

Improving Heuristics for Planning as Search in Belief Space

Piergiorgio Bertoli and Alessandro Cimatti

IRST - Istituto Trentino di Cultura, 38050 Povo, Trento, Italy

{bertoli,cimatti}@irst.itc.it

Abstract

Search in the space of beliefs has been proposed as a convenient framework for tackling planning under uncertainty. Significant improvements have been recently achieved, especially thanks to the use of symbolic model checking techniques such as Binary Decision Diagrams. However, the problem is extremely complex, and the heuristics available so far are unable to provide enough guidance for an informed search.

In this paper we tackle the problem of defining effective heuristics for driving the search in belief space. The basic intuition is that the “degree of knowledge” associated with the belief states reached by partial plans must be explicitly taken into account when deciding the search direction. We propose a way of ranking belief states depending on their degree of knowledge with respect to a given set of boolean functions. This allows us to define a planning algorithm based on the identification and solution of suitable “knowledge subgoals”, that are used as intermediate steps during the search. The solution of knowledge subgoals is based on the identification of “knowledge acquisition conditions”, i.e. subsets of the state space from where it is possible to perform knowledge acquisition actions. We show the effectiveness of the proposed ideas by observing substantial improvements in the conformant planning algorithms of MBP.

Introduction

Planning in nondeterministic domains, i.e. with uncertainty in the initial condition, partial observability, and uncertain action effects, is increasingly being recognized as an important research area. In practical domains, the ability to deal with contingencies during the planning process is often required, for instance in order to anticipate unpredictable events and counter possible failures. Significant relationships are also being recognized with the fields of controller synthesis, diagnosis (Thiebaux & Cordier 2001), and Automated Test Pattern Generation (Cimatti, Roveri, & Bertoli 2001). Planning in nondeterministic domains is also one of the most challenging problems, proved to be significantly harder than classical planning (De Giacomo & Vardi 1999). Intuitively, this is due to the fact that a plan can be associated with possibly many traces, and in order to guarantee goal achievement they all must be taken into account.

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Several approaches to planning in nondeterministic domains have been recently tackled (Pryor & Collins 1996; Kabanza, Barbeau, & St-Denis 1997; Weld, Anderson, & Smith 1998; Cimatti, Roveri, & Traverso 1998a; 1998b; Rintanen 1999). One of the most successful research directions recasts planning in nondeterministic domains as search in the space of beliefs (Bonet & Geffner 2000). A condition of uncertainty is represented as a *belief state*, i.e. a set collecting all possible states in which the domain could be. Since the belief space is exponential in the number of states in the domain, the problem turns out to be extremely complex. The use of symbolic techniques, based on Binary Decision Diagrams, allows to stretch the scalability in conformant planning, both for breadth-first search (Cimatti & Roveri 2000) and for heuristic search (Bertoli, Cimatti, & Roveri 2001b), and in conditional planning (Bertoli *et al.* 2001b; Bertoli, Cimatti, & Roveri 2001a).

The most evident limitation of these approaches is that, even in trivial cases, the algorithms may get lost in the search space. This often results in a blow-up in the computational resources needed to solve the problem, and in highly sub-optimal solutions. The main reason for this is in the heuristic functions used to drive the search, that do not provide enough information. In this paper, we tackle the problem of defining more effective heuristic functions for searching the space of beliefs. The basic intuition is that the *degree of knowledge* associated with a certain belief state must be explicitly taken into account. In particular, we propose a way of ranking belief states depending on their degree of knowledge with respect to a given set of boolean functions. We define a new conformant planning algorithm, based on the identification and solution of suitable *knowledge subgoals*, used as intermediate steps to drive the search. The solution of knowledge subgoals is based on the identification of *knowledge acquisition conditions*, i.e. subsets of the state space from where it is possible to perform knowledge acquisition actions. The effectiveness of the proposed approach is demonstrated by a thorough experimental evaluation. In some very small instances of the simpler problems, the new algorithm may be slightly less efficient due to the overhead of reasoning about knowledge subgoals and knowledge acquisition conditions. However, the significance of such an overhead decreases with the growth of the dimension of the problem. Moreover, in any other problems we tackled, the

new algorithm exhibits improved efficiency, and always produces plans of high quality.

This paper is structured as follows. First we provide some background on conformant planning as search in the belief space. Then, we discuss the ideas underlying the definition of more effective selection functions, and we describe the algorithm for conformant planning based on such ideas. In the following section, we discuss the implementation in the setting of Planning via Symbolic Model Checking, and we present an extensive experimental evaluation. In the final section, we draw some conclusions and we discuss some related and future work.

Conformant Planning as Search in Belief Space

We consider nondeterministic planning domains represented as finite state automata. This general model can be used for providing a semantics to languages such as PDDL and its nondeterministic extension, and AR (Giunchiglia, Kartha, & Lifschitz 1997), where parallel/concurrent actions have preconditions, conditional and uncertain effects.

Definition 1 (Planning Domain) A Planning Domain is a 4-tuple $\mathcal{D} = (\mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R})$, where \mathcal{P} is the (finite) set of atomic propositions, $\mathcal{S} \subseteq \text{Pow}(\mathcal{P})$ is the set of states, \mathcal{A} is the (finite) set of actions, and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation.

Intuitively, a proposition is in a state if and only if it holds in that state. In the following we assume that a planning domain \mathcal{D} is given. We use s , s' and s'' to denote states of \mathcal{D} , and α to denote actions. $\mathcal{R}(s, \alpha, s')$ holds iff when executing the action α in the state s the state s' is a possible outcome. We say that an action α is applicable in s iff there is at least one state s' such that $\mathcal{R}(s, \alpha, s')$ holds. We say that an action α is deterministic in s iff there is a unique state s' such that $\mathcal{R}(s, \alpha, s')$ holds. An action α has an uncertain outcome in s if there are at least two distinct states s' and s'' such that $\mathcal{R}(s, \alpha, s')$ and $\mathcal{R}(s, \alpha, s'')$ hold.

Consider for instance the simple navigation domain outlined in the rightmost square in Figure 1 where a robot can move in the four directions a 4x4 square. The example is very simple, and is used here only for explanatory purposes. The positions in the square can be represented by means of two fluents x and y , both ranging from 0 to 3. For this simple domain, the propositions are equalities between the variables and their possible values (e.g. $x = 0$). The black location in (1,2) is a hole that, if entered, will cause the robot to crash. This means, for instance, the GoWest action is not applicable in (0,2). When moving toward a wall, the robot does not move. For instance, if the robot performs a GoSouth action in location (0,0), it will remain in (0,0). The location in (2,1) is a slippery spot, that will make the robot move unpredictably sideways when performing an action in it. This introduces nondeterministic action effects. For instance, the effect of performing a GoWest action starting from state (2,1) may result in any of the states in $\{(3, 0), (3, 1), (3, 2)\}$.

In this paper we consider plans to be sequences of actions, i.e. elements of \mathcal{A}^* . We use ϵ for the 0-length plan, π and

ρ to denote plans, and $\pi; \rho$ for concatenation. Conformant planning is the problem of finding a plan that, if executed in any initial state in $\mathcal{I} \subseteq \mathcal{S}$, takes the domain into a set of states $\mathcal{G} \subseteq \mathcal{S}$, regardless of nondeterministic action effects. Following (Bonet & Geffner 2000), we model this problem as search in the space of *belief states*. A belief state is a set of states, intuitively expressing a condition of uncertainty, by collecting together all the states which are equally possible. The execution of actions is lifted from states to belief states by the following definition.

Definition 2 (Plan Applicability, Execution) An action α is applicable in a belief state $\emptyset \neq Bs \subseteq \mathcal{S}$ iff α is applicable in every state in Bs . If α is applicable in Bs , its execution $\text{Exec}[\alpha](Bs)$ is the set $\{s' \mid \mathcal{R}(s, \alpha, s'), \text{ with } s \in Bs\}$. The execution of a plan π in a belief state, written $\text{Exec}[\pi](Bs)$, is defined as follows.

$$\begin{aligned} \text{Exec}[\epsilon](Bs) &\doteq Bs \\ \text{Exec}[\pi](\emptyset) &\doteq \emptyset \\ \text{Exec}[\alpha; \pi](Bs) &\doteq \emptyset, \text{ if } \alpha \text{ is not applicable in } Bs \\ \text{Exec}[\alpha; \pi](Bs) &\doteq \text{Exec}[\pi](\text{Exec}[\alpha](Bs)), \text{ otherwise} \end{aligned}$$

We say that a plan is applicable in a belief state when its execution is not empty. For a conformant planning problem, solutions are applicable plans, for which all the final states must be goal states. Formally, the problem is defined as follows.

Definition 3 (Conformant Planning Problem) Let $\emptyset \neq \mathcal{I} \subseteq \mathcal{S}$ and $\emptyset \neq \mathcal{G} \subseteq \mathcal{S}$. The plan π is a conformant solution to the problem $\langle \mathcal{I}, \mathcal{G} \rangle$ iff $\emptyset \neq \text{Exec}[\pi](\mathcal{I}) \subseteq \mathcal{G}$.

In the example navigation domain, the problem is for the robot to go from the initial belief state $\{(0, 1), (0, 2)\}$ to the goal belief state $\{(3, 1), (3, 2)\}$. A possible solution to this problem is the plan (GoSouth; three times GoEast; GoSouth; GoNorth): the associated traces are depicted in the middle row in Figure 1. (The hole and the slippery spot are reported only in the first square, while in the following ones only the belief state is reported.) The reader can check that the associated plan is conformant: for all the possible associated traces, the plan never violates the applicability conditions of the attempted actions, and ends in a goal state. Notice that the first action is GoSouth in order to make sure that the preconditions of the following GoWest are met in all the states of the belief state. Notice also the last GoEast action: intuitively, the uncertainty in the position of the robot is increased by the nondeterministic effect of the action. The following GoSouth action is needed to regain some knowledge: the robot will result in (3,3) if it executes GoSouth both in (3,2) and in (3,3).

The problem of *finding* a conformant plan in this setting amounts to searching the belief space, and looking for a path from the initial belief state to the goal belief state. Figure 1 outlines (a subset of) the search space for the example, constructed forward, starting from the initial set of states toward the goal. An expansion step consists in trying all the applicable movement actions. Notice that different plans can result in the same belief state, and that cycles are possible. Notice also that, even for this simple example, several different plans are possible, since the three leftmost belief states are

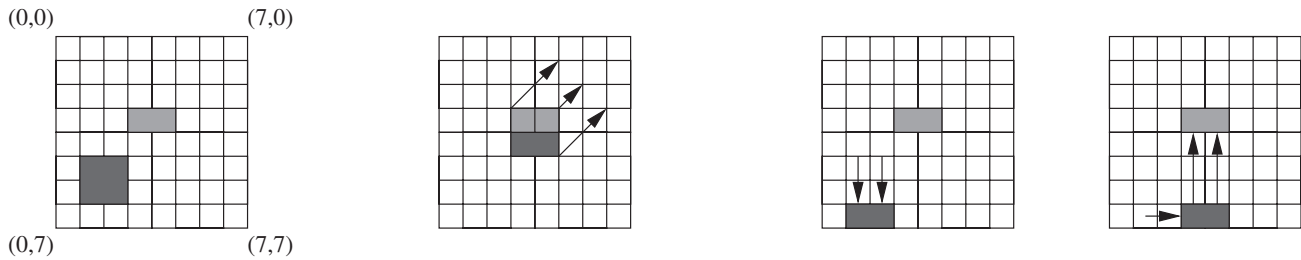


Figure 2: Different ways of searching the belief space

portant to be able to express the heuristic functions in non-boolean terms, in order not to lose the structure of the problem. A variable is known to have a value in a belief state if this is the case in all the states in the belief state. Boolean functions are treated similarly. The known range is the set of all possible values of a variable in a belief state. Notice how the cardinality of the known range gives a rough but valuable information on how uncertain we are on the variable. For instance, in the above case, the cardinality of the known range of the y variable in the initial belief state is 2, while in the goal belief state it is 1.

A more informed view of a boolean function in a belief state is the following.

Definition 5 (Truth Percentage) *The truth percentage of ϕ in S , written $|\mathcal{K}|_{\top}(\phi, S)$, is defined as $\frac{|\{s \in S | s \models \phi\}|}{|S|}$.*

The notion of truth percentage gives an idea of how close a belief state is to the truth of a boolean function. Again, this is clearly an abstract view, since there may be correlations that are lost. For instance, the two boolean functions $x = 0$, and $y = 0$ have the same truth percentage in the belief states $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ and $\{(0, 1), (1, 0)\}$. In some cases, truth percentage can be effectively used to reach a certain belief state, such as the goal, by “hill-climbing”, i.e. trying to greedily explore belief states of increasing truth percentage.

However, simple hill climbing on truth percentage is not enough in problems with more complicate structure. Therefore, we exploit the idea that a certain knowledge is necessary to reach the goal. Similar conditions, that we call “knowledge subgoals”, can be identified by taking an abstract view of the belief space. For instance, in the case of navigation in the empty room, it is possible to detect that there is no way in which the goal could be reached without passing through a belief state where the known range of y has cardinality 1. This kind of information, resulting in knowledge subgoals such as “the known range of x must have cardinality 1”, can be acquired by simple comparison of the known range of domain variables in the initial and goal belief states.

Our approach exploits this idea, and tries to acquire the necessary knowledge before trying to reach the goal. In order to solve knowledge subgoals, we preprocess the domain and identify “knowledge acquisition conditions”, i.e. belief

states where it is possible to execute an action that guarantees the solution of the knowledge subgoal. The motivation is that a knowledge acquisition condition may be easier to reach by greedy search than the associated knowledge subgoal. Formally, a knowledge acquisition condition is defined as follows.

Definition 6 (Knowledge acquisition condition) *The belief state Bs is a knowledge acquisition condition for the boolean function ϕ iff there exists an action such that $\mathcal{K}_{\top}(\phi, Exec[\alpha](Bs))$.*

For instance, the belief state defined by $x \in \{0, 1\}$ and $y \in \{0..7\}$ (corresponding to the width-two rectangle close to the left wall) is a knowledge acquisition condition for the function $x = 0$, since the application of a GoWest action in it guarantees that $x = 0$ is known in the resulting belief state. These notions introduce a way to partition the search in the belief space. Clearly, in this framework we have to face the problem of guessing the right knowledge subgoals, and of identifying whether reasonable knowledge acquisition conditions exists. In practice, we restrict knowledge subgoals to boolean functions of the form $x = v$. We discover the knowledge acquisition conditions associated with a knowledge subgoal by means of the conformant preimage primitive. The conformant preimage is a backward computation that, given a belief state Bs , returns all pairs $\langle \alpha_i . Bs_i \rangle$ such that $Exec[\alpha_i](Bs_i) \subseteq Bs$. From this analysis, sufficient conditions for detecting the unsolvability of a problem based on knowledge degree considerations can be derived. For instance, if we get rid of the walls to create a “circular” navigation domain, the problem of Figure 2 is clearly unsolvable. There is no way to reduce the cardinality of the known range of the variables, and it is therefore impossible to reach a more “informed” portion of the search space.

The Conformant Planning Algorithm

We devised an algorithm for conformant planning based on the ideas outlined in previous section. Figure 3 outlines the Finite State Machine of the control flow of the algorithm. The algorithm is entered by initializing the target knowledge, i.e. a set of knowledge goals that are estimated to be convenient intermediate steps to solve the given problem $\langle \mathcal{I} . \mathcal{G} \rangle$. If the initial belief state \mathcal{I} is such $\mathcal{K}_{\top}(\phi_i, \mathcal{I})$ for

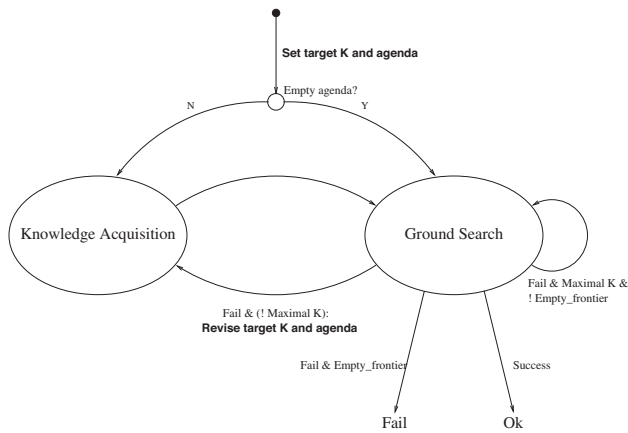


Figure 3: The high-level control flow of the algorithm

each knowledge subgoal ϕ_i in the target knowledge, then the algorithm enters the Ground Search mode, and greedily searches toward the final goal \mathcal{G} . Otherwise, the algorithm enters the Knowledge Acquisition mode, where it tries to reach the knowledge subgoals. In order to do so, the search considers a set of active knowledge subgoals, maintained in a subgoal agenda. The search greedily directs toward the knowledge acquisition conditions associated with the subgoals in the agenda, and then applies the suitable actions to achieve those subgoals. Once the Knowledge Acquisition mode terminates, the algorithm enters in the Ground Search mode, and tries to reach \mathcal{G} . Upon failure in the Ground Search mode, the target knowledge is refined, thus requiring the dynamic update of the knowledge subgoal agenda. When it is impossible to refine the target knowledge, and the list of open belief states is empty (i.e. all the state space has been covered), then the algorithm returns with failure. As other algorithms based on search in the beliefs space, the algorithm described above always terminates, returning a conformant plan if the problem is solvable, and failure otherwise. The conformance of plans is guaranteed by construction, while the termination follows from the complete storage of the traversed portion of search space, and on the finiteness of the belief space. To the best of our knowledge, the explicit identification of a dynamic knowledge agenda, and the revision of a target knowledge, are unique to this algorithm.

The greedy search strategy, applied in the Ground Search mode (while trying to reach the final goal) and in the Knowledge Acquisition mode (while trying to reach the knowledge acquisition conditions), is a forward expansion of the search space, similar to that described in (Bertoli, Cimatti, & Roveri 2001b). Basically, the currently selected belief state Bs is expanded by applying every action α_i that is applicable in it. The expansion returns the belief states of the form $Exec[\alpha](Bs)$, from which the belief states that have been previously generated are eliminated. Each belief state is also associated with a conformant plan for reaching it from \mathcal{I} . The newly generated belief states are merged into the open

list of belief states that deserve further expansion. The list of open belief states is ordered according to the following criteria (applied in decreasing priority):

- maintain previously reached knowledge subgoals;
- minimize the estimated distance to the local goal currently being pursued (i.e. the final goal in Ground Search mode, or a knowledge subgoal in Knowledge Acquisition mode);
- maximize the truth percentage of the local goal currently being pursued;
- in Knowledge Acquisition mode, try not to increase the distance from the final goal.

In order to reduce the cost of frontier analysis, but also to give more penetration to the search, more recent belief states are privileged: “old” sections of the frontier are considered only if the newly generated belief states all decrease current scoring function.

The implementation of the algorithm is based on this high-level control flow, together with a suitable definition of distances and intersections. In order to achieve efficiency, a number of additional refinements have been realized.

- In order to reduce the cost of identifying knowledge subgoals, we only consider a fixed set of simple knowledge subgoals, corresponding to the values for fluent variables of the problem. Though simple, for many domains this seems to be a reasonable choice.
- In order to reduce the cost of identifying knowledge acquisition conditions we adopt a lazy approach. Knowledge acquisition conditions are only searched for when a search at the knowledge level has been triggered. Moreover, we perform an approximation by associating each knowledge subgoal with the union of the knowledge acquisition conditions associated with it (rather than to each knowledge acquisition condition related to an action). Again, this simplification yields to significant speed ups without significantly worsening the level of information of the search.
- In order to reduce the cost of computing the estimate of distances (from a belief state to either the ground goal, or a knowledge acquisition condition), we adopted three ideas. First, we defined the distance $D(Bs, Bs')$ as the sum of distances $D_i(Bs, Bs')$, each based on the projection of the domain over the i -th fluent. Second, we defined $D_i(Bs, Bs')$ as the maximal distance between any state in Bs and any state in Bs' . Statewise distance is computed by approximated reachability. Third, we enforced laziness: distances are only computed if no immediate increasing of the truth percentage of the current local subgoal is possible. The first two ideas, coupled with the choice of knowledge subgoals presented in (a), allow to independently handle more than one knowledge subgoals at once, and to handle a simplified knowledge subgoal agenda representation. The third idea significantly reduces overheads in several cases.

Implementation and Experimental Results

We implemented the algorithm described in previous section in MBP (Bertoli *et al.* 2001a). MBP (Model Based Planner) is a general system for planning in nondeterministic domains, able to tackle conditional planning problems under the hypotheses of full and partial observability, conformant planning, and planning for extended goals. MBP is built on top of the NuSMV model checker (Cimatti *et al.* 1998), and is based on symbolic model checking techniques, Binary Decision Diagrams (Bryant 1992) in particular, to compactly represent and efficiently explore the domain description in form of a Finite State Automaton. Binary decision diagrams (BDDs) are a canonical form for the representation of boolean formulae, and can be used to represent sets of states and their transformations. The implementation of the conformant planning algorithms is based on the main contribution of (Bertoli, Cimatti, & Roveri 2001b), i.e. the integration of BDD-based techniques within a heuristic search framework. Basically, each belief state is represented by a BDD. Visited BDDs are stored in a hash table, and the approach takes full advantage of the ability of the CUDD (Somenzi 1997) package to compactly represent large numbers of BDDs. Furthermore, the expansion and pruning of belief states is carried out by means of logical transformations (corresponding to the application of boolean connectives and quantifications) that avoids the explicit enumeration of possible actions.

This approach provides an effective set of primitives that we also use in the implementation of the algorithm described in this paper. Further details, not reported here for lack of space, can be found in (Cimatti & Roveri 2000) and in (Bertoli, Cimatti, & Roveri 2001b). Here we focus only on the computation primitives that are more closely related to the algorithm described in this paper. In particular, the computation of the heuristic functions is based on BDD-based transformation. It is possible to evaluate $\mathcal{K}_\top(\phi, Bs)$ simply by checking whether the BDD representing Bs entails the BDD representing ϕ . Since the BDD for ϕ is the set of states satisfying ϕ , we are checking if the set of states in Bs is included in the set of states that satisfies ϕ . The truth percentage $|\mathcal{K}_\top(\phi, Bs)|$ is also computed by means of BDD manipulations. $Bs \cap \phi$ is computed as the conjunction of two BDDs. To compute $|Bs|$ and $|Bs \cap \phi|$, we use the primitive that counts the models of (the propositional formula represented by) a given BDD, the so-called “CountMinterms”. Although this operation may appear to be complex, the computation is efficiently carried out by means of a traversal on the structure of the BDD, and is quite feasible in practice. Finally, we estimate of the distance of a belief state Bs from a target belief state by means of a backward reachability analysis, that proceeds breadth-first from the target and computes the sets of states (layers) at increasing distance until Bs is completely included. Since this approach is in general inefficient, it is approximated by means of a relaxed reachability, where the transition relation of the domain is considered variable-wise, abstracting away the interactions among variables. The set of approximate transition relations is also obtained by means of projection operations.

In the following we call KACMBP (Knowledge Acquisi-

tion CMBP) the implementation of the algorithm described in this paper. We evaluated our approach by comparing it with the approach presented in (Bertoli, Cimatti, & Roveri 2001b), in the following referred to as HSCP (Heuristic Conformant MBP). Basically, HSCP performs a backward search, from the goal to the initial state, with a pure heuristic (where the cost of reaching the belief state under analysis is disregarded) that drives the exploration toward belief states of higher cardinality. We limit the comparison to these two systems, that share the same (efficient) symbolic machinery to isolate the differences in the heuristic component. A somewhat indirect comparison of KACMBP with other relevant systems such as GPT, CGP (Smith & Weld 1998), QBFPLAN (Rintanen 1999), and the breadth-first conformant planning algorithm of MBP (Cimatti & Roveri 2000), can be derived from (Cimatti & Roveri 2000) and (Bertoli, Cimatti, & Roveri 2001b), where such systems are tested against (and outperformed by) HSCP.

We carried out an extensive experimental evaluation, by covering the standard problems presented so far in the literature to evaluate conformant planners (i.e. BT, RING, SQUARE and CUBE), and by defining some new ones. (For lack of space, we refer to (Cimatti & Roveri 2000) for the description of the standard problems.) All the experiments were run on an Pentium II 300MHz with 512Mb of memory running Linux, fixing a memory limit of 128Mb and a CPU time limit of 1 hour. For each problem instances, we report both the CPU time (in seconds) required to find a solution for the given planning problem, and the length of the generated plan. Notice that the time scale is logarithmic. The automaton construction times are not reported, being exactly the same in the two approaches. For all the examples reported in this paper, however, they never require more than 0.5 seconds.

The results of the comparison on the classical domains are depicted in Figure 4. Each plot refers to a problem class. The darker track corresponds to (the computation times and plan lengths for) HSCP, while the lighter to KACMBP (resp. Bwd and H-Fwd in the figures). We see that KACMBP either behaves very much alike that of HSCP, or it scales much better. In particular, the algorithms behave very similarly for highly constrained problems, such as the RING and the corner and face version of the CUBE domain. In such cases, the simple best-first, greedy approach of HSCP results in a well-driven search which produces an optimal plan. However, already for those slightly more complex problems where knowledge acquisition must be either interleaved with or followed by some kind of ground-level search, such as BMT or the center version of the CUBE, KACMBP shows a better scale up in time.

We also considered several variations of a “circular” navigation domain (i.e. a cube with no walls), where no trivial way of refining the knowledge is available. First, we considered a simple repositioning problem where no knowledge acquisition is needed. The difference in performance results from the more directed ground level search of KACMBP.

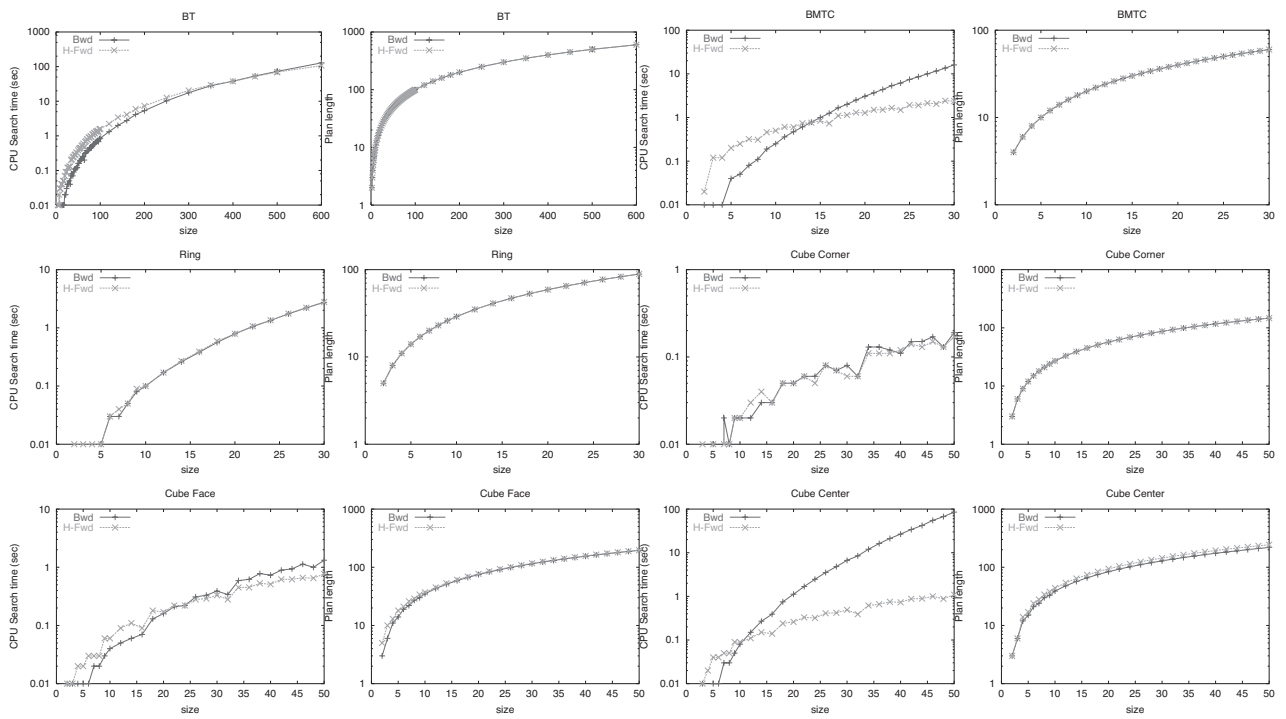
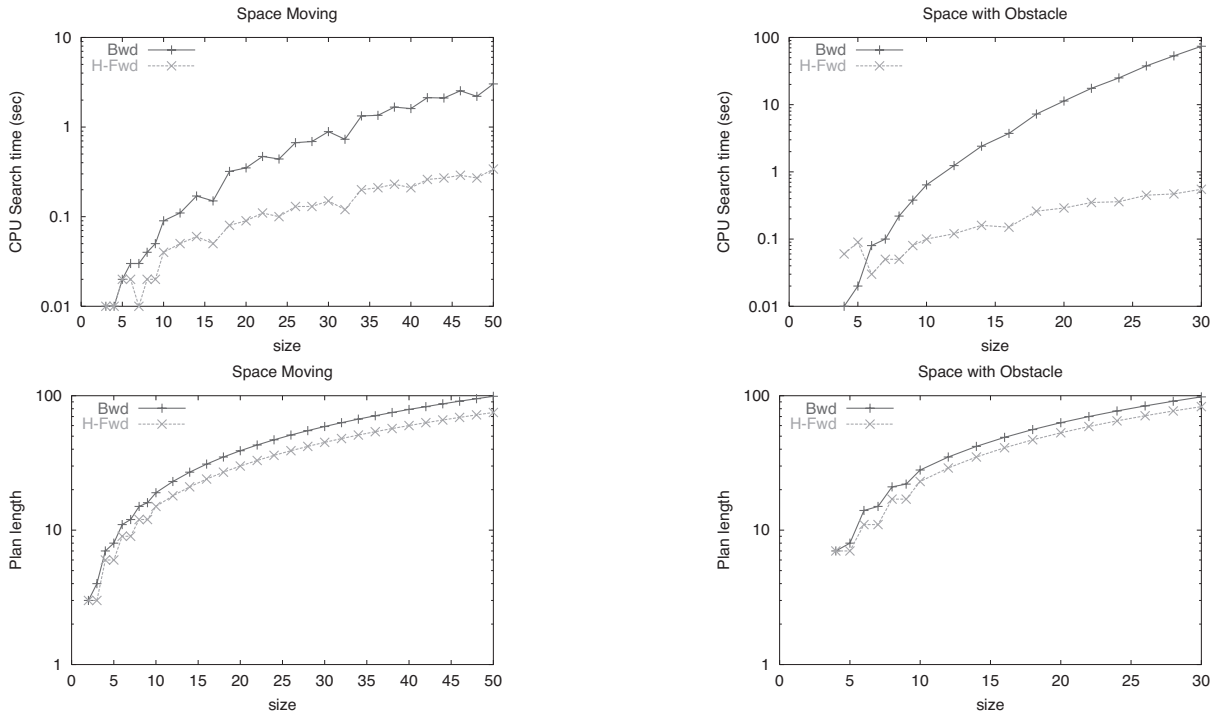


Figure 4: Experimental results on the BT, SQUARE, CUBE and RING domains



The table below reports the performance of the systems on a more difficult repositioning problem (Space with Obstacle), where only a column is available in the domain for repositioning the robot. For this problem, both knowledge acquisition and ground-level search turn out to be necessary.

Finally, in the Treasure Finding problem, the goal is to get hold of a treasure when incomplete information is available on the position. Notice how the simple structural heuristic of HSCP is unable to drive the search in the right direction.

