LDFS with Deterministic Plan Based Subgoals

Rajesh Kalyanam and **Robert Givan**
Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47907
{*rkalyana, givan*}*@purdue.edu*

Abstract

This paper describes a framework that leverages any probabilistic planning problem determinization method, any distance heuristic computation method and any search based probabilistic planner to produce a stronger probabilistic planner. As the size of a planning problem increases, explicit state space probabilistic planners are often unable to cope with the large state spaces typical of such problems. We propose a method that makes use of the subgoals generated from a determinization of the probabilistic planning problem to divide the planning problem into smaller manageable problems. The original probabilistic planner can then be applied on these individual problems resulting in a modified planner than can handle larger state spaces.

Introduction

Search-based probabilistic planners perform a form of value iteration that avoids visiting the whole state space by making use of search techniques and only considering the states reachable from the start state. The search heuristic functions both as an initial estimate of the value function and a guide for the search process towards states with Bellman errors. The values of such states are then repeatedly updated until they converge to their optimal values. The quality of this heuristic plays a major role in the speed of convergence of the planner. In very large state spaces even a fairly accurate heuristic is not sufficient for exploring the state space in a manner required for the process to converge. A common class of techniques that make the state space more manageable involve hierarchical or temporal abstractions where the larger problem is broken into several manageable subproblems. These techniques fall under the general class of *"divide-and-conquer"* algorithms and we employ a similar approach in our planner framework.

We make use of a determinization of a probabilistic planning problem to break down the original problem into a sequence of subproblems each identified by a start state and terminal state set. We then run any existing search based probabilistic planner on each of these subproblems to extract a policy for this subproblem. The sequence of policies from these subproblems forms the policy for the original problem. Our framework is generic enough to allow the use of any determinization method, deterministic planner and search based probabilistic planner; however for the purpose of this competition entry we make use of the FF-replan style *"all-outcomes"* determinization (Yoon and Givan 2007), the FF deterministic planner (Hoffmann and Nebel 2001) and the Learning Depth First Search probabilistic planner (Bonet and Geffner 2006). We describe these in more detail in the appendix. The following sections describe the generation of subgoals which are used to identify the sequence of subproblems and the details of our implementation.

Deterministic Subgoal Generation

Given a determinization of a probabilistic planning problem and a heuristic computation for the distance to the goal on this determinized state space, we can generate a deterministic plan from the initial state to the goal using any heuristic search based deterministic planner. This plan is a sequence of states : $(s_0, s_1, s_2, ...s_{n-1}, s_n)$, where s_0 is the start state of the problem and s_n is the goal state. For each state $s_{i<n}$, we consider the probabilistic planning problem of reaching any later state $s_{j>i}$, from s_i. Each of these problems identified by the start state s_i and the target set of termination states $s_{j>i}$, forms a subproblem of the original problem. We then use the solutions of these n subproblems to move from s_0 to later and later states s_i and eventually to the goal s_n.

Once a deterministic plan has been generated, we modify the original state space by considering the states along the plan as absorbing termination states. The presence of these additional termination states aids in a quick convergence of the probabilistic planner. We first start from the original start state s_0 and run the planner until it converges and returns a policy that includes one or more of these additional termination states. We then rerun the planner on each successive plan state $s_{i<n}$, after removing it from the set of absorbing termination states. This sequence of policies from each of the planner runs forms a nonstationary policy for the original problem.

During execution we first apply the policy from the subproblem corresponding to the state s_0. The execution of this policy leads to any one of the termination states s_i in that policy tree. We then apply the policy starting at state s_i and continue this process until the goal s_n is reached. The sequence of policies are essentially used to get from one state to the next along a deterministic plan to the goal, where the choice of these states depends on that particular execution trace. Since we have a policy starting at each state along the plan we ensure that we have a complete policy (i.e. an action for each state reachable from the start state), that eventually leads to the goal.

Implementation Our planner is implemented in C and makes use of several existing technologies. Since we use the FF-replan determinization method and FF for generating the deterministic plan, we have built our planner as an extension to the FF deterministic planner. This allows us to leverage the heuristic computation and deterministic planning capabilities of FF. We have modified FF's problem definition parser to handle probabilistic outcomes and made use of the hashing technique used by FF to memoize our policy and value function. Our code for LDFS is based on the pseudocode provided in the paper on LDFS.

Appendix

Learning Depth First Search LDFS (Bonet and Geffner 2006) unifies the concepts of dynamic programming and heuristic search to create a method that uses iteratively deepened depth first search to find and update memoized state value functions until they converge to the optimal value. A heuristic evaluation function is used to initialise the state value functions. By selectively identifying and updating regions of the state space LDFS is able to converge by driving all the Bellman errors to zero. An important factor aiding this convergence is the use of the planning problem's initial state as the starting point for the algorithm. This restricts the state space that LDFS needs to work on, making it much more efficient than traditional value iteration. In addition, the goal state from the planning problem is regarded as an absorbing state with a value of zero, which provides a termination condition for the search process. LDFS works by examining the Bellman error of the current state for the current value function. If the error is zero the action corresponding to the minimum Q-value is expanded, adding the successor states to a depth-first search stack. This action is also recorded in the policy for this state. If at some point a state is reached where the Bellman error is not zero; a backup is performed where the state's value is updated to equal the minimum Q-value among its applicable actions. This update is then reflected by backing up along the depth first search path to this node and performing any value updates as necessary. This process terminates when the depth first stack is empty, thus implying that all states in the current policy have converged to their optimal values.

LDFS can be formulated for probabilistic planning by assuming an action cost of one and considering the value of the absorbing goal states to be zero. The optimal value function for such domains is essentially the minimum expected distance to the goal. We seek an initial heuristic value function that can provide a rough but admissible estimate of this distance so that the number of updates necessary in order to converge to the optimal value are minimal. In the deterministic planning framework the relaxed plan heuristic used by the FF planning system has proven to be very successful in providing an admissible estimate of the distance to the goal. This heuristic is calculated by first relaxing the problem complexity by ignoring all the delete effects of actions and then calculating the distance to the goal. We apply a similar heuristic as the initial value function for use with LDFS on probabilistic planning problems since it closely approximates the distance to the goal. However, we need to first make some modifications to this relaxed plan heuristic calculation method so that it can be applied to the probabilistic domain.

FF-replan determinization In order to be able to apply FF's heuristic computation method we need to first determinize the domain definition. We follow the *"all-outcomes"* determinization principle used by FF-replan that considers each probabilistic outcome of an action as a separate deterministic action. Once the domain definition has been determinized, we follow the relaxed plan heuristic computation on this modified domain definition. Since each probabilistic outcome is considered as a separate deterministic action FF is free to choose among any of these outcomes in the heuristic computation. The combination of this fact, and the fact that the heuristic distances are calculated on a relaxed domain free of delete effects makes it possible to derive admissible heuristic estimates on most domains. In addition to serving as the initial value function estimates for use with LDFS the heuristic distance values are used with the regular FF code to derive a plan from the start state to the goal on the determinized domain. This plan is then used in determining the sequence of subproblems that are used to solve the original probabilistic planning problem.

References

Bonet, B., and Geffner, H. 2006. Learning depth-first search : A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In *"International Conference on Automated Planning and Scheduling"*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 253–302.

Yoon, S.; Fern, A., and Givan, R. 2007. FF-replan: A baseline for probabilistic planning. In *"International Conference on Automated Planning and Scheduling"*.