POND: The Partially-Observable and Non-Deterministic Planner

Daniel Bryce
Department of Computer Science and Engineering
Arizona State University, Brickyard Suite 501
699 South Mill Avenue, Tempe, AZ 85281
dan.bryce@asu.edu

Abstract

This paper describes *POND*, a planner developed to solve problems characterized by partial observability and non-determinism. *POND* searches in the space of belief states, guided by a relaxed plan heuristic. Many of the more interesting theoretical issues showcased by *POND* show up within its relaxed plan heuristics. Namely, the exciting topics are defining distance estimates between belief states, efficiently computing such distance estimates on planning graphs, and sharing planning graphs and relaxed plans between belief states.

Introduction

The *POND* planner solves many types of planning problems characterized by uncertainty, whether they are non-deterministic/probabilistic, are non/partially observable, or have deterministic/uncertain actions. *POND* accepts PPDDL-like[1] (Younes & Littman 2004) problem descriptions and generates conformant and conditional plans. *POND* searches forward in the space of belief states, similar to GPT (Bonet & Geffner 2000), using various search algorithms (A*, AO*, LAO*, Enforced Hill-Climbing) depending on the problem and user preferences. To compute heuristics for search, *POND* can use several different planning graph techniques. We start by discussing some of the theory that goes into computing the planning graph heuristics used by *POND*, and then describe the planner implementation.

Theory

Since *POND* can handle several types of planning problems, we concentrate on the techniques used for conformant non-deterministic planning. We refer to (Bryce, Kambhampati, & Smith 2006a) and (Bryce, Kambhampati, & Smith 2006b) for additional techniques, not described here.

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[1]PPDDL extended for various things such as non-determinism, observations, goal probability thresholds, etc.

Belief State Distance: To search in belief space, *POND* estimates the conformant plan distance between the belief state(s) at the end of its current plan prefix and a goal belief state. The distance between belief states is taken as an aggregate measure of the underlying distances between states in the belief states. For instance, a possible admissible measure would find the minimum distance from every state in the current belief state to a state in the goal belief state, then take the maximum of these (this is the measure used by GPT). Since taking the maximum of the minimum state distances assumes full positive interaction between the states, we would not account for many of the actions that differ between the sequences for each state (i.e., miss independence). Taking the summation of the minimum state distances would assume full independence, but miss the positive interaction. Instead, we use a measure that exploits both positive interaction and independence. By analogy to plan merging, we would like to merge the action sequences for each of the states in the current belief state so that actions *overlap* as much as possible (Bryce, Kambhampati, & Smith 2006a). The resulting merged plan contains all actions used in common or independently by the different states in the belief state. We can obtain this measure by computing a classical relaxed plan for each state in our current belief state and merging the relaxed plans. However, there may be many states in our belief state and computing a planning graph for each state is costly.

Heuristic Computation: In order to compute our belief state distance measure without constructing multiple explicit planning graphs, we use a planning graph generalization, called the Labeled Uncertainty Graph (*LUG*) (Bryce, Kambhampati, & Smith 2006a). The *LUG* represents multiple explicit planning graphs implicitly. The idea is to use a single planning graph skeleton to represent common action and proposition connectivity, and use annotations (labels) that denote which planning graph components exist in the explicit planning graphs. Labels are propositional formulas, whose models correspond to states in the belief state. We can determine which explicit planning graphs contain a proposition by examining the models of the proposition's label. If a state entails the label of the proposition at level k, the proposition is in the explicit planning graph for the state at level k.

Using LUG connectivity, we can determine which actions are needed to support the goal propositions, and using labels we know when we have chosen enough actions to support the goals from each of the states in our belief state. Thus, we can extract a relaxed plan that represents an implicitly merged plan for each of the states in our belief state. This relaxed plan indicates the plan distance to transition each of the states in a belief state to a goal belief state.

State Agnostic Planning Graphs: The LUG implicitly represents a set of explicit planning graphs. Using a state agnostic planning graph (SAG) (Cushing & Bryce 2005), a generalization of the LUG, we can build a LUG for every possible state. The SAG and LUG are identical except for which states are represented and how we compute relaxed plans. To extract the relaxed plan for a belief state from the SAG (assuming the belief state is represented by a propositional formula) we need to take the conjunction of each label with the belief state to reveal the LUG for the belief state. Those planning graph elements where the conjunction is satisfiable are in the revealed LUG. By computing the SAG, we construct a single, sometimes costly, LUG whose cost is amortized over each belief state.

Global Relaxed Plan: Alternative to using the SAG to compute a relaxed plan for each belief state, we can compute a global relaxed plan. The global relaxed plan continues the SAG generalization by making a state agnostic relaxed plan. We extract the global relaxed plan, which is a relaxed plan for the belief state containing all states, and then for each belief state encountered in search we restrict the global relaxed plan to the actions needed for the belief state. By restricting the global relaxed plan, we mean that we take the conjunction of each action's label in the global relaxed plan with the belief state formula. Those actions where the conjunction is satisfiable are in the relaxed plan. The global relaxed plan is admittedly less accurate than extracting a relaxed plan from the SAG for a specific belief state, but is very fast to compute.

Lazily Enforced Hill-Climbing: $POND$ uses a lazily enforced hill-climbing search, guided by three heuristics: belief state cardinality (Bertoli, Cimatti, & Roveri 2001), the global relaxed plan and the SAG relaxed plan. These heuristics have an increasing computation cost (and accuracy). The basic idea is to the use the cardinality heuristic in hill-climbing, as long as it improves the heuristic distance to the goal belief state. If the cardinality of the current children search nodes does not decrease, then we re-evaluate the children with the global relaxed plan (a slightly more costly, but better heuristic). If the global relaxed plan cannot find a better child, then we switch to the SAG relaxed plan in an A* search rooted at our current search node.

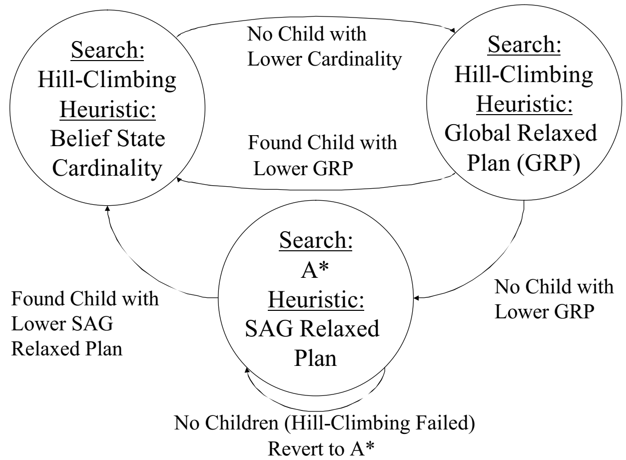

Figure 1: Lazily Enforced Hill-Climbing strategy.

The SAG relaxed plan is the costliest heuristic, but the most informed. Once the A* search finds a better cost child, we resume hill-climbing with cardinality. If search fails, we revert to A* search with the global relaxed plan heuristic. The automata in Figure 1 depicts our search strategy. Since the heuristic landscape defined by cardinality, the global relaxed plan, and SAG relaxed plan are different, we say the search is lazily enforced hill-climbing. It is not always the case that using one heuristic to escape a heuristic plateau of another heuristic decreases the original distance to the goal belief state. However, since we use the heuristics in order of increasing accuracy, we are more confident in the direction chosen by the heuristics even if it means an increase in the original heuristic distance.

Implementation

$POND$ is implemented in C++ and uses several existing technologies. It employs the PPDDL parser (Younes & Littman 2004) for input, the IPP planning graph construction code (Koehler *et al.* 1997) for the LUG, and the CUDD BDD package (Somenzi 1998) for representing belief states, actions, and labels. $POND$ resembles MBP (Bertoli *et al.* 2001) because it uses BDDs to represent belief states and actions, and uses BDD operations to symbolically compute the transition between belief states. $POND$ is perhaps most similar to KACMBP (Bertoli & Cimatti 2002) because we use both cardinality and reachability heuristics, however our reachability heuristics are based on conformant relaxed plans.

Acknowledgements: This work was supported by the NSF grant IIS-0308139, the ONR grant N000140610058, the MCT/NASA summer fellowship program, the ARCS foundation, an IBM faculty award, and Arizona State University. We would like to thank Subbarao Kambhampati, David E. Smith, and William Cushing for contributing to the theory used in $POND$.

Much additional thanks is also given to William Cushing for help with implementation.

References

Bertoli, P., and Cimatti, A. 2002. Improving heuristics for planning as search in belief space. In *Proceedings of AIPS'02*.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of IJCAI'01*.

Bertoli, P.; Cimatti, A.; and Roveri, M. 2001. Heuristic search + symbolic model checking = efficient conformant planning. In *Proceedings of IJCAI'01*.

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

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of AIPS'00*.

Bryce, D.; Kambhampati, S.; and Smith, D. 2006a. Planning graph heuristics for belief space search. *JAIR*. (To appear).

Bryce, D.; Kambhampati, S.; and Smith, D. 2006b. Sequential monte carlo in probabilistic planning reachability heuristics. In *Proceedings of ICAPS'06*.

Cushing, W., and Bryce, D. 2005. State agnostic planning graphs. In *Proceedings of AAAI'05*.

Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y. 1997. Extending planning graphs to an adl subset. In *Proceedings of ECP'97*.

Somenzi, F. 1998. *CUDD: CU Decision Diagram Package Release 2.3.0*. University of Colorado at Boulder.

Younes, H., and Littman, M. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical report, CMU-CS-04-167, Carnegie Mellon University.