A Framework for Planning Under Uncertainty

Jonathan King Tash
Group in Logic and the Methodology of Science
University of California
Berkeley, CA 94720
tash@math.berkeley.edu

Abstract

This work presents a planning paradigm which integrates the flexibility and clear semantics of decision theory with search procedures providing many of the efficiency properties of classical heuristic planning methods. We define plans in a manner similar to the classical paradigm but extended to represent nondeterministic actions, using utilities to represent goals. We then outline a search procedure for good plans which is guided by decision theory, and discuss knowledge representation issues from the point of view of improving decisions. This paradigm puts plan choice on the same level as action choice, and has as emergent properties variants of desirable features of more classical planners, such as abstraction and reactivity.

Introduction

Classical planning has developed under assumptions of a deterministic description of the effects of actions. In contrast, decision theory allows one to model problems where this assumption fails but, applied naively, leads quickly to intractability. This paper develops a methodology which may avoid these drawbacks. We define plans in a manner similar to the classical paradigm but extended to represent nondeterministic actions, using utilities to represent goals. We then outline a search procedure for good plans which is guided by decision theory, and discuss knowledge representation issues from the point of view of improving decisions.

The planning methodology described below puts plan choice on the same level as action choice. This allows the planner to control its computational efforts in the same manner it controls its actions. Other work (notably [Russell and Wefald 1991]) has considered decision-theoretic control of computation, but without explicitly including future planning efforts in the descriptions of plans. Such inclusion increases the planner's ability to adapt to time pressure and respond to unexpected situations. Also, by scheduling future deliberations, a planner can consider underspecified courses of action, providing a form of abstraction.

This perspective also makes clear that the point of knowledge representation is to inform decision making. Previous work (such as [Breese 1990]) has attempted to integrate standard knowledge representation methods with decision procedures, but consideration of the planning process has consequences for the form a representation should take. For example, the representation should include abstractions which simplify the computations required by the planner. Some of these consequences are explored below.

Plans and their value

We can describe a plan as a course of action designed to achieve certain goals. The planner's problem is to choose an appropriate course of action. If we can describe the problem in decision-theoretic terms, we can use the calculus of probabilities and utilities to guide this choice.

Let us consider a world whose state at any given time is described by a set of attributes or propositions which hold in that state (as in the situation calculus of [McCarthy 1968]). The planner has at its disposal a set of executable actions, which map from a given state to a set of possible results, each of which is a particular state and the time the action took to reach that state, weighted by their probability. Thus, the planner is uncertain what state a given action will leave it in, and how long it will take, but knows how likely the various results are. Even an action of "doing nothing" can have uncertain consequences if the planner has a nondeterministic model of the dynamics of the world. (This model for actions is similar to the "domain causal theory" of [Drummond and Bresina 1990].) For simplicity, we will assume that these maps are complete, or applicable in any state, though the notion of an action precondition can be captured by mapping unreasonable initial states into a "dead" state. We will also assume for now that these action maps are fixed, although how they might change with experience will be discussed below.

A planner is supposed to choose its actions so as to further its goals. Let a world line be a possible history of the world, a function from time to the set of world states. We can then describe goals in terms of a utility function over world lines, associating a real number value to each possible history. (For example, a planner whose goal is to win a game of chess might assign a utility of 1 to each

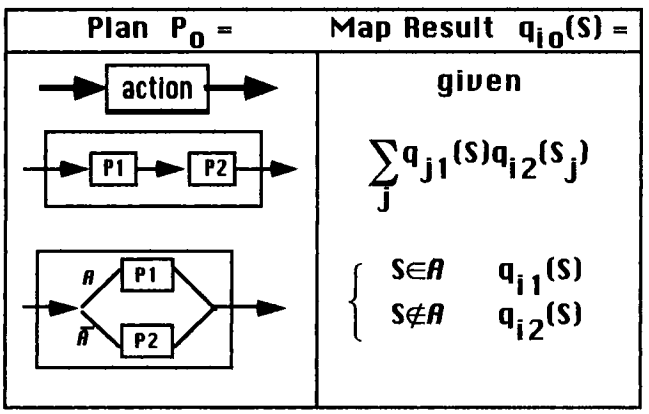
*This work was supported by the National Aeronautics and Space Administration.

world line in which it wins, a utility of zero to those in which it ties, and -1 to losses. Its actions could be possible moves followed by a probabilistic description of its opponent's response, with illegal moves mapping to states of high negative utility.) Thus, individual world states, and hence actions, do not necessarily have an intrinsic utility independent of future developments, but only as part of a history. (Of course, different states can be more or less likely to lead to valuable world lines, depending on later actions, and hence inherit an expected utility from their expected futures.)

We are now in a position to define a plan. We will proceed inductively. The following are plans:

- Any action.
- One plan followed by another.
- One plan or another, depending on the world state.

This definition admits to various generalizations (for example, a more complicated scheduling of subplans, allowing for concurrent actions), but will suffice for the ensuing discussion. Thus, a plan is a sequence of actions, where action choice can be conditioned on the current state of the world.



Inductive Plan Construction

$q_{ik}(S)$ is the probability of getting result S_i from plan P_k with initial condition S

Note that a plan implements a map similar to those of actions. One can calculate the map implemented by two consecutive plans by averaging the results of the second over its various possible initial states, weighted by their probability of resulting from the first. A conditional plan maps each state to the result of the plan appropriate to that state. Alternatively, each plan can be seen as a map from states to probability distributions over possible future world lines. A plan has an expected utility, which is just the average over world lines of their respective utilities, weighted by their probability of resulting from that plan. So, with each plan we can associate two functions:

- map: initial state —> probability distribution over future world lines.
- value: map x initial state x current time x goals —> utility.

A planner with adequate computational resources and perfect knowledge of the world state would do best to choose the plan of highest expected utility. (Incomplete knowledge of the world state will be dealt with below.) Of course, given the combinatorial nature of the space of plans and the limited computing capacity of the planner, finding such a global optimum will in many cases be impractical.

This representation does not explicitly include partially ordered plans, but if the result of a sequence of actions is independent of their order, then its utility will not depend on that order, and multiple orderings can be treated as equivalent for purposes of optimizing plan choice. Similar considerations hold for plans invariant under replacement of one action by another. Also, deterministic actions can be considered to be adding and deleting attributes of the world state, depending on the current state, similar to classical planning actions. (See [Christensen and Grove 1991] for a description of classical nonlinear plans.)

Computations, treated as actions (as in [Russell and Wefald 1991]), fit into the above representation without modification. The utility of a plan containing computations depends as usual on the relative likelihoods of the resulting world lines. If state attributes can be divided into "internal" and "external" classes, with utility and noncomputational actions depending only on external attributes, and computations affecting only internal attributes, then a computation's only effect on a plan's utility can come through the dependence of action choice on internal state. Under these conditions, the optimal plan need contain no computations, since all actions affecting utility have nontrivial maps only over the external state.

If a planner does not have direct access to the external state, then its actions must include observations which, like computations, affect its internal state. Such a planner can only condition its actions on internal state.

Time is explicitly modelled in this representation, both in the definition of actions and in the description of the planner's goals. One important external consequence of computation is the consumption of time. The utility of a world line often depends on how soon a certain state is achieved, causing the utility of a plan to achieve that state to decrease with the time it takes. For example, if the goal is to achieve a certain state by a specified time, then a particular plan's utility is proportional to its probability of achieving that state by the specified time. If this probability is a simple function of time, then one can easily calculate the utility cost of adding delays (such as calculations) to the plan.

A computationally limited planner

We have defined plans and their value, but have not yet specified a mechanism for efficiently finding plans of high value. The heart of most planners is their algorithm for finding appropriate plans. We must therefore adapt the

above discussion to planners able to devote only limited computational resources to searching for a good plan.

If we consider the planner's search efforts to be computational actions in the above sense, then the planner can be considered to be executing a set plan, chosen by the designer, at all times, and its internal efforts to "choose a good plan" are simply ways this plan conditions its choices among future actions based on its computations. As designer, we are faced with the same difficulties as the planner in choosing an optimal plan, and so are forced to simply supply a plan which we hope will generally function well. In an effort to produce an algorithm of general use, we can define a computational paradigm designed to choose among various courses of action using decision-theoretic principles, basing these choices on the information available to the planner before computation. This paradigm has several nice properties, such as allowing for explanation of why certain actions are chosen in terms of standard decision theory criteria. It is possible, however, that the cost of such an algorithm is great enough that the planner would be better off using an alternative means of choosing actions. (In fact, if the designer can "precompute" the optimal response to a given situation, that response should be built into the planner, avoiding the need for computation.)

Let us assume that the planner knows (i.e. has an explicit representation for) the goals and the noncomputational action maps, and wants to choose a plan of high utility. However, the planner does not know, before computation, the utility of each plan, or even the map each plan implements. The planner's motivation for computing, then, is to improve its guess as to the best future course of action (possibly including further computation), which it will then choose to execute. We can therefore specialize our planning framework by allowing only actions of these forms:

- Any noncomputational action.
- An update on the estimated value of a plan.

We can similarly restrict conditional plans to be of these forms:

- One plan or another, depending on the external world state (or its internal representation).
- One plan or another, depending on their estimated value.

We can then describe the planner as executing the set plan "execute the plan of highest estimated value." We have only to describe the updating mechanism to completely specify the planner's behavior.

This planning mechanism gives a form of reactive behavior, or appropriate response to time pressure. Since the value of a plan depends on the time it takes, and computations take time, a planner with reasonable value estimates would choose a course of action involving less computation when time is critical, even if such a plan has deficiencies which could be fixed by further computation. For example, if the planner is considering plans beginning with a nondeterministic action, it may choose to simply take the action and then compute the best plan given the results, instead of computing the best plan for every possible result of the action, even though the latter would give the planner a better estimate of the action's expected value. The planner will schedule its computational efforts, as well as its external actions, according to its estimate of what will best achieve the goals. The "value of information" provided by a computation is thus estimated in a manner homogeneous to the value estimation of any other action.

The planning mechanism also provides a form of abstraction. A plan to update the estimated values of a set of plans and then choose the one of highest estimated value can be seen as an abstract representation of the considered set of plans. For example, a plan to estimate the value of shopping at various stores and then choose the highest value plan can be considered a plan to shop at some unspecified store. Thus, an abstract plan is one that requires computation to determine the resulting course of action. Note that changing the expected value of an abstract plan can result in changing the expected value of all the plans it is an abstraction from, so this mechanism easily allows for local updates to have global effects on the knowledge base. It also allows for a plan to be built from abstract subplans which only need to be fleshed out when required for execution or more accurate assessment of the plan.

Knowledge about plans

We will now describe a representation for the planner's knowledge about plans and a mechanism for updating that knowledge. What the planner wants to know is the plan with the highest utility. It would certainly be sufficient to know the utility of each plan. The planner does not know this, but it can describe its guess as a probability distribution over functions mapping plans to utilities. We can simplify this description by just associating with each plan a probability distribution over utilities. (This is a distribution over the expected utility of a plan, not the distribution over world line utilities that is used in calculating the true expected utility of a plan.) Such a description does not explicitly capture known dependencies between the utilities of various plans, but some of these dependencies can be expressed in the rules used to update the estimated utility of a plan depending on the estimated utility of other plans. If we assume that the utility of a plan is readily derivable once we know the map implemented by a plan, we need only store for each plan a probability distribution over these maps. (This assumption can be removed if we store for each plan an estimate of its utility for each initial state and time, and allow a computational action which updates this estimate based on knowledge of the plan's map.) In fact, we can average with respect to this probability distribution to get an "effective map" for each plan. This map assigns to each possible resulting state a probability which is the average of the probabilities assigned by the possible maps. The "effective map" is all that is needed to compute the estimated utility

of a plan. We will therefore use a knowledge representation which associates with each plan such a map, an estimate of the true map implemented by the plan.

One advantage of storing primarily information about these maps is that such information is reusable by the planner, since it doesn't depend on the goals the planner may have. (Which information is important to know accurately will still depend on the goals.) Another is that since these maps are inductively defined, knowledge about a subplan's map is all that is needed to calculate the contribution of that subplan to any plan of which it is a part. This observation immediately suggests an update mechanism: update the knowledge about a plan's map by considering the plan to be a combination of subplans (in the form "A then B" or "A or B") and setting its map to the appropriate combination of the current estimated maps of the subplans. The update mechanism can be simplified by only adjusting the map between states of current interest. The choice of plan to update can be treated like any choice between actions -- it is made by following the plan currently estimated to be of highest utility. Of course, an update can have high utility only when the planner has better knowledge about the subplans than about the plan to be updated.

We have defined the action "update plan X" for the case where X is a combination of subplans. When X is a computational action of the form "update plan Y," we should again provide an improved estimate for the map implemented by X. This map takes the maps for subplans of Y and produces a map for Y. An estimate for it can be represented as taking maps for subplans of Y and producing a probability distribution over maps for Y. This distribution should be singular, putting all weight on the known value, when Y has already been updated using the given subplan maps. If one knows of dependencies between these probability distributions (such as a notion of smoothness for the update map), then producing these singularities will affect other map values as well. For example, one could cluster maps with similar features, such as involving Ys similarly composed from subplans and having a similar probability distribution over world lines. One could then estimate an update map using statistics gathered over updates performed in its cluster, such as how much an update tends to change the probability of a possible result of Y. (A similar method was used by [Russell and Wefald 1991] to estimate the utility of a node expansion for game tree search.)

This update mechanism has the desirable property that the association of plans to their true maps is a fixed point (unaffected by further updates) when the maps of noncomputational actions are known, since all other maps are defined inductively in terms of those by the same means used to update them. It also allows for straightforward encoding of prior information possessed by the designer, in the form of initial map estimates for each plan.

Knowledge about the world

We have not yet considered how a planner might represent incomplete knowledge of the state of the world or the map implemented by an action. We also need a more compact representation for sets of world lines in order to efficiently represent plan maps. We can address these issues by coarse-graining the state space. Let us partition the state space into "coarse states", each having a probability distribution over its members. For example, a coarse state could leave one attribute unspecified, weighting each possible value by its relative probability. Every plan map has a coarse version, formed by averaging over the members of initial coarse states and coarse-graining the map's range. Note that updating by combination of subplan maps does not commute with coarse-graining: updating first provides a better estimate, by maintaining all intermediate states, at a higher computational cost. The planner might therefore choose to compute in the coarse-grained space first, using the results to better choose computations in the original space. Coarse-grained spaces thus function like the abstraction spaces of classical planning (described in [Knoblock 1991]).

Different coarse-grainings are useful in representing different portions of a planner's knowledge. An observation the planner makes in trying to determine the current world state might provide incomplete information, such as the value of only one attribute. The planner's knowledge would then best be represented as a coarse state. Similarly for the planner's knowledge of the state dependence of utility or action maps. In fact, if the planner must learn an action map from observations of the action's results under various initial conditions, generalization will be easier with a simpler state space representation. (Choice of map, and appropriate coarse-graining, can be done by testing the various hypotheses using Bayesian methods, as described in [MacKay 1992]. These methods include an "Occam's razor" bias towards simpler representations.) We can translate between different coarse-grainings by going through the original space. Thus, each state in one space corresponds to a probability distribution over states in the other.

To adapt an example from [Pearl 1988], consider a planner trying to decide whether or not to go home after receiving a telephone call from a neighbor reporting the sound of a burglar alarm. In calculating the consequences of such an action, the relevant state information is whether or not a burglary occurred. However, the observations only directly determine the state in a coarse space of attributes determined by the call. The planner must translate between these two representations in order to make a decision. (It may be that the appropriate translation distribution is not explicitly stored, and the translation must be done through intermediate representations, such as a specification of attributes of the neighbor relevant for determining the veracity of the call.) A Bayesian network (described in [Pearl 1988]) will efficiently represent the dependencies between states in the two spaces if they are sparse.

Coarse-graining can be controlled like other computational actions. Estimation of the map implemented by a coarse-graining action could be performed by clustering and gathering statistics as with update maps. An important factor in determining the effect of such an action is what information is lost in ignoring certain distinctions, which depends on what level of detail of representation the planner has applicable knowledge. Of course, relevance of lost distinctions to final plan choice and computational savings from reducing the representation will effect the estimated utility of any plan containing coarse-graining.

Concluding remarks

We have presented a planning paradigm which integrates the flexibility of decision theory with search procedures providing many of the efficiency properties of classical heuristic planning methods. Search is controlled using computed estimates of the effects various computational actions will have on the planner's knowledge. We have described some of the properties which emerge from this paradigm and some of the computations expected to be of use to a planner. The paradigm allows for graceful response to time pressure and naturally incorporates several types of abstraction. The inductive nature of the plan definition has guided construction of a computational system allowing for iterative improvement of the planner's knowledge about plans, as deemed useful by the planner.

This work can be viewed as an extension of the normative theory of action choice, provided by decision theory, to include constraints on the planner's ability to compute the consequences of its own knowledge (an approach discussed in [Hacking 1967]). We hope that taking a well-understood normative theory and incorporating the constraints facing any implementation will lead to a more natural model for planning than could otherwise be constructed.

References

[Breese 1990] Breese, J., "Construction of Belief and Decision Networks," to appear in *Computational Intelligence*.

[Christensen and Grove 1991] Christensen, J. and Grove, A., "A Formal Model for Classical Planning," *Proceedings IJCAI-91*.

[Drummond and Bresina 1990] Drummond M. and Bresina J., "Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction," *Proceedings AAAI-90*.

[Hacking 1967] Hacking, I., "Slightly More Realistic Personal Probability," *Philosophy of Science*, vol. 34, 1967.

[Knoblock 1991] Knoblock, C., "Search Reduction in Hierarchical Problem Solving," *Proceedings AAAI-91*.

[MacKay 1992] MacKay, D., "Bayesian Interpolation," *Neural Computation*, vol. 4, 1992.

[McCarthy 1968] McCarthy, J., "Programs with Common Sense," in: Minsky, M. (Ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968.

[Pearl 1988] Pearl, J., *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Mateo, CA, 1988.

[Russell and Wefald 1991] Russell, S. and Wefald, E., *Do The Right Thing*, MIT Press, Cambridge, MA, 1991.