Conditional Planning under Partial Observability as Heuristic-Symbolic Search in Belief Space

Piergiorgio Bertoli[1] and **Alessandro Cimatti**[1] and **Marco Roveri**[1,2]
{bertoli,cimatti,roveri}@irst.itc.it
[1] ITC-IRST, Via Sommarive 18, 38055 Povo, Trento, Italy
[2] DSI, University of Milano, Via Comelico 39, 20135 Milano, Italy

Abstract

Planning under partial observability in nondeterministic domains is a very significant and challenging problem, which requires dealing with uncertainty together with and-or search. In this paper, we propose a new algorithm for tackling this problem, able to generate conditional plans that are guaranteed to achieve the goal despite of the uncertainty in the initial condition and the uncertain effects of actions. The proposed algorithm combines heuristic search in the and-or space of beliefs with symbolic BDD-based techniques, and is fully amenable to the use of selection functions. The experimental evaluation shows that heuristic-symbolic search may behave much better than state-of-the-art search algorithms, based on a depth-first search (DFS) style, on several domains.

Introduction

In this paper, we tackle the problem of conditional planning under partial observability, where only part of the information concerning the domain status is available at run time. In its generality, this problem is extremely challenging. Compared to the limit case of full observability, it requires dealing with uncertainty about the state in which the actions will be executed. Compared to the limit case of null observability, also known as conformant planning, it requires the ability to search for, and construct, plans representing conditional courses of actions. Several approaches to this problem have been previously proposed, e.g. (Weld, Anderson, and Smith 1998), based on extensions of GraphPlan, and (Bonet and Geffner 2000), based on Partially Observable Markov Decision Processes (POMDP).

Our work builds on the approach proposed in (Bertoli et al. 2001b), where planning is seen as and-or search of the (possibly cyclic) graph induced by the domain, and BDD-based techniques borrowed from symbolic model checking provide efficient search primitives. We propose a new algorithm for planning under partial observability, able to generate conditional acyclic plans that are guaranteed to achieve the goal despite of the uncertainty in the initial condition and in the effects of actions. The main feature of the algorithm is the heuristic style of the search, that is amenable to the use of selection functions, and is fully compatible with the use of a symbolic, BDD-based representation. We call this approach *heuristic-symbolic* search. The proposed approach differs from (and improves) the depth-first search proposed in (Bertoli et al. 2001b) in several respects. First, depending on the selection function, it can implement different styles of search, including DFS. Furthermore, the use of selection functions allows to overcome potentially bad initial choices, and can therefore result in more efficient computations and higher quality plans. Finally, it opens up the possibility of using preprocessing techniques for determining domain/problem-dependent heuristics. The heuristic-symbolic algorithm was implemented in the MBP planner (Bertoli et al. 2001a), and an extensive experimental evaluation was carried out. The results show that, even considering a simple domain-independent heuristic, for several classes of problems the heuristic-symbolic algorithm significantly improves the performance and constructs better plans with respect to DFS.

The paper is organized as follows. In Section we describe partially observable planning domains and conditional planning. In Section we present the planning algorithm. In Section we give an overview of the experimental evaluation, draw some conclusions and discuss some future work.

Domains, Plans, and Planning Problems

A partially observable planning domain is a tuple $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{O}, \mathcal{X} \rangle$. $\mathcal{P}$ is a finite set of propositions. $\mathcal{S} \subseteq Pow(\mathcal{P})$ is the set of states. $\mathcal{A}$ is a finite set of actions. $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation, describing the effects of (possibly nondeterministic) action execution. We say that an action a is applicable in a state s iff there exists at least one state s' such that $\mathcal{R}(s, a, s')$. $\mathcal{O}$ is a set of boolean observation variables, whose values can be observed during plan execution. $\mathcal{X} : \mathcal{O} \rightarrow Pow(\mathcal{S} \times \{\top, \bot\})$ is the observation relation. Intuitively, $\mathcal{X}$ associates each possible value of each observation variable to the set of states where observing the variable returns such value. (We consider observation variables to be always defined, and independent from actions performed prior to observing. The full framework (see (Bertoli et al. 2001b)) and the actual MBP implementation are free from these constraints. The current presentation is simplified for reasons of space.)

We consider conditional plans, branching on the value of observable variables. A plan for a domain $\mathcal{D}$ is either the empty plan ϵ, an action $a \in \mathcal{A}$, the concatenation $\pi_1; \pi_2$ of

two plans π_1 and π_2, or the conditional plan $o\ ?\ \pi_1 : \pi_2$ (read "if o then π_1 else π_2"), with $o \in \mathcal{O}$. The execution of a plan π must take into account, at each step, that the executor can be unable to distinguish between a set of possible states for the current situation. We call such a set of indistinguishable states a "belief state". An action a is applicable to a belief state Bs iff a is applicable in all states of Bs. Intuitively, the execution of a conditional plan π starting from a belief state Bs results into a set of states recursively defined over the structure of π by either (a) the application of the transition relation, if an action is performed, or (b) the union of the executions on every branch resulting from the possible observation values for a variable o, if the plan branches on o. We say that a plan is applicable on Bs if no non-applicable actions are met during its execution on Bs. A planning problem is a 3-tuple $\langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ whose components are a planning domain $\mathcal{D}$, a set of initial states $\mathcal{I}$ and a set of goal states $\mathcal{G}$. A solution consists of a plan π which is applicable over $\mathcal{I}$, and whose execution on $\mathcal{I}$ results in a belief state equal to $\mathcal{G}$ or contained into it.

Consider the example of a simple robot navigation domain $\mathcal{D}_0$, a 2x2 room with an extra wall (Figure 1). The propositions of the domain are `NW`, `NE`, `SW`, and `SE`, i.e. the four positions in the room. Exactly one of them holds in each of the four states in $\mathcal{S}$. The robot can move in the four directions, unless the action is not applicable due to a wall standing in the direction of motion. At each time tick, the information of walls proximity in each direction is available to the robot (observation variables `WallN`, `WallS`, `WallW` and `WallE`). In the figure, the execution of the plan π_0 = `GoEast ; WallN ? (GoSouth ; GoWest ) : GoWest`, starting from the uncertain initial condition `NW` or `SW`, is outlined by solid lines. The plan π_0 is a solution for the problem P_0 = {$\mathcal{D}_0$, {`NW`, `SW`}, {`SW`} }.

Planning under Partial Observability

When planning under partial observability, the search space can be seen as an and-or graph, recursively constructed from the initial belief state, expanding each encountered belief state by every possible combination of applicable actions and observations. The graph is possibly cyclic; in order to rule out cyclic behaviors, however, the exploration can be limited to its acyclic prefix. Figure 1 depicts the finite prefix of the search space for the problem P_0 described above. Each node in the prefix is associated to a path, describing how the node has been reached, and to the corresponding belief state.

The prefix is constructed by expanding each node in all possible ways, each represented by an outgoing arc. Single-outcome arcs correspond to applicable actions (action execution is deterministic in belief space). For instance, N4 expands into N7 and N8. Multiple outcome arcs correspond to observations. For instance, node N2 results in nodes N4 and N5, corresponding to the observation of `WallN`. The application of an observation is what gives the "and" component in the search space: we have a solution for (the belief state associated with) N2 if we have a solution for both N4 and N5 (whose associated belief states are obtained by conjoining the beliefs associated by $\mathcal{X}$ to the observed values of `WallN` with the belief state associated to N2). Some non-informative observations are not reported in the graph.

```
HEURSYMCONDPLAN(I, G)
1  graph := MKINITIALGRAPH(I, G);
2  while (GRAPHROOTMARK(graph) ∉ {Success, Failure})
3    node := EXTRACTNODEFROMFRONTIER(graph);
4    if (SUCCESSPOOLYIELDSSUCCESS(node, graph))
5      MARKNODEASSUCCESS(node);
6      NODESETPLAN(node,RETRIEVEPLAN(node, graph));
7      PROPAGATESUCCESSONTREE(node,graph);
8      PROPAGATESUCCESSONEQCLASS(node,graph);
9    else
10     orexp := EXPANDNODEWITHACTIONS(node);
11     andexp := EXPANDNODEWITHOBSERVATIONS(node);
12     EXTENDGRAPHOR(orexp, node, graph);
13     EXTENDGRAPHAND(andexp, node, graph);
14     if (SONSYIELDSUCCESS(node))
15       MARKNODEASSUCCESS(node);
16       NODESETPLAN(node,BUILDPLAN(node));
17       PROPAGATESUCCESSONTREE(node, graph);
18       PROPAGATESUCCESSONEQCLASS(node, graph);
19     else if (SONSYIELDFAILURE(node))
20       MARKNODEASFAILURE(node, graph);
21       NODEBUILDFAILUREREASON(node, graph);
22       PROPAGATEFAILUREONTREE(node, graph);
23       PROPAGATEFAILUREONEQCLASS(node, graph);
24 end while
25 if (GRAPHROOTMARK(graph) = Success)
26   return GRAPHROOTPLAN(graph);
27 else
28   return Failure;
```

Figure 2: The planning algorithm

The expansion of a node n is halted when (a) n is associated with a belief state contained in the goal, or (b) n is a loop back, i.e. it has an ancestor node with the same belief state. For instance, node N6 loops back onto node N1, while node N11 loops back onto node N4. Node N9 and N10 are associated with the goal belief state.

The planning algorithm for conditional planning under partial observability is described in Figure 2. It takes in input the initial belief state and the goal belief state, while the domain representation is assumed to be globally available to the subroutines. The algorithm incrementally constructs the finite acyclic prefix described above. The algorithm relies on an extended data structure, stored in the $graph$ variable, which represents the prefix being constructed. Each node in the structure is associated with a belief state and a path. In addition to son-ancestor links, the graph has links between the nodes in the equivalence classes induced by equality on associated belief states, and presents an explicit representation of the frontier of nodes to be expanded. A success pool contains the solved nodes for the graph.

At line 1, the algorithm initializes the graph: the root node corresponds to the initial belief state, and the success pool contains the goal. Then (lines 2-24) the iteration proceeds by selecting a node and expanding it, until a solution is found or the absence of a solution is detected. At loop exit, either

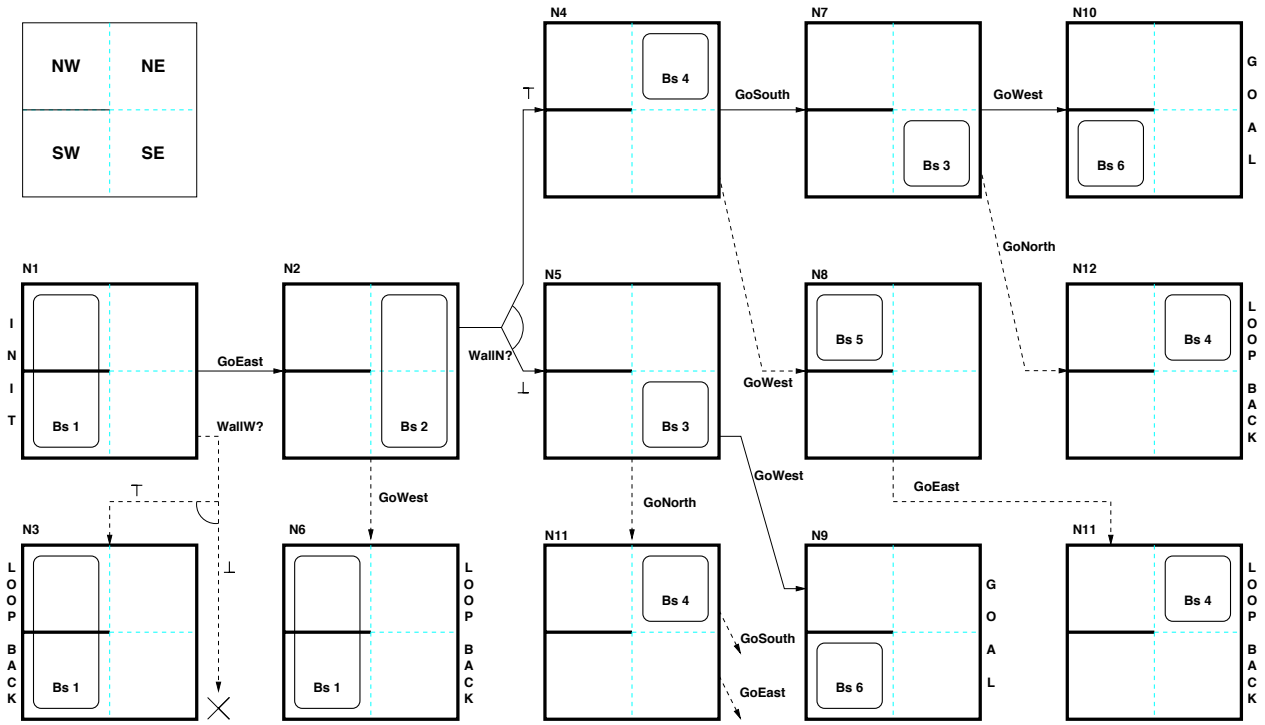

Figure 1: A simple robot navigation domain

a plan or a failure is returned (lines 25-28).

With the first step in the loop, at line 3, a node is extracted from the frontier in order to be expanded. The EXTRACTNODEFROMFRONTIER primitive embodies the selection criterion and is responsible for the style (and the effectiveness) of the search. Then, at line 4, we check whether the belief state associated to the selected $node$ is entailed by some previously solved belief state in the success pool. If so, the formerly detected plan is reused for $node$, which is marked as success. Moreover, the success is recursively propagated both to the ancestors of $node$ (line 7) and to the nodes in its equivalence class (line 8). The rules for success propagation directly derive from the and-or graph semantics. Recursive success propagation takes also care of removing descendents of success nodes from the frontier (as their expansion would be useless).

If the success of $node$ cannot be derived by the success pool, then the expansion of $node$ is attempted, computing the nodes resulting from possible actions (line 10) and observations (line 11). The graph extension steps, at lines 12-13, construct the nodes associated to the expansion, and add them to the graph, also doing the bookkeeping operations needed to update the frontier and the links between nodes. In particular, for each node, the associated status is computed. For instance, if a newly constructed node has a belief state that is already associated with a plan, then the node is marked as success. Newly constructed nodes are also checked for loops, i.e. if they have an ancestor node with the same belief state then they are marked as failure.

If it is possible to state the success of $node$ based on the status of the newly introduced sons (primitive SONSYIELDSUCCESS at line 14), then the same operations at line 5-8 for success propagation are executed. Similarly, at lines 19-23, if it is possible to state the failure of $node$ based on the status of the newly introduced sons, failure is propagated throughout the and-or graph. Failure can happen, for instance, due to loop detection. The failure of the node is stored in such a way that it can be reused in the following search attempts. Notice however that, differently from success, a failure depends on the node path. For instance, in a subsequent search attempt it could be possible to reach a belief state with a non-cyclic path. Therefore, each belief state is associated with a set of belief states representing the failure reason. Intuitively, the failure reason contains the sets of belief states that caused a loop in all the search attempts originating from the belief state marked with failure.

The algorithm is integrated in MBP (Bertoli et al. 2001a), a general planner for nondeterministic domains which allows for conditional planning under full observability, also considering temporally extended goals, and for conformant planning. MBP is based on the use of symbolic model checking techniques (McMillan 1993). In particular, it relies on Binary Decision Diagrams, structures that allow for a compact representation of sets of states and an efficient expansion of the search space (see (Cimatti and Roveri 2000) for an introduction to the use of BDDs in planning).

Results and Conclusions

We carried out an extensive experimental analysis of the heuristic-symbolic algorithm presented in previous section, comparing it with the DFS approach of (Bertoli et al. 2001b), shown to outperform other conditional planners such as SGP and GPT. For lack of space, we only provide a high-level description of the considered domains and results. The details can be found in (Bertoli, Cimatti, and Roveri 2001). We considered the standard benchmark problems for PO planning used in (Bertoli et al. 2001b): MAZE, Empty Room (ER), RING. In the MAZE, a moving robot must reach a fixed position in a maze, starting from anywhere and being able to observe the walls around its current position. Basically, this problem reduces to gathering knowledge about the robot position. Unless the maze is significantly symmetric, almost at each move of the robot, observing contributes to the purpose. Furthermore, the problem is highly constrained: observing is forced in many situations by the lack of applicable actions prior to that. In the ER, the same problem is tackled considering, rather than a maze, a wide empty room; the robot starts from anywhere in the room. In this formulation, most moves of the robot will lead to "enabling" some useful sensing. In the RING, the aim is to have all windows of a ring of connected rooms locked by a moving robot. Each window must be closed, if open, before being locked (if unlocked). Here, the key issue is that of locality: before moving around, the robot should better solve the local problem of locking the local window. Otherwise, plans may become extremely lengthy. Thus, observing and moving must be interleaved "in a sensible way". Furthermore, we considered some variations to the ER. In the VER problem, the robot initially is in one of two positions near the center of the room. This forces the robot to execute long sequences of actions before being able to gather some useful information from sensing the walls. In the ERS, a portion of the empty room is a "sink", i.e. once entered there, the robot cannot exit it.

The performance of the heuristic-symbolic algorithm heavily depends on the selection function, that controls which portions of the search space are explored. The problem of finding an effective, problem dependent selection function for controlling and-or search appears to be in general very hard. In all our experiments, we considered a simple structural selection function, that gives high scores to nodes whose equivalence class contains many open nodes and few failed ones. In spite of this choice, the results are quite promising. In the RING, ER, VER, ERS problems the timing of the search, and the length of the plan (defined as its maximum depth) are much better than in the original DFS search. The MAZE problem evidences a reasonable loss of efficiency. This is due to the fact that the problem is very constrained, and drives the DFS search accordingly. In this case, the overhead of maintaining a graph structure and having explicit propagation routines explains the result.

The conclusion that can be drawn is that giving up the DFS-style search is in general an advantage, leading in many cases to better results even in absence of highly tuned scoring mechanisms, and opens up the possibility for further improvements. Future research will be directed to the definition of preprocessing techniques and more effective heuristic functions, with the goal to obtain "smarter" behaviors from the heuristic-symbolic algorithm. Another direction of future research is the extension of the partially observable approach presented in this paper to find strong cyclic solutions, and to deal with goals expressed in a temporal logic.

References

Bertoli, P.; Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2001a. MBP: a Model Based Planner. In *Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information*.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001b. Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking. In *Proc.* 7^{th} *International Joint Conference on Artificial Intelligence (IJCAI-01)*. AAAI Press.

Bertoli, P.; Cimatti, A.; and Roveri, M. 2001. Conditional Planning under Partial Observability as Heuristic-Symbolic Search in Belief Space. Technical report, IRST, Trento, Italy. Extended version of this paper.

Bonet, B., and Geffner, H. 2000. Planning with Incomplete Information as Heuristic Search in Belief Space. In Chien, S.; Kambhampati, S.; and Knoblock, C., eds., 5^{th} *International Conference on Artificial Intelligence Planning and Scheduling*, 52–61. AAAI-Press.

Cimatti, A., and Roveri, M. 2000. Conformant Planning via Symbolic Model Checking. *Journal of Artificial Intelligence Research (JAIR)* 13:305–338.

McMillan, K. 1993. *Symbolic Model Checking*. Kluwer Academic Publ.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending graphplan to handle uncertainty and sensing actions. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 897–904. Menlo Park: AAAI Press.