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, UK

Abstract

Automatic extraction of heuristic estimates has been extremely fruitful in classical planning domains. We present a simple extension to the heuristic extraction process from the well-known HSP and FF systems which allow us to apply them to reward maximisation problems. These extensions involve computing an estimate of the maximal reward obtainable from a given state when ignoring delete lists, which are then used to guide the backward search in the FF system. The heuristics are evaluated in a simple robotic delivery task and shown to be effective in reducing the number of nodes evaluated. 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.

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 arbitrary 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) we suggest that similar methods may be used in reward maximisation problems. To investigate this idea we present a modification of the heuristics used in HSP (BG99) and FF (HN01) which are applicable in the reward maximisation case.

This paper is organised as follows. Section presents a mathematical model of STRIPS problems and its reward based extensions. Section gives the derivation of our new heuristics and an outline of the algorithms used to calculate them. The final sections include experimental results discussion for possible further work and present out conclusions.

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 $o \in O$ and atoms $a \in A$ are all assumed ground (all variables replaced by constants). Each operator, o, has precondition, add and delete lists which we denote as Pre(o), Add(o) and Del(o) respectively, given by sets of atoms from A. This can be seen as representing a state space where:

1. states $s \in S$ are finite sets of atoms from A, i.e. $s \subset A$.
2. the state transition function $f(s, o)$ maps from states and actions to states such that:

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

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

$$s_n = f(s_0, \langle o_1, .., o_n\rangle) = f(f(s_0, \langle o_1, ..., o_{n-1}\rangle), o_n) \tag{2}$$

In reward based planning the domain description is augmented to give $\mathcal{P} = \langle A, O, s_0, R\rangle$ where $s_0 \in S$ 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. For simplicity a finite horizon is used in this work.

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 approximation must be used.

The $h^{\max}$ heuristic

In STRIPS problems one of the most successful approximations is that used in the HSP system developed by Bonnet and Gruffer (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 modified version of the same approximation technique of decomposing and reconstructing 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 s_0 to this state in t steps. This value could be calculated directly in state space using the forward Bellman Equation:

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

where $V(s,t)$ is the value of the state s at time step t, $R(s)$ and $R(o)$ are the rewards for being in state s and performing operation o respectively, $f^{-1}(s,o)$ is the inverse state transition operator which returns the state s' from which application of operator o 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 intractable. Hence we approximate (3) by re-formulating the problem to apply over the space of atoms, giving;

$$V(p,t) = \max_{p \in \text{Pre}(r)} R(r,t) + \max_{p \in \text{Eff}(o)} [R(o) + V(\{\text{Pre}(o)\}, t\text{-}1)] \quad (4)$$

where $V(\text{Pre}(o), t)$ is the reward for for being in atom set $\{p : p \in \text{Pre}(o)\}$ at time step t, and $\text{Pre}(r)$ 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 as 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. The base case of the recursion is given by the initial state, s_0, where $V(p,0) = 0$ if $p \in s_0$ and is undefined otherwise.

The accuracy of the function, $V(\{B\},t)$, used to estimate value of an atom set, B, from the values of its constituent atoms, $p \in B$, is critical to the accuracy of the heuristic and hence performance of the planner. In (BG99) Bonnet and Geffner suggest using either the sum or maximum of the atom values. The sum assumes atoms are totally independent such that to achieve a set of atoms all its members must be achieved independently. The maximum assumes 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.

In reward problems using the sum can give both pessimistic and optimistic estimates, when atom values are negative or positive respectively, leading to an uninformed estimate. To avoid this problem, in this paper we assume atoms are totally dependent, so the value of a set of atoms 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. Hence, we obtain Equation (5).

$$V(B,t) = \begin{cases} \max\limits_{\substack{a \in B \\ l \leq \text{len}(a) \leq t}} V(a,t) & \text{if } \forall a \in B, V(a,t) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases} \quad (5)$$

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) and (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 preconditions are valid. The set of valid atoms is then used to compute the updated value for all the atoms using equations (4,5).

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, $t_{\max}$, of the relaxed graph, Eqn (6).

$$h^{\max}(s, t_{\max}) \overset{def}{=} \max_{p : V(p, t_{\max}) \text{ is defined}} V(p, t_{\max}) \quad (6)$$

Algorithm	Maximum Plan Length								
	20			35			40		
	$\|N\|$	$t(s)$	Rew	$\|N\|$	$t(s)$	Rew	$\|N\|$	$t(s)$	Rew
Dynamic Programming	2083	0.19	8	-	>5min	-	-	>5min	-
A* - $h(s) = 0$	330	0.16	8	1935	0.3	20	16932	2.03	30
A* - $h^{\max}(s)$	105	0.38	6	238	1.1	16	696	2.7	28
A* - $h^{\text{ff}}(s)$	96	0.27	7	168	0.8	20	289	1.3	30

Figure 1: Comparison of heuristic performance for different plan horizon lengths on a simple robot delivery task. (Timings performed on a PIII 500MHz.)

The h^{ff} heuristic

In the $h^{\max}$ heuristic the value of all but the single best path needed to achieve the atoms in an atom set are ignored, due to the assumption of total atom dependence. Therefore, the heuristic estimate could be improved if the reward from paths ignored when this assumption does not hold could be re-introduced.

In the FF (HN01) system this same problem is addressed by performing a greedy backward search for actions to support all the atoms in a set. Exactly the same technique can be used in reward problems except in this case the $h^{\max}$ heuristic is used for search guidance and the search starts from the maximum value atom at $t_{\max}$. The result of this procedure is a set of states, S_t, and actions, A_t, to be applied at each time step, t, to achieve the goal. Given these sets the revised heuristic value is given by;

$$h^{\text{ff}}(s, t_{\max}) \overset{def}{=} \sum_{t=0,\ldots,t_{\max}} \left[\sum_{a_i \in A_t} R(a_i) + \sum_{s_i \in S_t} R(s_i) \right]. \tag{7}$$

Experimental Results

The above heuristics have been implemented in C++ based upon a modified GraphPlan style planner. To evaluate their effectiveness they were used in an A* search procedure and compared with dynamic programming and uninformed A* on a simple reward based planning problem[1] for different plan horizon lengths. The results of these runs are given in Fig 1.

Fig 1 clearly shows the benefit of forward search in avoiding the worst execesses of exponential blowup in run-time with plan horizon length associated with dynamic programming - at the expense of possibly sub-optimal solutions. It is also clear that the heuristics do result in a significantly more informed A* search, reducing the number of nodes evaluated by at least a factor of three. However this information comes at significant computational cost with the heuristic computation reducing the node expansion rate by a factor of 20. This means that for all but the largest problems, i.e. longest horizon length, the net computational benefit of the heuristics is minimal. Also, as expected, the h^{ff} heuristic appears to be more informed than the $h^{\max}$ heuristic.

[1]This problem was a simple delivery problem on an open 9x9 grid with 3 packages to deliver for rewards of 25,20 and 15 and 9 special locations which gave reward of 1 when entered. All actions had reward of -1.

Further Work

As the experiments pointed out the most pressing need in this work is to reduce the computational cost of using the heuristics. This could be achieved by either reducing the computational cost of heuristic evaluation or extracting additional search control information from the heuristic (as in done in FF (HN01) to identify a set of *helpful actions*). Another avenue we intend to investigate is extending the heuristics to a Decision Theoretic problems (Put94) by including probabilistic actions and propagating utilities rather than rewards, as was done in (BL98) and (PC00) for the Graphplan algorithm.

Conclusions

A method for extending the techniques of heuristic planning, as used in the well known HSP and FF systems, to the more expressive language of reward based planning was presented. The development of two domain independent heuristics for reward maximisation problems forms the crux of our work. These heuristics are based upon computing an estimate for the maximal reward obtainable in a relaxed problem where delete lists are ignored. The first heuristic, $h^{\max}$, was developed by modifying HSP's max-atom heuristic to cope with reward accumulation and goal less planning. The second heuristic, h^{ff}, was developed to improve the first by re-introducing some ignored problem aspects. Experimental analysis demonstrates the informedness of the heuristics and their high computational cost.

References

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

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

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

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.

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

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

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

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