transitions that pass a filter based on their probability are considered as possible successor states. For each of these, a heuristic value is calculated based on the state's probability and estimated remaining work to satisfy β. A subset of the successors with highest heuristic value are added to the list of open situations, from which the next current situation is chosen. The size of this subset is fixed by the beam-width parameter of the search. When a state is chosen whose path from the initial state satisfies β, the system compiles search control rules that specify the actions chosen along the path, picks a new behavioural constraint from the strategy and takes this state as the new initial state. Thus the system eagerly commits to a subplan for each constraint in turn, a strategy called "cut and commit". Actions and events have durations, which are used to give a time stamp to each state. These are used to verify that a state-action sequence adheres to a temporal behavioural constraint.

In this system the planner and the projection system work as one, which the authors term "synthetic projection" to distinguish it from "analytic projection" which is separate from the plan creation step, as in Buridan. The system uses a richer representation for goals and actions, but uses a relatively weak planning algorithm. Forward chaining on applicable actions can lead to many possible successor states, and the system is left to the mercy of the local heuristic when pruning takes place. Since the heuristic counts separate predicates in a conjunction equally when estimating remaining work, and also because of the cut-and-commit strategy, it is not clear how the planner can deal with goal interactions. Backtracking over previously planned-for behaviours is briefly discussed, although the authors consider leaving the compiled control rules in place.

Weaver

Weaver (Blythe 1994) is a planning system that deals with uncertainty in the initial state, action outcomes and exogenous events. It uses a probabilistic extension of ADL to represent actions, similarly to Buridan and C-Buridan. It also represents exogenous events in the same way and creates subgoals based on their preconditions. In addition to the precondition and effect distribution tree, an action $\mathcal{A}$ also has a duration d($\mathcal{A}$) and a single effect set inter($\mathcal{A}$) that models the state during the execution of the action. Thus, in the absence of exogenous events, if action $\mathcal{A}$ is performed in state s, the state is immediately transformed to s' = result(s, inter($\mathcal{A}$)), and the probability distribution for the state at d($\mathcal{A}$) time units later is $P(.|s', \mathcal{A})$, based on $\mathcal{A}$'s effect distribution tree as defined in section . In addition to a single effect set, an event $\mathcal{E}$ has a probability p($\mathcal{E}$), which is its probability of occurrence over one time unit given that its preconditions are satisfied.

If exogenous events are allowed to take place over the duration of the operator's application, the probability distribution over S that arises is more complicated but still well-defined, under the constraint that each pair of events has either a disjoint domain or a disjoint effect set. That is, if two events can take place in the same state, they have orthogonal effects and are therefore order-independent. The actual probability distribution is a little involved and can be found in (Blythe 1995).

Weaver uses the Prodigy planner (Carbonell *et al.* 1992) as a subroutine and builds up a conditional plan by calling Prodigy with different initial conditions, and occasionally modifying its actions. Weaver models a plan using a Bayes net to calculate its probability of success and to choose bugs to fix, an approach similar to (Dean & Kanazawa 1989). Rather than include a node for each state variable at each time point, however, the net is computed in such a way that nodes are added on demand, when there is a possibility that an event or undesired action outcome may adversely affect the overall plan.

Weaver's algorithm works as follows: first, the Bayes net is initialised to reflect the probability of the goal being true in the initial state. If this is above a user-defined threshold probability the algorithm terminates. Otherwise a node in the Bayes net is found which with sufficiently high probability does not have a value for which the plan succeeds, along with a parent node that explains the failure - either representing an event or an action with an outcome that contributes to the failure. Weaver tries two ways to improve the plan to fix this problem: either planning under the assumption that the bad outcome takes place and creating a conditional plan, or amending the plan to reduce the probability of the bad action outcome or event. If neither of these techniques works, Weaver tries to fix a different problem in the plan. If no problem can be fixed, it backtracks over earlier planning

choices.

When Prodigy plans as a subroutine to Weaver, it plans for a single initial state and ignores the possibility of exogenous events taking place in the domain. Prodigy therefore ignores the uncertainty of the domain and yields faulty plans that Weaver analyses and attempts to fix. The clear boundaries of the part of the system that deals with uncertainty and the part that doesn't can lead to great inefficiencies as Prodigy may produce plans with very low probabilities of success.

XFRM

XFRM (McDermott 1992) is a transformational planner that works with the Reactive Plan Language (RPL) (McDermott 1991). This language is very rich, and the planner is very wide in scope, so I will concentrate here only on aspects of the language most relevant to the planning system, and the planning system itself. In particular, much of the work in this system on interleaving planning and execution will be ignored.

Like Buridan, the planner creates partially-ordered plans. However, RPL also has facilities for loops, and can include arbitrary lisp code in its plans. Rather than derive plans from goal descriptions and the dynamics of the world, XFRM relies on libraries of plans installed by the user, and applies plan transformations to try to improve their expected performance. This is currently measured as a weighted difference of the number of top-level user goals achieved and the time the plan takes. Although the system is capable of representing exogenous events, it is not dealt with in the current version of the planner.

Given a set of top-level goals, XFRM initialises its list of plans with one built from the plan library, gets a set of projections for this plan and runs its plan critics to find its "worst" bug. Then it repeatedly selects a plan from the list, send it to the executor, generates a set of new plans by fixing the worst bug and for each of the new plans generates a set of projections of the plan and runs its plan critics on the projection set to determine the worst bug and score the plan. Since this routine is designed to be run concurrently with execution, the termination criterion is that execution has finished.

The key operations are plan projection, running plan critics and fixing the worst bug. A plan is evaluated by making a set of projections. The projector takes a partially ordered plan and on each call generates one scenario of running the plan — a totally-ordered set of events in a timeline. Reasoning about the effects of actions is done by a set of rules that specify the events that take place when actions are performed, and the effect that events have on the world in terms of "occasions", which are predicates with a temporal extent, and clipping rules. Typically an action is modelled with two events representing its start and finish. Forward-chaining rules are used to add occasions for events and to specify what occasions an event might "clip", or make untrue. Backward-chaining rules can find which events might add occasions in response to queries, for example to establish if the next chosen action is applicable. Typically, XFRM runs about 3 projections per plan, using them to estimate the goodness of the plan and the severity of the worst bug.

Plan critics are separate routines that analyse the set of projections for various bugs. Some of these are domain-dependent, specified by the user, but some are generic, such as a protection-violation critic that notices when parts of the partially ordered plan may interact with each other. The transformations made to the plan to fix a protection violation essentially add ordering constraints, similarly to one of Buridan's responses to a threat.

In principle the planner has access to the same representation as the projector, since a critic and a bug repair routine can look at any part of the plan. In practice, however, the generic critics do not make use of much more than explicit protection statements and ordering statements in the plan, and most of the RPL language is used by the projector and controller.

Discussion

Although they use quite different planning techniques, the four systems described in this paper use largely similar representations for actions and events. They all model uncertainty in the effects of actions probabilistically, and those that model exogenous events use probabilities for them too. Those which model action durations model them either as fixed (ASP and Weaver) or as functions of the state (XFRM); none model probability distributions over the durations. Nevertheless, there are differences in the representations used, both in their power and their flexibility. In this section I hope to draw out some of these differences and explore the extent to which they depend on the algorithms used, and the extent to which they are arbitrary and could be swapped between the different systems with little penalty.

One fundamental difference that leads to different design choices for the systems is the relation between planning and execution. Buridan and Weaver both concentrate on building a plan off-line, before execution begins, while ASP and XFRM both attempt to interleave planning and execution. This most directly affects the representation of plans. ASP uses situated control rules and XFRM a full programming language, while C-Buridan and Weaver both use sets of actions including branches, C-Buridan's partially ordered and Weaver's totally ordered. The plan representation used by XFRM in turn makes it hard to use any evaluation method other than collecting a set of projections. Estimations of probabilities can be slow to converge in such schemes, but this does not pose a serious problem for XFRM, since it is only likely to mistakenly swap a plan for a worse one when their actual values are quite close.

Analysing the task network for a projection can yield as much information as the more direct evaluations of Weaver and Buridan. What restricts XFRM from using them in general is the need to specify, for example, a confrontation method that can cope with plans involving loops and semaphores. However there is no reason why critics couldn't be provided that can mimic the more powerful planning refinements when the plans are simple enough.

At the other end of the spectrum, we might consider how far the plan and goal languages of Buridan and Weaver

could be scaled up while their planning techniqes were still effective. There is plenty of room for improvement in the action model used by both systems. For example, probabilistically independent effects of actions are currently split up artificially in effect probability distributions only to be re-joined with multiple causal links or plan branches.

An SNLP-based planner could probably cope with a representation of exogenous events such as used by Weaver, applying steps similar to threat resolution to reduce the impact of unwanted events. However, Buridan's probability evaluation technique using causal links would no longer produce a lower bound in general unless all possible events were analysed for a plan. Weaver's model of exogenous events allows planning techniques to be applied to a restricted class of dynamic uncertain planning problems, but it remains unclear whether uncertainty can be combined with richer temporal models of goals or actions in such a way that the simple goal-reduction techniques of classical planners can be of use. The key may lie either in transforming plan bugs into more well-understood subgoals, as is done in Weaver, or in enlarging the set of plan refinement techniques as in Buridan and C-Buridan.

References

Blythe, J. 1994. Planning with external events. In de Mantaras, R. L., and Poole, D., eds., *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*, 94–101. Seattle, WA: Morgan Kaufmann.

Blythe, J. 1995. Bayes nets for planning with uncertainty. Technical report, School of Computer Science, Carnegie Mellon University.

Carbonell, J. G.; Blythe, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Kahn, D.; Knoblock, C.; Minton, S.; Perez, A.; Reilly, S.; Veloso, M.; and Wang, M. 1992. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University.

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

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

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In Hammond, K., ed., *Proc. Second International Conference on Artificial Intelligence Planning Systems*, 31–37. University of Chicago, Illinois: AAAI Press.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proc. Eighth National Conference on Artificial Intelligence*, 138–144. AAAI Press.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Goldman, R. P., and Boddy, M. S. 1994. Epsilon-safe planning. In de Mantaras, R. L., and Poole, D., eds., *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*, 253–261. Seattle, WA: Morgan Kaufmann.

Kushmerick, N.; Hanks, S.; and Weld, D. 1993. An algorithm for probabilistic planning. Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proc. Ninth National Conference on Artificial Intelligence*, 634–639. AAAI Press.

McCarthy, J., and Hayes, P. 1969. *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, volume 4. Edinburgh, Scotland: Edinburgh University Press. chapter 4, 463–502.

McDermott, D. 1991. A reactive plan language. Technical Report YALEU/CSD/RR/864, Yale University.

McDermott, D. 1992. Transformational planning of reactive behavior. Technical Report YALEU/CSD/RR/941, Yale University.

Ow, P., and Morton, T. 1986. Filtered beam search in scheduling. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University.

Pednault, E. P. D. 1986. *Toward a Mathematical Theory of Plan Synthesis*. Ph.D. Dissertation, Stanford University.

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