An Easy “Hard Problem” for Decision-Theoretic Planning

John L. Pollock
Department of Philosophy
University of Arizona
Tucson, Arizona 85721
pollock@arizona.edu
http://www.u.arizona.edu/~pollock

Abstract

This paper presents a challenge problem for decision-theoretic planners. State-space planners reason globally, building a map of the parts of the world relevant to the planning problem, and then attempt to distill a plan out of the map. A planning problem is constructed that humans find trivial, but no state-space planner can solve. Existing POCL planners cannot solve the problem either, but for a less fundamental reason.

1. Introduction

Decision-theoretic planners can be divided roughly into state-space planners and POCL (partial-order causal-link) planners. State-space planners first build a world model and then distill a plan out of it. POCL planners try instead to build a plan from the bottom up, relying entirely upon local relationships. Classical POCL planning was once the only game in town, but its image has been tarnished by recent work on high-performance classical planners and the success of MDP (Markov decision process) planning in decision-theoretic contexts. As a result, most work on decision-theoretic planning has focused on state-space planners. There are only a few exceptions (Mahinur [21,22,23], Burdian [17] , C-Buridan [12], B-Prodigy [5]). The general sentiment seems to be that POCL planning is a dead-end, the future of planning lying with state-space planners. The purpose of this paper is to is to present a planning problem that human beings find easy but existing state-space planners are in principle incapable of solving. Existing POCL planners are also incapable of solving it, but for a less fundamental reason. I presume that if human beings find the problem easy, then a good planning algorithm ought to be able to solve it. So this constitutes a challenge problem for decision-theoretic planning.

2. FOMDP Planning

Most work on decision-theoretic planning has proceeded within the framework of MDP planning (see [7] for an excellent survey). The best understood MDP planners are *fully observable* MDP planners (FOMDP’s, introduced by Bellman [2]). FOMDP’s are state-space planners. They consider the space of all possible world-states, and the probability that an action performed in any world-state will lead to a transition to any other world-state, and then they construct *optimal policies*, which prescribe the optimal action to perform in each world state.

The “standard” approach to solving FOMDP’s is based upon Bellman’s *principle of optimality*

This work was supported by NSF grant no. IIS-0080888.

and linear programming [2], and in effect constructs an optimal policy by regressing backwards from the end of the plan. This approach chooses an optimal "*k*-steps to go" action for each world-state relative to a previously selected (k–1)-step optimal policy. The latter prescribes optimal (k–1)-steps-to-go actions, optimal (k–2)-steps-to-go actions, etc. The optimal 1-step-to-go actions (at a world-state) are simply those having maximal expected-values in that state. Relative to a choice of optimal 1-step-to-go actions, the optimal 2-steps-to-go actions (at a world-state) are those producing maximal expected-values when combined with the 1-step-to-go actions. And so on. An optimal *n*-step policy consists of the optimal *n*-steps-to-go actions, (n–1)-steps-to-go actions, ... , 1-step-to-go actions.

3. POMDP Planning

In executing a policy, an agent can only respond to those parts of the world-state that are known to it. FOMDP's assume that the world is fully observable, so that the choice of action can be made contingent on all features of the world simultaneously. Obviously, in most contexts full observability is an unrealistic assumption. This has led to the investigation of *partially observable* MDP's (POMDP's introduced by Aström [1]). Here it is assumed that the agent's knowledge can be represented as a probability distribution over the space of world-states. The probability distributions constitute *epistemic states*, actions lead to transitions with various probabilities from one epistemic state to another, and the task is to find an optimal policy in the space of epistemic states. The general idea is to apply solution techniques for FOMDP's to the space of epistemic states [29].

4. The Challenge Problem

A generally recognized problem for MDP's and POMDP's is that the size of the state-space grows exponentially with the number of features of the world that are taken into account. However, it does not seem to be appreciated how serious this problem is. Very simple problems that humans solve easily are completely intractable for MDP's and POMDP's. To illustrate the difficulty, consider a simple planning problem that generalizes Kushmerick, Hanks and Weld's [17] "slippery gripper" problem. We are presented with a table on which there are 300 numbered blocks, and a panel of correspondingly numbered buttons. Pushing a button activates a robot arm which attempts to pick up the corresponding block and remove it from the table. We get 100 dollars for each block that is removed. Pushing a button costs two dollars. The hitch is that some of the blocks are greasy. If a block is not greasy, pushing the button will result in its being removed from the table with probability 1.0, but if it is greasy the probability is only 0.1. The probability of any given block being greasy is 0.5. We are given 300 chances to either push a button or do nothing. In between, we are given the opportunity to look at the table, which costs one dollar. Looking will reveal what blocks are still on the table, but will not reveal directly whether a block is greasy. What should we do? Humans find this problem terribly easy. Everyone I have tried this upon has immediately produced the optimal plan: push each button once, and don't bother to look at the table. Note that the order makes no difference, so if plans are identified with linear sequences of actions then there are 300! optimal plans. This is approximately 10^{614}.

We can cast this as a 599 step POMDP. Odd-numbered steps consist of either pushing a button or performing the null action, and even-numbered-steps consist of either looking at the table or performing the null action. World-states are determined by which blocks are on the table (T_i) and which blocks are greasy (G_i). The actions available are *nil* (the null action), P_i (push button *i*), and L (look at the table). This cannot be cast as a FOMDP, because the agent cannot observe which blocks are greasy. But notice that even if it could, we would immediately encounter an overwhelming computational difficulty. The number of world states would be 2^{600}, which is approximately 10^{180}. To get an idea of what an immense number this is, note that it has been estimated that there are approximately 10^{78} elementary particles in the entire universe. An optimal policy specified an optimal action for each state, so there is no way that an agent could even *represent* an optimal policy in this state-space, much less find one.

The state-space gets larger when we move to POMDP's. In general, POMDP's will have infinite spaces of epistemic states corresponding to all possible probability distributions over the underlying state-space, however a reachability analysis can often produce a smaller state-space with just finitely many possible probability distributions. In the slippery blocks problem, it can be shown that in reachable epistemic states prob(T_i) can taken any value in the set {1.0, $.5 \cdot .9$, $.5 \cdot .9^2$, ... , $.5 \cdot .9^{300}$, 0} and prob(G_i) can take the values .5 and 1.0. Not all combinations of these values are possible, but the number of reachable epistemic states is greater than 301^{300}, which is approximately 10^{744}. So it is computationally impossible for an implemented system to even represent optimal policies in this POMDP, much less find them by linear programming.

5. Factored MDP's

This kind of difficulty has led to work on *factored* MDP's (see [7] for a general discussion of factored MDP's), and techniques for solving them. *Abstraction techniques* for solving factored MDP's observe that there are often differences between states that are not relevant to solving the problem.[1] It may be possible to cluster states together so that the computation of optimal *k*-steps-to-go actions is the same for all the states in any given cluster. In fact, in computing optimal *k*-steps-to-go actions for the slippery blocks problem, the only relevant difference is whether, for the different choices of *i*, prob(T_i) = 1. P_i is an optimal *k*-steps-to-go action in an epistemic state that will actually be reached by an optimal policy, relative to a terminal sequence of k–1 optimal actions, iff prob(T_i) = 1 and P_i is not a member of the terminal sequence. It is unclear whether algorithms for factoring MDP's will be able to figure this out, but suppose we have an algorithm that can. Unfortunately, this means that there is still one cluster for each subset of the buttons, and so there are 2^{300} clusters. Furthermore, linear programming algorithms will have to regress backwards through all the 300! orderings of button pushings and compare them in order to compute that any of them is optimal. It is computationally impossible for an implemented system to do this.

6. Parallel Decomposition

There is a different technique for solving large MDP's that seems initially like it might be helpful. This is *parallel decomposition* [7]. When the sources of value in an MDP can be split into independent subsources, with the overall value of a state being the sum of the values contributed by the subsources, it is sometimes possible to divide the MDP into MDP's for each subsource, find optimal policies for the parallel MDP's, and then merge them to obtain an optimal policy for the original MDP. The sources of value in the slippery blocks problem can be separated into the discrete values obtained by removing each individual block from the table, so it may seem that parallel decomposition might work. However, if we consider the problem of getting a single block off the table, the optimal policy is to push the corresponding button, look to see if the block is still there, and if it is then repeat the process until we either run out of turns or the block is removed. The optimal policy for the original problem involves no looks and no repeats, so

[1] This has been applied to FOMDP's in [6], [9], [11], and [14]. It was applied to the solution of POMDP's in [10] and [13].

there is no obvious way to construct it out of the optimal policies for these smaller problems.

There are certainly ad hoc decompositions that will solve the problem. For instance, give each block one button push and one chance to look at the table. Then the optimal policy is to push the button corresponding to that block and not look at the table. Combining these policies produces the optimal policy for the whole problem. However, a decomposition technique must be *sound*, in the sense that it will produce optimal policies for all problems to which it is applicable. There is no obvious general decomposition technique that is sound and will produce this decomposition. For example, why should we distribute the steps equally among the constituent goals? Obviously, there are problems for which that will not work.

My conclusion is that the slippery blocks problem is unsolvable using existing strategies for solving POMDP's. It is to be emphasized that this is actually an easy (for humans) problem, and it only becomes hard by casting it as a POMDP. It must be concluded that, at least for this particular problem, this is the wrong way to do decision-theoretic planning. I do not want to conclude that there is in principle no way of solving this problem by factoring or decomposing POMDP's. Boutilier, Dean, and Hanks [7] make a case for the view that classical goal-regression and POCL planning can be viewed as limiting cases of MDP planning, and I am going to suggest below that decision-theoretic generalizations of POCL planning may be able to solve this problem. So my claim will only be that we should take decision-theoretic POCL planning seriously, either as an alternative to POMDP planning or as a particularly important special case of it.

7. Decision-Theoretic Planning Based Upon High-Performance Classical Planners

The image of POCL planning in classical contexts was severely tarnished by the sucess of GRAPHPLAN [3.4] and BLACKBOX [15,16], which often outperform POCL planners by several orders of magnitude. There is some recent work on creating decision-theoretic planners based upon GRAPHPLAN and BLACKBOX, so let us see how they fare on the slippery blocks problem.

High performance satisfiability planners deriving from BLACKBOX convert the problem into a problem of finding assignments to propositional variables that satisfy a complex propositional formula. Majercik and Littman [18,19] have extended this to probabilistic planning with their planners C-MAXPLAN and ZANDER. These are not actually decision-theoretic planners, because they do not take account of values — just the probabilities of achieving goals, and they aim at constructing plans that maximize the probability of goal achievement. It seems likely that the same ideas can be extended to decision-theoretic planning, and I will think of these planners in that way. C-MAXPLAN encodes the planning problem into a propositional formula, and then computes the probability that a plan will achieve its goal by computing all satisfying assignments to the chance variables and summing them. However, there are as many satisfying assignments as there are reachable world-states, i.e., more than 301^{300}, so the problem cannot be solved in this way.

ZANDER uses a different propositional encoding and then computes all possible trees of variable assignments, choosing an optimal subtree. But again, there are as many satisfying assignments as reachable world-states, so this tree cannot actually be built and the problem cannot be solved in this way.

It is also worth noting that both C-MAXPLAN and ZANDER must compute and compare all "plausible candidates" for optimal plans in order to determine which are actually optimal. Sophisticated pruning algorithms may make it unnecessary to compare all possible plans, but even with maximally efficient pruning all of the 300! optimal plans would have to remain unpruned and be compared with each other to verify that none is surpassed by another. A real agent cannot compare 300! (i.e., 10^{614}) plans.

GRAPHPLAN is not really a state-space planner. It is more like a goal-regression planner supplemented with a reachability analysis. GRAPHPLAN gains its efficiency by first constructing the plan-graph (used by the reachability analysis) and then searching backwards through it by goal-regression. DT-GRAPHPLAN [27] applies the same idea to decision-theoretic planning. To do this it computes probabilities and utilities for each proposition in the plan-graph at each stage of its expansion, including multiple entries for the different possible probabilities of each proposition. The original version of DT-GRAPHPLAN assumed that propositions were independent, and so it could not handle the slippery blocks problem in which there is little independence. More recently [26], probabilistic dependence has been handled by maintaining a Bayesian net in parallel with the plan-graph that computes the non-independent probabilities. To see how this fares, let us modify the slippery blocks problem slightly by supposing that instead of knowing that the probability of a block being greasy is .5, what we know is that half the blocks are greasy, each having the same initial probability of being greasy. The point of the change is that the probability of a block being greasy then changes as other blocks are removed from the table. This change does not affect what plans are optimal. It is not clear how to cast this problem as a Bayesian net. Bayesian nets have to be acyclic, but if we include nodes for the greasiness of the blocks, acyclicality fails. The probability of a block being on the table after the corresponding button is pushed is influenced by whether it is greasy, and the probability of its being greasy given that the button is pushed is influenced by whether it is still on the table. If we do not include nodes for the greasiness of the blocks, then the nodes just concern which blocks are on the table at each stage and which buttons have been pushed. However, as noted above, the probability of a block being greasy is influenced by what other blocks are on the table, and that in turn affects the probability that the block will still be on the table after its button is pushed. Thus the Bayesian net must encode as a primitive probability every probability of the form prob(T_i/P_i & $\prod_{j\in K}T_j$) where K is a set of block numbers and $i\notin K$. There are 2^{300} such probabilities, so this Bayesian net cannot actually be built or encoded in a real agent.

There are principled reasons why state-space planners cannot solve this problem. They begin by constructing a complete map of the relevant parts of the world, and then cut the solution to the planning problem out of it. Their reasoning is, in an important sense, global, and the difficulties they encounter arise from the fact that “global” is often just too big. The slippery blocks problem was explicitly designed to foil state-space planners. But, in fact, it is still a toy problem. The real world is a big place. Unless we carefully massage them (and in fact, oversimplify considerably), real-world planning problems will involve much larger cardinalities than the slippery blocks problem. It is computationally impossible to solve such problems by global reasoning.

8. Decision-Theoretic POCL Planners

By contrast, POCL planners only engage in local reasoning. Decision-theoretic POCL planners are generalizations of classical POCL planners [20,25]. They replace deterministic causal links by probabilistic ones. I believe that the only decision-theoretic POCL planner that has actually been constructed is Mahinur [21,22,23], although there are also probabilistic POCL planners like Buridan [17], C-Buridan [12], and B-Prodigy [5] that could be modified to produce decision-theoretic POCL planners. POCL planners first construct a crude plan on the basis of simple local relationships between actions and goals. Then they search for other local relationships that give rise to destructive interference. Then they search for still further local relationships that will enable the planner to refine the plan to avoid the destructive interference. Finally, they search for additional local relationships that may make it possible to improve the plan, generating a plan with a higher expected-value. In sum, POCL planners are *refinement planners* that start with a crude plan and then make it better. All of the considerations underlying both the discovery of the crude plan and its subsequent refinement are local considerations, and the search for refinements is driven by the *particular plan* that is being refined. Notice that, by way of contrast, the use of reachability analyses and factoring in state-space planners is driven by the *planning problem* rather than by attempts to refine a single candidate plan, and that is much less effective because it must take account of all possible solutions to the planning problem. So POCL planners appear to avoid the kind of theoretical difficulties that beset state-space planners. As we will see, there is a simple reason why no existing POCL planners can solve the slippery blocks problem, but there seems to be no reason in principle why they might not.

How might a POCL planner solve the slippery blocks problem? The sources of value in this problem consist of getting the individual blocks off the table. A POCL planner can begin by constructing a plan for getting each block off the table individually, e.g., push the corresponding button. It is probably important that humans do this reasoning just once, for an *unspecified* block, whereas existing POCL planners must do it individually for each of the 300 blocks. This is something that should be explored as a way of improving POCL planners, possibly dramatically, but I will leave it aside for now. POCL planners then merge these individual subplans into the overall plan of pushing each button once. Then they search for and try to repair flaws in the resulting plan. Classically, this is threat detection and threat resolution, but in decision-theoretic contexts the flaws can be of more general sorts. The general idea behind POCL decision-theoretic planning is that the planner makes the defeasible assumption that the expected-value of the overall plan is the sum of the expected-values of the subplans, and then it must search for destructive interactions between different parts of the plan that make that assumption false. If such interactions are detected, an attempt is made to refine the plan to avoid or minimize the interactions. In the slippery blocks problem, there are no destructive interactions, and a POCL planner should be able to determine that quickly.

However, just finding the plan that is in fact optimal is not sufficient to solve the planning problem. The planner must also verify that the plan is a solution to the problem. Different planners understand the planning problem in different ways. POMDP planners usually take their task to be that of finding an optimal plan. But decision-theoretic POCL planners like Mahinur [21,22,23], take their task to be that of finding a plan whose expected-value comes up to a threshold set by the user. If we understand the planning problem in the latter way, and we set the threshold to be the value that is in fact the expected-value of the optimal plan, then merely finding the optimal plan suffices to solve the problem. This way of understanding planning problems is based upon Herbert Simon's notion of *satisficing* [28]. However, it does not seem to me to be a satisfactory way of understanding planning problems. If we set the threshold too high (higher than the value of the optimal plan), no plan can be found, and if we set it too low the planner will be content with finding plans that are gratuitously inferior to the optimal plan. In effect, we must solve the problem of finding an optimal plan before we know how to set the threshold. And notice that humans have no difficulty recognizing that the plan they produce for the slippery blocks problem is optimal, so a planning algorithm ought to be able to do that as well.

If the objective is to find an optimal plan, the POCL planner cannot stop with just finding the plan that is in fact optimal. It must verify that it is actually optimal. Notice that for a POCL planner there are not 300! optimal plans — there is just one, with the steps unordered with respect to each other. POCL planners only order the steps if there is some reason to do so, and in this case there is none. This gives them a huge advantage over those state-space planners that insist on comparing all plausible linear plans. However, even though there is only one optimal plan, the details of verifying that it is optimal remain a bit murky. A POCL planner *might* be able to reason that changing any of the button-pushing steps in its plan would result in not pushing one of the buttons (and possibly pushing another one more than once), which would lower the expected-value of the plan; and looking instead of not looking would raise the costs without raising the expected payoff, and so would also lower the expected-value. In this way, a POCL planner *might* be able to solve the slippery blocks problem quickly and efficiently. But as remarked, the details are murky because no existing decision-theoretic POCL planners even try to find optimal plans. If we set the threshold right, a planner like Mahinur [21,22,23] might well return the desired plan when applied to the slippery blocks problem, but this does not establish that the plan is optimal.

These considerations at least suggest that decision-theoretic POCL planning should be taken seriously. There is, however, a well-recognized problem for classical POCL planners. They cannot solve very big problems. Satisfiability-based planners seriously outperform them. But now let us get really speculative. There may be room for hope. In a classical planning environment, such planners are trying to solve difficult search problems by brute force. It is very difficult to see how the search of a *general-purpose* classical POCL planner could be made more efficient (except by building in ad hoc heuristics for specific domains). Viewed as a brute-force search problem, decision-theoretic planning is harder than classical planning. However, decision-theoretic planners also have much more knowledge at their disposal that may be useful in directing plan search in intelligent ways. This is knowledge about probabilities and utilities. This may, in the end, enable decision-theoretic POCL contingency planners to solve real-world problems. The performance of such a planner will be dramatically influenced by the way it uses probabilities and utilities to direct its search. This is nicely illustrated by some recent work by Onder and Pollack [21,22]. By comparison, classical planners plan while wearing blinders, and can do little to improve performance except for employing faster search algorithms. So I do not think we can rule out the possibility that sophisticated decision-theoretic POCL planners may eventually be able to solve problems in truly complex environments.

On problems sufficiently simple to be amenable to solution by state-space planners, POCL planning may be orders of magnitude slower. But on problems of real-world complexity, POCL planning may still be the only game in town. Whether POCL planners can actually solve such problems remains to be seen, but at least they will not be subject to the same kinds of difficulties as those encountered by state-space planners.

9. Decision-Theoretic POCL Planners

I have presented a challenge problem for decision-theoretic planning. For principled reasons, existing planners cannot solve it. Very fundamental cardinality problems make it intractible for state-space planners. Existing POCL planners do not even try to find optimal plans, so they do not even try to solve the problem. Humans, on the other hand, find it easy. A challenge for the planning community is to find a planning algorithm that does as well as human beings on this problem.

Bibliography

[1] Aström, K. J., "Optimal control of Markov decision processes with incomplete state estimation", *J. Math. Anal. Appl.* **10** (1965), 174-205.

[2] Bellman, R., *Dynamic Programming*. Princeton University Press, Princeton, N.J., 1957.

[3] Blum and Furst, "Fast planning through planning graph analysis". In *Proceedings of the Fourteenth International Joint Conference on AI,*1995, 1636-42.

[4] Blum and Furst, "Fast planning through planning graph analysis", *Artificial Intelligence* **90** (1997), 281-300.

[5] Blythe, Jim, and Manuela Veloso, "Analogical replay for efficient conditional planning", *AAAI97.*

[6] Boutelier, Craig, "Correlated action effects in decision-theoretic regression", *UAI97*, 30-37.

[7] Boutelier, Craig, Thomas Dean, and Steve Hanks, "Decision theoretic planning: structural assumptions and computational leverage", *Journal of AI Research* **11** (1999), 1-94.

[8] Boutelier, Craig, and R. Dearden, “Approximating value trees in structured dynamic programming”, *Proceedings of the Thirteenth International Conference on Machine Learning* (1996), 54-62.

[9] Boutelier, Craig, R. Dearden, and M. Goldszmidt, “Exploiting structure in policy construction”, *IJCAI95*, 1104-1111.

[10] Boutelier, Craig, and David Poole, “Computing optimal policies for partially observable decision processes using compact representations”, *AAAI96*, 1168-1175.

[11] Diettrich, T. G., and N. S. Flann, “Explanation-based learning and reinforcement learning: A unified approach”, *Proceedings of the Twelfth International Conference on Machine Learning* (1995), 176-184.

[12] Draper, Denise, Steve Hanks, and Daniel Weld, “Probabilistic planning with information gathering and contingent execution”, *Proceedings of AIPS94*.

[13] Hansen, E. A., and Z. Feng, “Dynamic programming for factored POMDP’s”, *AIPS2000 Workshop on Decision-Theoretic Planning*, 41-48.

[14] Hoey, J., R. St-Aubin, and C. Boutilier, “SPUDD: Stochastic planning using decision diagrams”, *UAI99*.

[15] Kautz, H. A., and B. Selman, “Pushing the envelope: planning, propositional logic, and stochastic search”, *Proceedings of AAAI-96*, 1194-1201.

[16] Kautz, H. A., and B. Selman, “Blackbox: a new approach to the application of theorem proving to problem solving”, in *AIPS98 Workshop on Planning as Combinatorial Search*, 58-60.

[17] Kushmerick, Hanks and Weld, “An algorithm for probabilistic planning”. *Artificial Intelligence* **76** (1995), 239-286.

[18] Majercik and Littman, “MAXPLAN: A new approach to probabilistic planning”, *AIPS98*, 86-93.

[19] Majercik and Littman, “Contingent planning under uncertainty via stochastic satisfiability”, *AAAI99*.

[20] McAllester, David, and Rosenblitt, David, “Systematic nonlinear planning”, *Proceedings of AAAI-91*, 634-639.

[21] Onder, Niluger, and Martha Pollack, “Contingency selection in plan generation”, *ECP97*.

[22] Onder, Niluger, and Martha Pollack, “Conditional, probabilistic planning: a unifying algorithm and effective search control mechanisms”, AAAI 99.

[23] Onder, Niluger, Martha Pollack, and John Horty, “A unifying algorithm for conditional, probabilistic planning”, *AIPS1998 Workshop on Integrating Planning, Scheduling, and Execution in Dynamic and Uncertain Environments*.

[24] Pednault, Edwin P., Toward a Mathematical Theory of Plan Synthesis, PhD dissertation,