CPA(C)/(H): Two Approximation-Based Conformant Planners

Dang-Vien Tran, Hoang-Khoi Nguyen, Enrico Pontelli, Tran Cao Son
Computer Science Department
New Mexico State University
Las Cruces, NM 88011

Abstract

This paper describes two approximation-based planners, CPA(C) and CPA(H), which search for plans in the space of sets of partial states. Both planners are built on CPA+ (Son & Tu 2006) and employ several simplification techniques that reduce the size of states encountered duing the search process.

Introduction

CPA(C)/(H) deal with planning problems with uncertainty about the initial states and arbitrary state constraints. They can be classified as approximation-based planners, which search for solutions in the space of sets of partial states instead of the space of belief states (Son *et al.* 2005). The approach relies on the observation that a belief state can (sometimes) be replaced by the intersection of its members, thereby reducing the size of the search space *significantly*. The original CPA planner is incomplete (Son *et al.* 2005). Its subsequent version, CPA+ (Son & Tu 2006), addresses this issue by identifying the necessary knowledge in the initial set of partial states. The completeness condition does not consider state constraints though. Both CPA and CPA+ accept problems described as $\mathcal{AL}$-action theories. Roughly, $\mathcal{AL}$-action theories can represent planning problems in PDDL with arbitrary axioms.

The two systems CPA(C)/(H) are modifications of CPA+. CPA(C) uses best-first-search with a different heuristic function. CPA(H) employs local search using depth-first search. To conform with the planning competition rules, both include a PDDL-parser and can accept problems in PDDL-format. The systems also include a preprocessor, which implement two simplification techniques. Both techniques are aimed at reducing the size of the states that the planners need to deal with during the search.

The next section introduces the basic concepts used in the development of the planners and is followed by a description of the organization of the systems.

Basic Concepts

This section describes the idea of approximation-based planning, the simplification techniques, and the heuristics used in the development of CPA(C)/(H).

Approximation-Based Planning

The approach to approximation-based planning adopted in CPA(C)/(H) relies on the 0-approximation semantics for reasoning about effects of actions in presence of incomplete information about the initial state (Son & Baral 2001). Intuitively, the approach (*i*) replaces a belief state by a *partial state*, which is a set of fluent literals; and (*ii*) specifies how to compute the successor partial state, i.e., the result of executing an action in a given partial state. This is appealing for conformant planning since it lower the complexity of conformant planning (Baral, Kreinovich, & Trejo 2000). To guarantee completeness, an approximation-based conformant planner might need to search for solutions in the space of sets of partial states, called *cs-states*.

Analysis and Simplifications

The analysis and simplification techniques implemented in CPA(C)/(H) help simplify the planning instances by reducing the number of actions, propositions, and size of the initial cs-state. These techniques include:

- *Basic Simplifications*: We consider two well-known basic steps: *forward reachability* and *goal relevance*. Several planners implement these two steps.

  Forward reachability is used to detect *(i)* propositions whose truth value cannot be affected by the actions in the problem specification (w.r.t. the initial state); *(ii)* actions whose execution cannot be triggered w.r.t. the initial state. This process can be modeled as a fixpoint computation.

  Goal relevance proceeds in a similar manner, by detecting actions that are relevant to the achievement of the goal.

- *Combination of* `oneof`*-Clauses*: `oneof`-clauses are used to specify the uncertainty about some propositions and/or mutual exclusion between propositions. The number of the `oneof`-clauses and their size (the size of an `oneof`-clauses is the number of its elements) determine the size of the initial cs-state.

  The idea of the combination of `oneof`-clauses technique is based on the *non*-interaction between actions and propositions in different sub-problems of a conformant planning problem. This idea is best illustrated with a simple example.

Let consider the planning problem P with the set of propositions $\{f, g, h, p, i, j\}$, the initial state $I = \{\texttt{oneof}(f, g), \texttt{oneof}(h, p), \neg i, \neg j\}$, the set of actions $O = \{\ a : f \to i \quad c : h \to j \quad b : g \to i \quad d : p \to j\ \}$, and the goal $G = i \wedge j$.

Here, a causes i to be true if f is true; c causes j to be true if h is true; b causes i to be true if g is true; and d causes j to be true if p is true.

It is easy to see that the sequence $\alpha = [a, b, c, d]$ is a solution of P. Furthermore, the search should start from the cs-state consisting of the four states:

$$\begin{array}{ll} \{f, \neg g, h, \neg p, \neg i, \neg j\} & \{\neg f, g, h, \neg p, \neg i, \neg j\} \\ \{f, \neg g, \neg h, p, \neg i, \neg j\} & \{\neg f, g, \neg h, p, \neg i, \neg j\} \end{array}$$

Let P' be the problem obtained from P by replacing I with I', where $I' = \{\texttt{oneof}(f \wedge h, g \wedge p), \neg i, \neg j\}$.

We can see that α is also a solution of P'. Furthermore, each solution of P' is a solution of P. This transformation in interesting since the initial cs-state now consists only of two states: $\{f, \neg g, h, \neg p, \neg i, \neg j\}$ and $\{\neg f, g, \neg h, p, \neg i, \neg j\}$. In other words, the number of states in the initial belief state (or initial cs-state) that a conformant planner has to consider in P' is 2, while it is 4 in P. This transformation is possible because the set of actions that are "activated" by f and g is disjoint from the set of actions that are "activated" by h and p, i.e., $preact(\{f, g\}) \cap preact(\{h, p\}) = \emptyset$.

Using this technique, many `oneof`-clauses can be combined into one, yielding several order of magnitudes reduction in the size of the initial cs-state.

- *Goal Splitting*: The key idea is that if a problem P contains a subgoal whose truth value cannot be negated by the actions used to reach the other goals, then the problem can be decomposed into smaller problems with different goals, whose solutions can be combined to create a solution of the original problem. This technique can be seen as a variation of the goal ordering technique in (Hoffmann, Porteous, & Sebastia 2004) and relies on the notion of dependence proposed in (Son & Tu 2006).

Heuristics

The heuristics implemented in CPA(C)/(H) are combinations of the following well-known heuristics.

- *The cardinality heuristic:* we prefer cs-states that have a smaller cardinality. In other words, $h_{card}(\Sigma) = |\Sigma|$ where Σ is a cs-state. Note that we use this heuristic in a forward fashion, and hence, is different from its use in (Bertoli, Cimatti, & Roveri 2001; Bryce & Kambhampati 2004). The intuition behinds this is that planning with complete information is "easier" than planning with incomplete information and a lower cardinality implies a lower degree of uncertainty.
- *The total sum heuristic:* for a cs-state Σ, we define $h_{sum}(\Sigma) = \sum_{\delta \in \Sigma} d(\delta)$, where $d(\delta)$ is the well-known sum heuristic value of the problem given that the initial state is δ^* which is the completion of δ (Nguyen, Kambhampati, & Nigenda 2002).
- *The number of satisfied subgoals:* denoted by $h_{sub}(\Sigma)$.

We investigate two combinations: $h_{cs}(\Sigma) = (h_{card}(\Sigma), h_{sub}(\Sigma))$ and $h_{css}(\Sigma) = (h_{card}(\Sigma), h_{sub}(\Sigma), h_{sum}(\Sigma))$ with the lexical ordering applying on their components. CPA(C) uses h_{css} and CPA(H) uses h_{cs}.

System Organization

The proposed system is organized as in Fig. 1. The first component is a front-end, that acts as a *static analyzer*. The static analyzer is in charge of applying several simplifications and optimizations to the input problem specification—initially expressed in PDDL. The simplified specification (expressed either in PDDL or in the action language $\mathcal{AL}$—the native input format of CPA(C)/(H)) produced by the static analyzer is then fed to the actual planner. The separation of the two stages allows us to investigate the use of different planners applied to the same simplified problem specification.

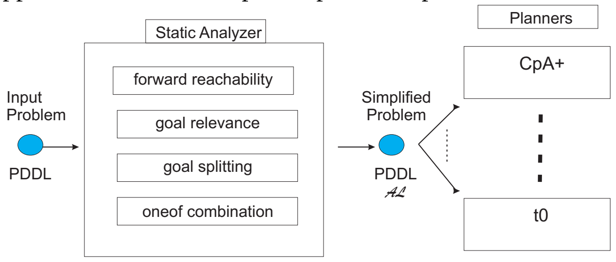

Figure 1: Overall System

The implementation of the static analyzer makes use of the PDDL parser originally developed for the CPAsystem; the parser has been modified to enable the construction of a Prolog representation of the problem specification. This Prolog representation is used as the input to the static analyzer, implemented in Prolog. The analyzer implements the forward/backward simplifications, the `oneof`-combination, and the goal-splitting algorithm. Its output is a sequence of simplified problems in $\mathcal{AL}$ which serve as input to the planners CPA(C)/(H). An option is also available to produce PDDL output from the static analyzer—that can be fed, for example, to a different planner.

The two planners, CPA(C)/(H), are implemented in C++. CPA(C) replaces CPA+'s heuristic function with h_{css} and makes use of a depth-first search algorithm. This search exhaustively explores all trajectories from the initial conditions to the goal. The h_{css} heuristic initially gives preference to the cs-states with a lower degree of uncertainty, i.e., cs-states that have a smaller cardinality. If the cardinality of two cs-states does not differ, then the heuristics gives preference to those cs-states that maximize the number of satisfied subgoal. Finally, if there are no differences, we compare the total sum heuristics of the cs-states and give preference to the ones with the smaller value. To measure the total sum heuristics, we compute the classical relaxed plan for each state and aggregate the relaxed plans.

CPA(H) makes use of h_{cs} in combination with a best-first search algorithm. Similarly to what described earlier, CPA(H) uses a combination of different heuristics to guide the search. The h_{cs} heuristic combines the cardinality heuristic and the number of satisfied subgoals heuristic. We evaluate the h_{cs} heuristics in a fashion analogous to the case of CPA(C), by first using the cardinality heuristic to discrminate between cs-states, and successively using the number of satisfied subgoals heuristic for refining the classification of cs-states.

Both planners employ an explicit representation of cs-states as sets of sets of propositions, and they make use of the C++ standard library `std` for sets manipulation. To reduce the space consumption, a partial state is created only once and it is shared by all cs-states containing it.

Conclusion and Future Work

We presented the main techniques implemented in the two conformant planners CPA(C) and CPA(H). Experimentally, these planners are competitive with state-of-the-art conformant planners in several benchmark domains. One of our main goals in the near future is to continue this line of research, to address the problems related to the number of actions in the planning problems. We would also like to investigate methods to improve the quality of the solutions.

References

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *AIJ* 122:241–267.

Bertoli, P.; Cimatti, A.; and Roveri, M. 2001. Heuristic search + symbolic model checking = efficient conformant planning. In Nebel, B., ed., *IJCAI*, 467–472. Morgan Kaufmann.

Bryce. D. 2006. POND: The Partially-Observable and Non-Deterministic Planner, Notes on The 5th IPC.

Bryce, D., and Kambhampati, S. 2004. Heuristic Guidance Measures for Conformant Planning. In *ICAPS*, 365–375.

Bryce, D.; Kambhampati, S.; and Smith, D. 2006. Planning Graph Heuristics for Belief Space Search. *JAIR* 26:35–99.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR*, 22:215–278.

Nguyen X.L.; Kambhampati, S.; Nigenda, R. 2002. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *AIJ*, 135, 73-123.

Son, T., and Baral, C. 2001. Formalizing sensing actions - a transition function based approach. *AIJ* 125(1-2):19–91.

Son, T. C., and Tu, P. H. 2006. On the Completeness of Approximation Based Reasoning and Planning in Action Theories with Incomplete Information. In *KRR*, 481–491.

Son, T. C.; Tu, P. H.; Gelfond, M.; and Morales, R. 2005. Conformant Planning for Domains with Constraints — A New Approach. In *AAAI*, 1211–1216.