Generating Plans in Concurrent, Probabilistic, Over-Subscribed Domains

Li Li
Department of Computer Science
Michigan Technological University
Houghton, MI 49931
lili@mtu.edu

Abstract

My thesis topic is plan generation in temporal, parallel, probabilistic domains with oversubscribed goals. I have defined a framework that includes two novel extensions. First, the plans can include parallel steps that serve the same goal and increase the probability of success in addition to parallel steps that serve different goals and decrease execution time. Second, already executing plan steps can be terminated if doing so saves resources to achieve more goals. My algorithm called CPOAO* (Concurrent, Probabilistic, Oversubscription AO*) can deal with these extensions. In this paper, I summarize the design and implementation of CPOAO* and its associated heuristics, and propose a plan of research.

Introduction

Generating plans in realistic domains presents two main challenges. First, there is uncertainty about the action effects as well as the state of the world. Second, the resources for carrying out the tasks are limited. Research in *probabilistic planning* deals with issues involving uncertainty (Kushmerick, Hanks, and Weld 1995; Bonet and Geffner 2005). *MDP-based planners* (Boutilier, Dean, and Hanks 1999) deal with both uncertain actions and resource limitations. Heuristics have been developed for dealing with time and resources in deterministic domains (Haslum and Geffner 2001). *Oversubscription planners* must select a subset of the goals to plan for because resource limitations do not allow all the goals to be achieved (Smith 2004; Benton, Do, and Kambhampati 2005). *Concurrent planners* generate plans with shorter makespan by using parallel actions (Little and Thiebaux 2006; Mausam and Weld 2008). My work extends recent research by considering durative, concurrent, probabilistic actions in over-subscribed domains; by exploring the use of parallel actions to increase expected rewards; and by providing a framework to terminate actions to optimize the expected rewards.

Consider a simplified Mars rover domain (Bresina et al. 2002) where two pictures must be taken within 5 minutes, and actions have fixed known durations. The rover has two cameras: cam_0 succeeds with probability 0.6, and takes 5 minutes; and cam_1 succeeds with probability 0.5, and takes 4 minutes. In a case where both pictures have a value of 10, the best strategy is to use cam_0 for one picture and cam_1 for the other in parallel. Because all the actions need to finish to collect the rewards, we call this *all-finish parallelism*. In a different case where the picture values are 100 and 10, the best strategy is to use both cam_0 and cam_1 in parallel to achieve the larger reward. In this case, the success of the earliest finishing action is sufficient. We call this case *early-finish parallelism*.

When both cameras are used for the same picture, if the faster camera (cam_1) succeeds in achieving the target reward, we can abort cam_0 immediately unless it serves other goals. In multi-agent domains or in parallel single-agent domains, there is advantage in terminating actions as soon as the expected result is obtained, or as soon as it becomes impossible to obtain the expected results. My algorithm is capable of marking the actions to be terminated so that resources can be saved to achieve other goals.

In this paper, I first define the planning problem. I then describe my progress on this topic. I conclude with an outline of my research plan.

The Planning Problem

In my thesis, I define a *planning problem* as a 5-tuple (S, A, s_0, R, T) where S is the state space, A is the set of actions, s_0 is the initial state, R is the reward set, and T is the time limit for plan execution.

A state is a triple consisting of a set of propositions, a vector of numeric resource values, and the set of currently executing concurrent actions. An action a consists of a precondition list, a resource consumption vector, and a set of probabilistic outcomes. The initial state contains all the propositions that are true at the onset. Each reward in R is a proposition-value pair. The time limit is a deadline for finishing the steps.

The solution to a planning problem is an *optimal contingent plan* that maximizes the expected reward of the initial state. We formally define a *plan* as a set of (state, <CAS,TS>) pairs where <CAS,TS> denotes the actions that will be taken at the given state. A CAS is a *concurrent action set* that denotes the set of plan steps that will be started in parallel. A TS is a *termination set* that denotes the set of executing steps that will be terminated. The objective of

my research is to develop an algorithm and heuristics that can solve such problems. In the next section, I outline my progress.

Research Activities

I have developed and implemented an algorithm named CPOAO* (Concurrent, Probabilistic, Oversubscription AO*). CPOAO* searches in a space of states considering the possible actions that can be started and the executing actions that can be terminated in each state. Each node in the search tree represents a state, and the root node is the initial state. The *value* of a state is the sum of the expected rewards in this state. The value of a terminal node is computed by adding the values of the rewarded propositions.

My research to date has been concerned with increasing the efficiency of CPOAO* by considering two questions related to the search process: (1) How can the set of applicable <CAS,TS> pairs be selected effectively? (2) How can the search be guided? To address the first question, I use rules to prune concurrent action sets based on a "better" relationship. For example, we should never choose a concurrent action set that undoes the effects of the just finished actions. For domains where time is the only resource modeled, starting an action earlier is always better than starting it later or being idle because we can abort it at any time. To address the second question, I designed and implemented a technique based on reachability analysis where I construct a *reverse planning graph (rpgraph)* to compute the expected rewards of a state. For each goal, an rpgraph is created by first placing the goal proposition at the initial level. Then, alternating levels of action and proposition nodes are created. For each proposition at the last level, an action that can achieve this proposition is inserted into the next level. Each proposition node contains a resource list indicating the resources required to reach the goal from this proposition node. The resource levels are updated using the resource requirements of the actions. The expansion of the rpgraph stops when the level of any resource exceeds the value present in the initial state. Each time a new intermediate node is generated, this graph is used to compute an upper bound on the expected rewards and thus heuristically guide the search. I have reported the preliminary results of my experiments in (Li and Onder 2007). My recent experiments show up to one order of magnitude improvement in the search process. In the next section, I outline my research plan.

Proposed Research

My future activities will be in the following areas:

Heuristics: I plan to improve the heuristics I have implemented. Possibilities include a factored representation of the reward structure and using previously cached values to find the expected rewards. Theories on how to detect useless actions in parallel domains can be developed. One possibility is to use causal link analysis to find the goals that are served by each action and terminating the actions that serve the same goal when an action finishes with success. Similarly, achieving some goals might become impossible when an action finishes with failure. The other actions that serve the same goal might be terminated. Both admissible and nonadmissible heuristics can be explored for finding expected rewards and action termination conditions.

Other approaches: I plan to integrate the research in concurrent MDP (co-MDP) based frameworks into my framework that includes early finish parallelism and action termination. I believe the transfer of ideas can be both ways: my reachability-based heuristics can be used in MDP frameworks and dynamic programming techniques for MDPs can be used in my framework.

Domain modeling: In light of my research findings in the previous two areas, I plan to extend the current domain model to include continuous resources, renewable resources, and stochastic resource consumption. This will bring my framework closer to real world systems.

References

Benton, J.; Do, M. B.; and Kambhampati, S. 2005. Oversubscription planning with numeric goals. *Proc. IJCAI-05*.

Bonet, B., and Geffner, H. 2005. mGPT: A probabilistic planner based on heuristic search. *J. of Artificial Intelligence Research* 24:933–944.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision theoretic planning: Structural assumptions and computational leverage. *J. of Artificial Intelligence Research* 11:1–94.

Bresina, J. L.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D. E.; and Washington, R. 2002. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proc. UAI-02*, 77–84.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *Proc. 6th European Conference on Planning*.

Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76:239–86.

Li, L., and Onder, N. 2007. Generating plans in concurrent, probabilistic, over-subscribed domains. In *ICAPS-07 3rd Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices for Planning in Execution*.

Little, I., and Thiebaux, S. 2006. Concurrent probabilistic planning in the graphplan framework. In *Proc. ICAPS-06*, 263–272.

Mausam, and Weld, D. S. 2008. Planning with durative actions in stochastic domains. *J. of Artificial Intelligence Research* 31:33–82.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proc. ICAPS-04*.