Beyond Plan Length: Heuristic Search Planning for Maximum Reward Problems.

Jason Farquhar and Chris Harris

Image Speech and Intelligent Systems
Department of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
[jdrf99r;cjh]@ecs.soton.ac.uk

Abstract. Recently automatic extraction of heuristic estimates has been shown to be extremely fruitful when applied to classical planning domains. We present a simple extension to the heuristic extraction process from the well-known HSP system that allows us to apply it to reward maximisation problems. This extension involves computing an estimate of the maximal reward obtainable from a given state by ignoring delete lists. We also describe how to improve the accuracy of this estimate using any available mutual exclusion information. In this way we seek to apply recent advances in classical planning to a broader range of problems.

Keywords: Domain Independent Planning, Reward Based Planning, Heuristic Search Planners.

1 Introduction

In this paper we investigate reward maximisation as an alternative to plan length for the optimisation criteria in STRIPS style problems. In reward maximisation problems we attempt to maximise the total reward obtained from the states visited and actions taken during plan execution, where the reward is an aribitary real-valued function over states and actions. In particular we focus on reward problems where the planners objectives are specified through the rewards allocated to different world states rather than as an explicit goal which *must* be achieved.

Inspired by the success of heuristic search in efficiently solving goal-based STRIPS problems (Bac00; BG99; HN01) we suggest that similar methods may be used in reward maximisation problems. To investigate this idea we present a modification of the heuristic used in HSP which is applicable in the reward maximisation case.

This paper is organised as follows. Section 2 presents a mathematical model of STRIPS problems and its reward based extensions. Section 3 gives the derivation of our new heuristic and an outline of the algorithm used to calculate it. The final sections discuss future work (Sec 4), compare related work (Sec 5) and present conclusions (Sec 6).

2 Reward Based Planning

Following (BG99) we represent a conventional STRIPS (FN71) domain as a tuple $D = \langle A, O \rangle$, where A is a set of atoms and O is a set of operators. The operators $op \in O$ and atoms $a \in A$ are all assumed ground (all variables replaced by constants). Each operator has precondition, add and delete lists which we denote as Pre(op), Add(op) and Del(op) respectively, given by sets of atoms from A.

Such a domain can be seen as representing a state space where:

1. the states $s \in S$ are finite sets of atoms from A
2. the state transition function $f(s, op)$ which maps from states to states is given by:

$$s' = f(s, op) = \begin{cases} s \cup \mathrm{Add}(op) \setminus \mathrm{Del}(op) & \text{if } \mathrm{Pre}(op) \subseteq s \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{1}$$

3. the result of applying a sequence of operators is defined recursively as

$$s_n = f(s_0, \langle op_1, .., op_n \rangle) = f(f(s_0, \langle op_1, ..., op_{n-1} \rangle), op_n) \tag{2}$$

In reward based planning the domain description is augmented to give $\mathcal{P} = \langle A, O, I, R \rangle$ where $I \subset A$ represents the initial situation and R is the reward function which maps from situations to real valued rewards. R consists of two components, a state based component, $R : s \mapsto \mathbb{R}$, and an operator based component, $R : op \mapsto \mathbb{R}$. The solution to a reward based planning problem is a sequence of operators $P = \langle op_1, ..., op_n \rangle$ that *maximises* the total reward received from the states visited and operators applied during plan execution.

One particular problem with reward based planning not found in goal based planning is the possibility of cyclic solutions with infinite reward. Such infinities are difficult to work with so it is usual to modify the optimisation criteria to remove them; for example, by discounting future rewards (Put94), optimising with respect to reward rate, or optimising with respect to a finite planning horizon. A finite horizon is used in this work though the method could be applied in the other cases .

3 Heuristics for Reward Based Planning Problems

Heuristic search planners use a heuristic function $h(s)$ to guide solution search in state space. To develop an effective heuristic we use the same trick that proved so effective in STRIPS planning (BG99; HN01), i.e. we solve a relaxed problem ignoring operator delete lists and use this solution as a heuristic estimate in the original problem. Unfortunately solving even the relaxed problem can be shown to be NP-hard (BG99) so an approximate solution must be used.

In STRIPS problems one of the most successful approximations is that used in the HSP system developed by Bonet and Genffer (BG99). This decomposes the problem of finding the shortest path to the goal state into one of finding the shortest path to each individual atom from which an estimate of the goal distance is reconstructed. This decomposition and reconstruction is performed because the number of atoms, $|A|$ is generally *much* smaller than the number of states, i.e. subsets of atoms, $|S|$, (which is exponential in the number of atoms, $|S| \leq |2^{|A|}|$). Hence performing computations in atom space can be significantly cheaper in time and space than the equivalent computations in state space.

For reward based problems we propose to use a modifed version of the same approximation technique of decomposing and resconstructing state values from atom values. We begin by defining the *value*, $V(s, t)$, of state s as the maximal reward obtainable in getting from the initial state I to this state in t steps. This value could be calculated directly in state space using the forward Bellman Equation:

$$V(s,t) = R(s) + \max_{op \in O} \left[R(op) + V(f^{-1}(s,op), t-1) \right] \tag{3}$$

where $V(s,t)$ is the value of the state s at time step t, $R(s)$ and $R(op)$ are the rewards for being in state s and performing operation op respectively, $f^{-1}(s,op)$ is the inverse state transition operator which returns the state s' from which application of operator op results in the new state s, and $V(I,0) = 0$.

The problem defined by (3) is equivalent to finding a maximal weight path through a weighted directed graph. The nodes represent states, edges the operators, and the edge and node weights the operator and state rewards respectively. A number of efficient algorithms which are polynomial in $|S|$ can be used to solve this problem. Unfortunately as mentioned above, $|S|$ is generally exponential in the number of atoms making even these algorithms costly. Hence we approximate (3) by re-formulating the problem to apply over the smaller space of atoms. This gives the equations (4) and (5).

$$V(p,t) = \max_{p \in \mathrm{Pre}(r)} R(r,t) + \max_{p \in \mathrm{Eff}(op)} \left(R(op) + V(\{\mathrm{Pre}(op)\}, t-1) \right) \tag{4}$$

$$V(p,0) = \begin{cases} 0 & \text{if } p \in I \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5}$$

where $V(\mathrm{Pre}(op), t)$ is the reward for for being in atom set $\{p : p \in \mathrm{Pre}(op)\}$ at time step t, and $\mathrm{Pre}(r)$ is the set of atoms which define reward state r. $R(r,t)$ is the reward obtained from being in reward state r at time t and is equal to the value of the reward state $R(r)$ if all the atoms in r are valid at time t and undefined otherwise.

Equation (4) defines the estimated value of an atom at time step t is the sum of the immediate reward received due to the current state, $R(r,t)$, and the propagated total reward of the maximum reward path to this atom from the initial state. Equation (5) sets the initial value of the atoms.

The accuracy of the function, $V(\{B\}, t)$, used to estimate value of an atom set, B, from the values of its consistituent atoms, $p \subset B$, is critical to the accuracy of the heuristic and hence performance of the planner. In (BG99) Bonnet and Genffer suggest using either the sum or maximum of the atom values. Using the sum pessimistically assumes that each atom is totally independent, hence the shortest path to the set becomes the sum of the best paths to each atom in the set. Using the maximum optimistically assumes that the atoms are totally dependent such that any path which achieves one atom will also achieve all other atoms which can be reached with shorter paths. Hence the shortest path to an atom set becomes the length of the shortest path to the last atom achieved.

In the reward maximisation case things become a little more complex. If independence is assumed then the atom set can only be valid when t is greater than the sum of the initial values for each atom in the set. If total dependence is assumed then the value of the set becomes the value of the highest reward path which could achieve the set, i.e. the last atom achieved initially and after that the highest reward path longer than the sets initially valid length. In this case, which is used in this paper, we obtain Equation (6).

$$V(B,t) = \begin{cases} \max\limits_{a \in B, l \leq \text{len}(a) \leq t} V(a,t) & \text{if } \forall a \in B,\ V(a,t) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{6}$$

where B is the set of atoms, $\text{len}(a)$ is the number of operations required to obtain atom a's value $V(a,t)$ and l is the number of operations required to first make B valid.

Solving the equations (4,6,5) also corresponds to finding the maximum weight path in a graph, the *connectivity graph* (HN01). In this graph atoms are nodes and operators/rewards are high order edges which only become available after a certain subset of the nodes (their preconditions) are valid.

Computation of the atom values can be done in polynomial time using a GraphPlan like forward propagation procedure based upon the Connectivity Graph. Briefly, the algorithm proceeds in a time step by time step manner propagating tokens from atoms to operators and rewards. The operators/rewards are then identified as available for propagating reward and atom validity to their effects when all their pre-conditions are valid. The set of valid atoms is then used to compute the updated value for all the atoms using equations (4,6,5).

Using the value function computed in this way, the heuristic value of the state s in the original problem is defined as the maximal value of a valid atom in the final layer of the relaxed graph, Eqn (7).

$$h(s,t) \overset{def}{=} \max_{p \in P^t} V(p,t) \tag{7}$$

where t is the maximum time step to which the relaxed solution has been computed and P^t is the set of atoms valid, i.e. with defined values, at time t.

3.1 Improving the Estimate Using Mutual Exclusion Information

One problem with the heuristic estimates produced by the above procedure is that it takes no account of the negative interactions between atoms. This is caused by ignoring the operators delete lists allowing extra paths to states or atoms being possible in the relaxed problem which do not exist in the original problem, making states become valid earlier or with higher reward. This is the same problem of ignored negative interactions examined in (NNK00; NK00) and can be addressed in a similar way using any available mutual exclusion information.

Mutual exclusions, called mutexs from now on, are used extensively in the Graphplan algorithm (BF97) and represent pairs of atoms that cannot occur together at some depth in any plan. In the heuristic value computation this information can be used to close off some of the extra paths by preventing or delay operators/rewards from becoming valid until their pre-conditions are all non mutex. For negligible cost this information can be used in the value computation algorithm by annotating each operator/reward by the first plan depth at which it is applicable, i.e. all its pre-conditions are non mutex. Then the operators/reward are restricted to only be available after this depth. The technique can be used both for *static* mutexs which hold for all states and plan lengths and *dynamic* mutexs which hold only up to a certain plan length.

The cost of computing the additional mutexs can be controlled by only calculating the static mutex's once at the start of problem solving, for example by running GraphPlans full graph construction algorithm to *level-off* from the initial state. Any mutex's which hold at *level-off* this point are static. Additional dynamic mutex's and then be computed to any required degree of accuracy only when necessary.

4 Further Work

A prototype reward based planner has been implemented using the above heuristic evaluation function both with and without the mutal exclusion enhanchements. Initial results appear to validate the system with the heuristic values showing good correlation to the true value of a state. We are currently in the process of developing an A* search engine a full test suite for the planner. We then intend to perform comparisons with other planners in both conventional STRIPS and reward based domains.

5 Related Work

There are obviously rich connections between this work and existing work on Graphplan (BF97) and the heuristic state space search planners (BG99; HN01) upon which it is based. The idea of using mutual exclusion information to account for negative interactions and hence improve the quality of the heuristic estimates is similar to that used in (NNK00) to improve the heuristic estimates in regression search.

This work is also closely related to work on adding probabilistic (BL98) and decision theoretic (PC00) abilities to the Graphplan algorithm. These systems rely on propagating additional probability information through the planning graph in much the same way that rewards are propagated in this work. This work also has significant connections to work on decision theoretic planning, where reward based formulations are also used. Traditionally these systems have used dynamic programming over a graphical representation of state space to find optimal solutions (Put94). As discussed above in Sec 2 and in (BDH99) this works well for reasonable state space sizes but tends to become infeasible for very large state spaces. Use of heuristic search to address such large problems has recently been proposed by Bonet and Geffner (BG00).

6 Conclusions

A method for extending the techniques of heuristic planning, as used in the well known HSP system, to the more expressive language of reward based planning was presented. The development of a domain independent heuristic for reward maximisation problems forms the crux of our work. This heuristic is based upon computing an estimate for the maximal reward obtainable in a relaxed problem where delete lists are ignored. We have shown how our heuristic function was developed to cope with reward accumulation and goalless planning problems. We have also demonstrated how this heuristic estimate can be improved by using any available mutual exclusion information to take some account of negative interactions.

References

[Bac00] F. Bacchus. Results of the aips 2000 planning competition, 2000. URL: http://www.cs.tronto.edu/aips-2000.

[BDH99] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence*, 11:1–94, 1999.

[BF97] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[BG99] B. Bonet and H. Geffner. Planning as heuristic search: New results. In *Proceedings of ECP-99*. Springer, 1999.

[BG00] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc AAAI-2000*, 2000.

[BL98] Avrim L. Blum and John C. Langford. Probabilistic planning in the graphplan framework. Technical report, Carnegie Mellon University, School of Computer Science, CMU, Pittsburgh, PA 15213-3891, 1998.

[FN71] R. E. Fikes and N. J. Nilsson. Strips: a new approach to the application of theorm proving to problem solving. *Artificial Intelligence*, 2(3-4):189–203, 1971.

[HN01] Jorg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[NK00] XuanLong Nguyen and Subbarao Kambhampati. Extracting effective and admissible state space search heuristics for the planning graph. Technical report, Arizona State University, Dept. of Computer Science and Engineering, Tempe, AZ 85287-5406, 2000.

[NNK00] R.S. Nigenda, X. Nguyen, and S. Kambhampati. Altalt: Combining the advantages of graphplan and heuristic state space search. Technical report, Arizona State University, Dept. of Computer Science and Engineering, Tempe, AZ 85287-5406, 2000.

[PC00] G. Peterson and D. J. Cook. Decision-theoretic planning in the graphplan framework. In *Proc AAAI-2000*, 2000.

[Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.