Planning with Contingencies Via a Fast and Informed Action Selection Mechanism

Alexandre Albore
Departament de Tecnologia, Universitat Pompeu Fabra
Passeig de Circumval·lació, 8 - 08003 Barcelona, Spain
alexandre.albore@upf.edu

Advisor: Héctor Geffner
Departamento de Tecnología
ICREA & Universitat Pompeu Fabra
08003 Barcelona, SPAIN
hector.geffner@upf.edu

Motivations

Contingent planning is the task of finding a (conditional) solution plan under uncertainty and partially observable domains. This task is very challenging due to its very general nature. A growing number of contingent planners (Contingent-FF, MBP, Pond) have dealt with this task in the last few years, solving problems of growing size and complexity. However many planning problems involving partial information and limited sensing remain out of reach of these planners. The principal reason is that constructing or even verifying plans for such problems would take exponential time: contingent plans of exponential size follow naturally from situations where the number of observations that need to be done is linear in the size of the problem. In fact, current languages for contingent planning do not admit compact solutions representation. This implies that a lot of planning problems are practically unsolvable.

However in classical or even in conformant planning, plans of exponential length are the exception. So, the question of whether it is possible to identify and use an extension of a classical problem to represent a contingent planning problem arises. In that way, it can be possible to develop an efficient (even if incomplete) procedure capable of solving non trivial problems quickly.

Approach

The work presented here focuses on the development of a domain-independent planning technique for dealing with contingent problems. Rather than aiming at constructing full contingent plans, we focus on an action selection mechanism that chooses the action to do next in order to be effective in the plan execution. Practical applications usually do not need complete plan verification: in a reactive platform, all that is needed to be provided is the next-to-execute action in a rapid but goal-directed way. For this to work, this selection mechanism has to be both fast and informed.

We start from the consideration that a basic relaxation for contingent planning problems, that implies relaxing the delete effects of an action and moving preconditions in as conditions (Hoffmann & Brafman 2005), leads to a formulation that corresponds to conformant planning problems (a kind of problems equivalent to a version of contingent planning where no sensing action is available). A wide class of conformant problems can be mapped onto classical problems by using conditional and simple epistemic rules (Palacios & Geffner 2006; 2007). Even if not complete, this translation is sound and has been shown to be empirically quite effective (Bonet & Givan 2006). The final consideration is that classical planning problems are succesfully and efficiently handled by state-of-the-art classical planners. Thus, such planners can be used to solve a contingent problem mapped onto a classical problem following the line just described, only on condition of extending the classical planning framework to accomodate conditional effects of the actions being done.

The Translation

To describe the contingent planning problem P we use a language that is basically STRIPS with conditional effects. The problem P is a tuple $P = \langle F, O, \mathcal{I}, G \rangle$ where F stands for the fluent symbols in the problem, O stands for the set of actions or operators a, the initial situation $\mathcal{I}$ is a set of clauses over F, and G is a set of literals over F defining the goal. In addition, every action a has a precondition given by a set of fluent literals, and a set of conditional effects $C \rightarrow L$ where C is a set of fluent literals and L is a literal. All actions are assumed to be deterministic and hence all uncertainty lies into the initial situation.

We consider sensing actions, that are actions a characterized by having a single unconditional effect $obs(x)$, where x is a fluent symbol, meaning that after doing action a the truth value of x will be known. Such sensing actions can have preconditions like any other action.

The basic conformant translation The initial translation from a contingent planning problem to a conformant planning problem starts by removing the information gathering actions from the original contingent problem. To this conformant "fragment" of the original problem P, we apply a translation $K(P)$ that maps this problem into a classical planning problem, and that is defined in terms of two parameters: a set T of *tags* (i.e. a set of literals from the original problem whose truth value is not known in the initial situation $\mathcal{I}$) and a set R of *merges rules*.

So, in the classical planning problem $K(P)$, the literals assume the form of KL/t, capturing the conditional "L is known to be true if t is initially true", for literals $L \in P$ and tags $t \in T$. The merges $m_{J,L}$ are defined over pairs (J, L), where J is a collection of tags such that at least one must be true in the initial situation, and L is a literal in P s.t. $m_{J,L} : \bigwedge_{t \in J} KL/t \rightarrow KL$. The rest of the translation is based upon moving actions from L literals to actions onto KL tagged literals, and by adding deductive rules that allow solve a wide range of problems by accounting for a simple form of reasoning by cases (Palacios & Geffner 2007).

The translation is sound, polynomial, and for problems with limited conformant width [1], complete. These and potentially other conformant problems P can be solved by feeding $K(P)$ into a classical planner, and removing then the 'merge' actions in the resulting plan.

Execution and Heuristic Models

The action selection mechanism is based on two different representations of the planning problem: an *execution model* $\mathcal{X}$, obtained from the mapping shown in the previous section built up with sensing actions expressed as non-deterministic actions, and a *heuristic model* $\mathcal{H}$, where the sensing actions with non-deterministic effect are mapped into deterministic actions.

Execution Model The execution model $\mathcal{X}$ is built on top of the former translation of the conformant fragment into a classical problem.

In $\mathcal{X}$, the sensing actions of the original problem P are reintroduced and represented as non-deterministic actions with observable effects. For instance, observing a literal x results in an action of the form: $obs(x) : Kx \vee K\neg x$, where obs is the sensing action and Kx and $K\neg x$ are literals representing that the truth values possibly assumed by x are known to be respectively true and false with certainty. This brings the action $obs(x)$ to have two possible outcomes.

The translation $K(P)$ is extended with a set of deductive rules in the execution model, to permit reasoning about possible situations: while in conformant planning one reasons only forward along the timeline, in a contingent setting one must reason both forward and backward.

The set of deductive rules is encoded as actions with no preconditions and with those conditional effects as their only effects:

1. $KL/t \wedge K\neg L \rightarrow K\neg t$
2. $\bigwedge_{t \in J} (KL/t \vee K\neg t) \rightarrow KL$
3. $KL/t \wedge Kt \rightarrow KL$
4. $\bigwedge_{t,t' \in J; t \neq t'} K\neg t' \rightarrow Kt$

Information gathering actions can help in identifying the current state (but not changing it), which has repercussions over the knowledge of the initial situation.

[1]The conformant width of a problem measures the maximum number of non-unary clauses in the initial situation, all of whose literals are relevant to a given precondition or goal literal L.

Heuristics Model From the translation $K(P)$, we derive a heuristic model $\mathcal{H}$ by moving preconditions in as conditions and removing the deletes (Hoffmann & Brafman 2005). The sensing actions are also represented in a different way: in $\mathcal{H}$ we apply a relaxation to information gathering actions that lead to a "precondition relaxation" where both the possible results of an observation are taken into account for the following search of a solution plan. Thus non-deterministic sensing actions are relaxed as normal *deterministic actions*. So, given an observable literal x and a sensing action *obs*, we have: $obs(x) : Mx \wedge M\neg x$, where Mx and $M\neg x$ are literals representing "contingent knowledge", i.e. "x has been observed and its value is possibly known". This brings the action $obs(x)$ to have one deterministic outcome in $\mathcal{H}$. This approach is more informed than the delete relaxation (Hoffmann & Brafman 2005) because the action preconditions are not ignored, but only relaxed in order to reason about sensing actions consequences. In order to provide a classical planner with informed heuristics, we enrich $\mathcal{H}$ with deductive rules, similar to the ones present in $\mathcal{X}$, that allow us to capture inferences that are critical in the contingent setting but seldom needed in conformant planning.

There are two reasons that explain why the resulting heuristic model $\mathcal{H}$, which is a classical planning problem, can provide a useful heuristic criterion for selecting actions in the contingent planning problem P. First, if action preconditions in P are ignored, then the result in the delete relaxation is a conformant problem (Hoffmann & Brafman 2005), whose classical translation is the precondition and delete-free version of $\mathcal{H}$. Second, $\mathcal{H}$ does not ignore the action preconditions in P, it just relaxes them in terms of the M-literals, and reasons about the sensing actions from which such relaxed preconditions can be achieved.

Implementation and Preliminary Results

We implemented the Closed-Loop Greedy planner basing ourselves on the above execution model $\mathcal{X}$ and the heuristic model $\mathcal{H}$, relying on the classical FF planner (Hoffmann & Nebel 2001) to provide the heuristic-based next-to-execute action. Given the current state s, first $\mathcal{X}$ then $\mathcal{H}$ deductively close it by applying all the deductive rules to end up in a state s_h, which includes both KL/t and ML literals. Then a modified version of FF is called upon $\mathcal{H}$ on s_h and returns an improving action sequence π, i.e. a sequence of actions that ends in a state with a heuristic value lesser than s_h. After that, the actions in π are applied in the execution model. If π ends or a sensing action is performed, deductive rules are applied to the resulting state to obtain a new state s_y. If s_y is a goal state, the loop ends succesfully, otherwise the process continues from s_y as initial state.

If a full contingent plan is desired, all possibilities must be tried; recording then the action sequences leading to the goal along each possible observation sequence.

The Closed-Loop Greedy action selection mechanism has been tested over various benchmarks. Here we show and discuss results over a single benchmark, called *colorballs*, which is quite representative. *colorballs-n-x* is the problem of collecting x colored balls in a $n \times n$ grid. Balls are colored

	Contingent FF		**CLG**			
problem	**time (s)**	**nacts**	**t0 time (s)**	**pddl size (Mb)**	**time (s)**	**nacts**
colorballs-3-1	0,02	95	0,04	0,24	0,12	95
colorballs-3-2	1,96	2781	0,08	0,46	3,51	2445
colorballs-3-3	84	> 75000	0,11	0,68	137,63	53836
colorballs-4-1	0,27	277	0,14	0,70	0,58	281
colorballs-4-2	36,33	18739	0,27	1,35	39,72	18232
colorballs-4-3	$> 30mn$		0,41	2,0	$> 30mn$	
colorballs-5-1	1,83	611	0,44	1,98	2,43	584
colorballs-5-2	867,28	71157	0,82	3,89	307,4	67945
colorballs-5-3	$> 30mn$		1,28	5,79	$> 30mn$	
colorballs-6-1	7,43	1091	1,17	5,01	9,48	1021
colorballs-6-2	$> 30mn$		2,19	9,91	$> 30mn$	
colorballs-7-1	42,03	1826	2,83	11,38	30,88	1614
colorballs-7-2	$> 30mn$		5,21	22,60	$> 30mn$	
colorballs-8-1	$> 30mn$		6,02	23,62	95,73	2397
colorballs-9-1	$> 30mn$		12,78	45,53	256,59	3384
colorballs-9-2	$> 30mn$		23,58	90,79	$> 1.8Gb$	

Table 1: Solution times for Contingent-FF and CLG over colorball domain for a complete contingent plan. 'nacts' stands for the total number of actions on the tree solution. 't0 time' is the translation time. 'time' is time spent in the modified FF.

of one of four colors and each one of them has to be disposed in the corner that matches its color. The locations and colors of the balls are initially not known but can be observed when agent and ball are in the same cell.

We compare our approach with the state-of-the-art contingent planner Contingent-FF, run both with and without the helpful actions pruning mechanism (Hoffmann & Brafman 2005). The experiments are obtained on a Linux machine running at 2.33 Ghz with 8Gb of RAM, with a cutoff of 30mn/1.8Gb of memory. In all the cases of the above table 1, we have used the Closed-Loop Greedy planner to generate complete contingent plans. From this first experiments, we see that the Closed-Loop Greedy planner performs in a comparable way with Contingent FF, at least in benchmarks of lesser size. When the dimension of the problem increases, the amount of possible contingencies make the generation of a full solution exponential in the solutions length. However, the compilation of the contingent problem makes our planner able to deal with larger instances of the problem, outside the reach of the other planner. For huge instances of the planning problem, the size of the compiled domain is a major bottleneck that strongly affects scalability. This can be noticed in instance *colorballs*-6-2, for example.

Yet a main motivation for this work has been to have a fast but informed Closed-Loop planner that can scale up to problems in which the contingent solutions have exponential size. For testing this we ran 25 executions, selecting the sensing results randomly, and found all these executions leading to the goal in instances for which no full contingent plans could be computed within the time and memory cutoff. These include instances like *colorballs*-9-2 and 7-4.

Conclusions and Future Work

We have developed a domain-independent action selection mechanism for planning with sensing and have tested it empirically over a number of problems, showing that it compares well with state-of-the-art contingent planners, in particular scaling up well on generating successful executions without generating a full solution.

As future work, we plan to improve the implementation, clean up the formulation by incorporating axioms or ramifications in the target language of the translation, and redefine the 'enforced hill climbing' (EHC) step that selects the action sequence to apply next, so that the non-deterministic execution model $\mathcal{X}$ would be used within the EHC state progression, leaving the deterministic heuristic model $\mathcal{H}$ for computing the heuristic only. This is needed for ruling out the possibility of loops during the execution and reduce the compiled problem size, which is one of our main preoccupations. Moreover, the potential coverage of the deductive rules is also to be investigated, including which combinations of them applies best from completeness and effectiveness points of view.

References

Bonet, B., and Givan, B. 2006. Results of the conformant track of the 5^{th} Int. Planning Competition. At www.ldc.usb.ve/∼bonet/ipc5/docs/results-conformant.pdf.

Hoffmann, J., and Brafman, R. 2005. Contingent Planning via Heuristic Forward Search with Implicit Belief States. In *Proc. 15th Int. Conf. on Automated Planning and Scheduling (ICAPS-05)*, 71–80. AAAI.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research* 14:253–302.

Palacios, H., and Geffner, H. 2006. Compiling Uncertainty Away: Solving Conformant Planning Problems using a Classical Planner (sometimes). In *Proc. of the 21th National conference on Artificial Intelligence (AAAI-06).*

Palacios, H., and Geffner, H. 2007. From Conformant into Classical Planning: Efficient Translations That May be Complete Too. In *Proc. of the 17th Int. Conf. on Planning and Scheduling (ICAPS-07).*