Planning with Sensing: Scaling Up

Hector Palacios
Departamento de Tecnología
Universitat Pompeu Fabra
08003 Barcelona, SPAIN
hector.palacios@upf.edu

Alexandre Albore
Departamento de Tecnología
Universitat Pompeu Fabra
08003 Barcelona, SPAIN
alexandre.albore@upf.edu

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

Abstract

We develop a domain-independent action selection mechanism for planning with sensing that does not build full contingent plans but just chooses the action to do next in a closed-loop fashion. For this to work, the mechanism must be fast and informed. We take advantage of recent ideas that allow delete and precondition-free contingent problems to be converted into conformant problems, and conformant problems into classical ones, for mapping the action selection problem in contingent planning to an action selection problem in classical planning. The formulation is then tested over standard benchmarks and over contingent problems with exponential size plans. The proposed action selection mechanism is shown to scale up in both settings, producing behaviors that lead to the goal in spite of the lack of explicit contingent plans.

Introduction

Contingent planning is concerned with the problem of achieving goals in the presence of incomplete information and sensing actions (Peot and Smith 1992; Pryor and Collins 1996). This is one of the most general problems considered in the area of planning and one of the hardest (Haslum and Jonsson 1999; Rintanen 2004). In the last few years, significant progress has been achieved resulting in a variety of contingent planners that can solve large and non-trivial problems, usually by casting the contingent planning problem as a search problem in belief space (Bonet and Geffner 2000).

In spite of this progress, however, a large obstacle remains: many problems involving incomplete information and sensing actions have solutions of exponential size. This is different than in classical or conformant planning where exponential length solutions are the exception. Contingent plans of exponential size follow naturally from situations where the number of observations required is linear in the size of the problem.[1]

The goal of this work is to use domain-independent planning techniques for dealing with such problems. However, rather than aiming at constructing full contingent plans, we aim at an effective *action selection mechanism* that chooses the action to do next in closed-loop fashion. For this to work, the action selection mechanism must be fast and informed. Indeed, while it is not possible to consider *explicitly* all possible combination of contingencies that may arise, such contingencies cannot be ignored.

An action selection mechanism that *implicitly* takes all contingencies into account, is the greedy policy $\pi_h(b)$ obtained with a sufficiently informed heuristic function $h(b)$ over the beliefs b (Bonet and Geffner 2000). Indeed, for suitable heuristic functions (e.g., the optimal value function), the resulting policies are optimal. In practice, however, this approach faces two problems: the difficulty of getting fast and informed heuristics over belief space, and the time and space complexity of carrying out the beliefs. Recent proposals such as (Hoffmann and Brafman 2005; Bertoli et al. 2006; Bryce, Kambhampati, and Smith 2006) address these issues, and show that through the use of OBDD representations and SAT solvers, the worst-case intractability of belief updates is not always a practical impediment. On the other hand, the problem of getting informed heuristics in belief space remains challenging.

A recent approach that tackles the effective representation of beliefs and the derivation of informative heuristics in the simpler setting of conformant planning is (Palacios and Geffner 2007), where conformant problems P are automatically converted into classical planning problems $K(P)$ that are solved by an off-the-shelf classical planner. The states s of the problem $K(P)$ provide a compact (but incomplete) representation of the belief state b in P by relying on literals of the form KL and KL/t that represent that L is true with certainty, and that L is true if t was initially true respectively. In this shift to the *knowledge level* the states s in $K(P)$ encode beliefs about P that can be updated efficiently (as in ADL planning), and the heuristics required over these beliefs are classical planning heuristics. The difference with (Petrick and Bacchus 2002) is that the translation is done automatically and results in problems that can be solved by a classical planner.

The work in this paper is an extension of this translation-based approach to planning with sensing. As we will see, a contingent planning problem P can be converted into a suitable planning problem $X(P)$ at the knowledge-level whose states represent belief states over P. However, $X(P)$ is

[1] Compact solutions to these problems are often possible in languages closer to the ones used in programming that accommodate loops and subroutines, yet planning with such constructs appears to be very hard (as hard as automatic programming), and as shown here, not always necessary.

no longer a classical planning problem, and hence, cannot be solved by a classical planner. Rather by virtue of the *sensing actions* $obs(L)$ in P where L is an atom, $X(P)$ contains *non-deterministic effects* of the form $KL|K\neg L$, meaning that after sensing L, either KL or $K\neg L$ becomes true. The problem $X(P)$ is thus a *fully observable but non-deterministic planning problem*, whose solutions represent solutions to the *contingent planning problem* P. In contrast, P is a non-deterministic planning problem but on belief space (Bonet and Geffner 2000).

For solving the knowledge-level characterization $X(P)$ of the contingent problem P, we exploit a result in (Hoffmann and Brafman 2005) that states that the relaxation obtained from a contingent planning problem P that results from removing 'deletes' and 'moving the preconditions in as conditions', is actually a conformant planning problem P' where the sensing actions are not needed. We then construct a *classical planning problem* $H(P)$, by transforming P' into $K(P')$, and strengthening $K(P')$ with new literals ML that express 'contingent knowledge' (i.e., that L is known in *some* branch). Literals ML are used to encode the sensing actions $obs(L)$ as *deterministic actions* with effects ML *and* $M\neg L$, and the preconditions L of the actions a in P as ML. A classical plan for $H(P)$ thus represents a conformant plan for the relaxation P' of P that uses the sensing actions to establish the preconditions ML of the actions in the plan. Classical heuristics for $H(P)$ hence estimate the length of such plans, that implicitly consider all the contingencies (it's conformant) while taking sensing actions into account (they are required to establish the ML preconditions).

With these transformations performed as preprocessing, the *action selection problem in planning with sensing* is mapped into an *action selection problem in classical planning*. The resulting *Closed-Loop Greedy planner (CLG)* works then as follows: in the current state s, the *heuristic model* $H(P)$ is used to select the next actions which are then executed in the *execution model* $X(P)$, resulting in a new state. This process iterates until the goal is reached. We show that this planner scales up well: constructing contingent plans faster than other planners over a broader range of scenarios, and producing also meaningful executions in problems where the construction of full contingent plans is not feasible.

The rest of the paper is organized as follows: we present in order the contingent problem P, the translation $K(P')$ of the conformant fragment of P' (no sensing actions), the execution and heuristic models $X(P)$ and $H(P)$, and the resulting CLG planner. We present then examples, experimental results, and a summary.

An early version of this paper was presented at a national conference (Spain). While the same general framework is maintained, there are several differences in detail that account for a large jump in performance, including order-of-magnitude speedups, and many more problems solved.

Contingent Planning Problem P

We consider a planning language that extends Strips with conditional effects, negation, an uncertain initial situation, and sensing actions. More precisely, a contingent planning problem is a tuple $P = \langle F, O, I, G\rangle$ where F stands for the fluent symbols in the problem, O stands for the set of actions or operators a, I is a set of *clauses* over F defining the initial situation, and G is a set of literals over F defining the goal.

A normal action a has preconditions 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. The sensing actions a, on the other hand, are assumed to have a single unconditional effect $obs(L)$ where L is a literal, meaning that after doing action a the truth value of L is known. Sensing actions can have preconditions as any other actions but for simplicity we assume that they have no other effects.

We refer to the conditional effects $C \rightarrow L$ of an action a as the *rules* associated with a, and sometimes write them as $a : C \rightarrow L$ to tag them with the action name, or as $C \rightarrow L_1 \wedge L_2$ to aggregate several effects with the same condition. In all cases, C may be empty. Finally, we take $\neg L$ to denote the complement of L.

We refer to the fragment of P with the sensing action removed, as the *conformant fragment* of P. The delete relaxation of this fragment is defined as in classical planning with negation and conditional effects, assuming that negation has been compiled away, introducing for every positive literal L, another positive literal $\bar{L}$ that stands for its negation. We remove then all the rules $a : C \rightarrow \neg L'$ with negated literals in their heads, that we call 'deletes'.

Conformant Translation $K(P)$

Recently it has been shown that conformant problems P can be compiled into classical planning problems $K(P)$ often quite effectively. The translation $K(P) = K_{T,M}(P)$ in (Palacios and Geffner 2007) involves two parameters: a set of *tags* T and a set of *merges* M, split into sets M_L for each literal L in P. A tag t is set of literals in P whose status in the initial situation I is not known. A merge $m \in M_L$ in turn is a collection of tags $t_1, \ldots, t_m$ one of which must be true in I ($t_1 \vee \cdots \vee t_m$ holds in I).

The fluents in the translation $K_{T,M}(P)$, for $P = \langle F, O, I, G\rangle$ are of the form KL/t for each $L \in F$ and $t \in T$, meaning that 'it is known that if t is true in the initial situation, L must be true'. In addition, $K_{T,M}(P)$ includes extra actions, called **merge actions**, that allow the derivation of a literal KL (i.e. KL/t with the empty tag t) when KL/t' has been obtained for each tag t' in a merge $m \in M_L$. More precisely, if $P = \langle F, O, I, G\rangle$ is the conformant problem, the classical problem $K_{T,M}(P) = \langle F', I', O', G'\rangle$ is:

$$
\begin{aligned}
F' =& \{KL/t, K\neg L/t \mid L \in F \text{ and } t \in T\}\\
I' =& \{KL/t \mid \text{if } I \models t \supset L\}\\
G' =& \{KL \mid L \in G\}\\
O' =& \{a : KC/t \rightarrow KL/t,\ a : \neg K\neg C/t \rightarrow \neg K\neg L/t\\
&\quad \mid a : C \rightarrow L \text{ in } P\} \cup \{\bigwedge_{t\in m} KL/t \rightarrow KL \mid m \in M_L\}
\end{aligned}
$$

with the preconditions of the actions a in $K_{T,M}(P)$ including the literal KL if the preconditions of a in P include the literal L.

The translation $K_{T,M}(P)$ is sound, meaning that the classical plans that solve $K_{T,M}(P)$ yield valid conformant plans for P (by just dropping the merge actions). On the other hand, the complexity and completeness of the translation depend on the choice of tags and merges T and M. The $K_i(P)$ translation, where i is a non-negative integer, is an special case of the $K_{T,M}$ translation where the tags t are restricted to contain at most i literals; it is exponential in i and complete for problems with *conformant width* less than or equal to i. In the CLG planner, we use the $K_i(P)$ translation for $i = 1$, which in the conformant setting turns out to be complete for a wide range of problems. In $K_1(P)$, tags t are single literals. From now on, $K(P)$ will denote $K_1(P)$ unless stated otherwise.

Execution Model $X(P)$

The execution model $X(P)$ of the CLG planner is the union of the translation of the *conformant fragment* of P and two other components: a suitable encoding of the *sensing actions*, expressed in the language of the epistemic conditionals K/t, and a collection of *deductive rules*, expressed as actions.

The sensing actions $a : obs(L)$ in the contingent problem P become in $X(P)$ the *non-deterministic actions*

$$a : \neg KL \wedge \neg K\neg L \;\rightarrow\; KL \,|\, K\neg L\,.$$

These are *physical actions* at the *knowledge level* that can result in a transition from $\neg KL$ and $\neg K\neg L$ into either KL or $K\neg L$. If either L or $\neg L$ is known before the action, the action has no effect.

Additional deductive rules are needed in $X(P)$ as the tags t that are initially unknown may become known due to *observations*. These deductive rules encoded as actions with single conditional effects are:

1. **Full Merge:** $\bigwedge_{t\in m}(KL/t \vee K\neg t) \;\rightarrow\; KL$
2. **Tag Refutation:** $KL/t \wedge K\neg L \;\rightarrow\; K\neg t$
3. **Tag Confirmed:** $KL/t \wedge Kt \;\rightarrow\; KL$
4. **Static OR's:** $\bigwedge_{L'\in D, L'\neq L} K\neg L' \rightarrow KL$

The meaning of the these rules is direct, yet let us comment on two of them. **Full Merge** is a generalization of the **Merge** actions in the conformant translation, where every conjunct KL/t is replaced by the disjunction $KL/t \vee K\neg t$. This is because in the contingent setting, a tag may be disconfirmed, thus effectively reducing the collection of tags t in a merge m. The **Static OR** rule, on the other hand, uses static clauses $D : L_1 \vee \cdots \vee L_n$, in order to derive that a literal L_i must be true when the others are known to be false (Palacios and Geffner 2006). A clause $L_1 \vee \cdots \vee L_n$ is static when it is an invariant, i.e., it is true in the initial situation and for every effect $a : C \rightarrow \neg L_i$ that deletes a disjunct L_i, there is another effect with the same body $a : C \rightarrow L_k$ that adds another disjunct L_k.

The use of the rules above is combined with a simple transformation that makes all *tag literals static*. If $t = L$ is not a static literal (i.e., can be added or deleted by some action), then we create a *static copy* L_0 of L and add the equivalence $L_0 \equiv L$ to I. Thus, the literal L_0, that has the same value as L in the initial situation but does not change, represents the initial value of L. The tags are then limited to such static literals only. Thus, when Kt and $K\neg t$ are used in these rules, t refers to L_0 and not to L.

The effect of the deductive rules is to remove some literals $\neg KL$ from the state while replacing them by literals KL. Although these rules are encoded as actions in $X(P)$, we treat them in a special way: every time that the state s changes, we close the new state by applying all these 'actions' until no more literals KL can be derived. Since the effect of the rules is monotonic (they can only add knowledge literals KL, never remove them), this closure operation is well-defined.

The execution model $X(P)$ provides a sound but incomplete representation of the contingent planning problem P at the knowledge level, meaning that a solution to the fully-observable but non-deterministic planning problem $X(P)$ provides a solution to the contingent planning problem P, but not the other way around. Recall that a solution in both cases is the solution to an AND/OR graphs where OR gates represent actions, and AND gates represent non-determinism (sensing in P and non-deterministic 'physical' actions in $X(P)$).

Heuristic Model $H(P)$

A simple way to get an heuristic for solving the model $X(P)$ is to consider the *delete and precondition relaxation* of P. In (Hoffmann and Brafman 2005) it is shown that the delete-relaxation of the contingent planning problem P extended by moving in the preconditions of actions as conditions, leads to a problem P' that is conformant, where the sensing actions in P can be dropped. This relaxation P' can then be compiled into a classical planning problem $K(P')$ following the transformation above. Heuristics for the classical planning problem $K(P')$ are easy to obtain and can be used to guide the search in $X(P)$. The main limitation of heuristics obtained in this way, however, is that they ignore the sensing actions. Hoffmann and Brafman get around this limitation by relaxing the preconditions, but only when "possible and needed". In our formulation, we approach this limitation in a different way: we relax all preconditions in P by moving them in as conditions, but add new literals ML to the language of $H(P)$ that express 'contingent knowledge' (that L is known in some branch), that are used for encoding sensing effects and action preconditions. The result is that the classical plans for $H(P)$ represent conformant plans for the 'delete and precondition relaxation' P' where sensing actions are required for achieving these ML preconditions.

The sensing actions $a : obs(L)$ in P, which in the execution model $X(P)$ are encoded as *non-deterministic actions*

$$a : \neg KL \wedge \neg K\neg L \;\rightarrow\; KL \,|\, K\neg L$$

are encoded in the heuristic model $H(P)$ as *deterministic actions* of the form:

$$a : \neg KL \wedge \neg K\neg L \;\rightarrow\; ML \wedge M\neg L \wedge\; o(L)$$

meaning intuitively that after observing L, if neither KL nor $K\neg L$ is known, one can predict that both L *and* $\neg L$ will be

known in some branch of the execution. The literal $o(L)$, which is initially false, expresses that L has been observed, playing a role in one of the deductive rules below. Literals L that appear in sensing actions are called *observables*.

The basis of the heuristic model $H(P)$ is the classical planning model $K(P')$, obtained from the translation of the delete and precondition relaxation P' of P with the sensing actions dropped. The encoding of the sensing actions above, along with ML preconditions and the deductive rules below, are then added to this translation.

Thus, if a is an action with precondition L in the contingent planning problem P, a will be an action with precondition KL in the execution model $X(P)$, and an action with precondition ML in the heuristic model $H(P)$. In the latter case, the precondition L of the action a in P is pushed in as a condition to yield effects $a : L, C_i \rightarrow L_i$ in P' for the effects $a : C_i \rightarrow L_i$ in P. By pushing the preconditions literals L in as conditions in P', it is ensured that 'wishful thinking' done over *action preconditions* in $H(P)$ does not translate into 'wishful thinking' about their *effects* (which still need L). At the same time, the use of the M-literals as preconditions in $H(P)$ ensures that sensing actions remain relevant.

Deductive rules, similar to the ones for K, allow us also to expand the collection of literals L that are assumed to be 'contingently known'. These rules are:

1. **Weak K:** $KL \rightarrow ML$
2. **M-Tag Refutation** $KL/t \wedge o(L) \rightarrow M\neg t$
3. **M-Tag Confirmed** $KL/t \wedge Mt \rightarrow ML$
4. **M-Tag Disjunction** $\bigwedge_{t' \in m, t' \neq t} M\neg t' \rightarrow Mt$

These rules are a weaker version of the rules for K-literals. The choice of the rules is mostly heuristic. For example, rather than adapting the **Static-OR** rule from K-literals to M-literals, we have found more convenient to adopt a special case, **Tag Disjunctions**, which is the Static-OR rule only for literals that are tags. In addition, in **Tag Refutation**, we have chosen to use $o(L)$ rather than $M\neg L$. The literal $o(L)$, that just means that L has been observed, implies $M\neg L$, but not the other way around. The rationale for this choice has to do with issues that arise in causal reasoning (Pearl 1988), where care needs to be taken so that a causal prediction ML is not taken as (diagnostic) evidence for an hypothesis t (Pearl's favorite example is that while it is legitimate to conclude that it rained, given the *observation* that the grass is wet, it is not legitimate to draw the same conclusion if we have made the grass wet by opening the sprinkler).

On the other hand, the only K-rule that is needed is the **full merge**

$$\bigwedge_{t \in m} (KL/t \vee K\neg t) \;\rightarrow\; KL$$

This is because $H(P)$ represents a conformant problem ($K(P')$) strengthened with the ML preconditions. Thus, in principle, only the merge rule of $K(P')$ is needed. However, since during the executions some tags t may be discovered to be false, the collection of tags t in merges m must be adjusted dynamically. This is actually what the full-merge does by replacing the literals KL/t by the disjunctions $KL/t \vee K\neg t$: the literal $K\neg t$ in these disjunctions is true *only* when it is true in the current state. Indeed, no plan in $H(P)$ can achieve $K\neg t$ from a state s unless $K\neg t$ is already true in s.

Finally, rules $a : MC \rightarrow ML$ are added to $H(P)$ for the M-literals, in correspondence with the analogous rules for the K-literals.

While the semantics of the execution model $X(P)$ is that it represents the contingent planning P at the knowledge-level and can be shown to be correct in this sense (solutions to $X(P)$ are solutions to P), the semantics of the heuristic model $H(P)$, as the semantics of relaxations in general, is a bit more subtle. As we said, classical plans for $H(P)$ represent conformant plans for the delete and precondition relaxation P', where the preconditions ML of the actions must be achieved by sensing and deduction. Due to their conformant nature, the plans for $H(P)$ take into account all possible 'contingencies'. Thus, single executions in $X(P)$ that lead to the goal do not necessarily provide action sequences that are plans for $H(P)$. On the other hand, the combination of the actions over all the possible executions of $X(P)$ (what can be deemed as the 'contingent plan' for $X(P)$), will normally provide such plans, although the conditions under which this is true need to be established and proved.

Action Selection and the CLG Planner

The action selection cycle in the *Closed-Loop Greedy planner CLG* relies on a modified version of the classical FF planner (Hoffmann and Nebel 2001) that uses the (classical) heuristic model $H(P)$ for computing the heuristics and selecting the helpful actions, and the (non-deterministic and 'true') execution model $X(P)$ for doing state progression. The modified FF accepts a single PDDL file where the two models $X(P)$ and $H(P)$ are combined, with flags used for fixing the right set of actions and fluents for performing one operation or the other. Every time an action is applied using the execution model $X(P)$, the resulting state is closed with the deductive K-rules. Likewise, every time the heuristic model $H(P)$ is used to compute the heuristic of a state s, the state is closed first by adding ML literals for each literal KL in s, resulting in a state s'. This second closure operation is straightforward; while the former is currently implemented by converting the K-rules into (Horn) clauses and running Unit Resolution in MiniSAT (Een and Sörensson 2003). Notice then that if an action a is found helpful by FF in a state s' using the heuristic model $H(P)$, then this action a is necessarily applicable in the state s of the execution model $X(P)$ (because ML is true in s' iff KL is true in s).

The working loop of the CLG planner proceeds then as follows:

1. given the current state s_x in $X(P)$ (initially I' and assumed to be K-closed), the modified FF planner computes an *improving action sequence* π,
2. the actions in π are then applied *in the execution model* $X(P)$, starting in the state s_x. By construction (see be-

low) if π contains a non-deterministic (sensing) action, this action must occur last in π. Non-deterministic actions are applied, letting the environment, a simulator, or a 'coin' choose the effect, which is observed and applied leading to a new state s_z. If a *full contingent plan* is desired, all possibilities must be tried, recording the action sequences leading to the goal for each possible outcome.

3. if the state s_z is a goal state in $X(P)$, the execution (along this branch in the full contingent plan setting) is successfully terminated, else the cycle repeats at 1 with $s_x := s_z$.

The 'improving action sequence' π in Step 1 refers to the action sequence found by FF for improving the heuristic value $h_{\text{FF}}(s'_x)$ of the state s'_x that extends s_x with the ML literals for $KL \in s$. This is done by FF using an enforced-hill climbing procedure (EHC) using the $H(P)$ model where actions that are not helpful are pruned. If this improvement step fails (which occurs in a single instance below), in our modified FF, a Best First Search for finding a state that improves the heuristic value $h_{\text{FF}}(s'_x)$ is used instead, where no actions are pruned (this is slightly different than in FF where the BFS, when triggered, is used to solve the whole problem, not to perform a local search).

We have also modified FF to establish a simple preference for sensing actions: when trying to improve the heuristic value $h_{\text{FF}}(s)$ of a state s: the value of the states s' resulting from sensing actions in the EHC Search is set $h_{\text{FF}}(s) - 1$. This implies that when a sensing action is found to be *helpful* the action is applied. One reason for this modification is that the application of a helpful action *a without deletes* (such as sensing actions in $H(P)$) in a state s, should normally decrease the value of the h_{FF} heuristic; yet since the relaxed plans are computed from scratch in every state in FF, this doesn't always happens. The same assumption is made for the Best-First Search, although as mentioned above, BFS gets triggered in a single instance. This choice ensures that if the improving sequence π contains a non-deterministic (sensing) actions, this action appears last in π.

A few extensions to the conformant translation $K(P)$ underlying the two models $X(P)$ and $H(P)$ have been incorporated as well. In particular, in the $K_1(P)$ translation in (Palacios and Geffner 2007), the merges $m \in M_L$ are the clauses $t_1 \vee t_2 \vee \cdots \vee t_m$ that are true in the initial situation such that each t_i is relevant to the literal L. A required change for the contingent setting was to replace this last condition by "t_i relevant to L or relevant to an observable", as that tags t_i may be found to be false. Other modifications were needed in the way the conformant planner T_0, built on top of the translation $K_1(P)$, simplifies and removes some of the literals, but we cannot provide the details here for lack of space.

Examples

We consider two examples: a simple one, for illustrating how the formulation works in some detail, and a more complex one, for illustrating the functionality and capabilities of the resulting planner.

Diagnosis

This is the problem known as 'medical' in the literature, abbreviated in the tables below as *medpks-n*, where n is the number of possible illnesses. We consider the version of the problem for $n = 1$, where the patient may have just one illness (i) or be healthy (h). The goal is to have the patient healthy. Two 'physical' actions are available, `medicate` with precondition i and effect h, and `stain` with no precondition and effect $i \rightarrow s$.[2] There is then a sensing action `inspect` for observing the true value of s. The initial situation is $\neg s \wedge \text{xor}(i, h)$ and a contingent plan for this problem is $\{$`stain`, `inspect`, if s `medicate`$\}$.

For generating one execution in CLG, let us assume that the patient is already healthy. The execution and heuristic models $X(P)$ and $H(P)$ use the conformant translation where the only tags t are i and h. In the encoding P, the atom i is static, while h is not (it can be made true; although i is not deleted then). Since h is not static, the tag $t = h$ refers to the static copy h_0 of h that captures the value of h in the initial situation. The initial state in $X(P)$ is given by the literals

$$\{K\neg s, K\neg s/i, K\neg s/h, Ki/i, Kh/h, \neg K\neg i, \neg Ks, \ldots\}$$

while the initial state for $H(P)$ adds to this set the literal $M\neg s$. From this initial state, the following relaxed plan is obtained for $H(P)$, where actions are listed in sequence along with the rules and preconditions involved:

(1)	`stain`:	$\neg K\neg i \rightarrow \neg K\neg s$
(2)	`inspect`:	$\neg Ks \wedge \neg K\neg s \rightarrow o(s)$
(3)	M-Tag Refut.:	$K\neg s/h \wedge o(s) \rightarrow M\neg h_0$
(4)	M-Tag Disj.:	$M\neg h_0 \rightarrow M\, i$
(5)	`medicate`:	Prec: $M\, i$. Eff: $Ki/i \rightarrow Kh/i$
(6)	merge Kh:	$Kh/i \wedge Kh/h \rightarrow Kh$

CLG selects then the action `stain` to be executed, activating effects like $Ki/i \rightarrow Ks/i$ and $\neg K\neg i \rightarrow \neg K\neg s$, leading to the state

$$\{Ks/i, K\neg s/h, Ki/i, Kh/h, , \neg Ks, \neg K\neg s, \ldots\}$$

The relaxed plan that is obtained by calling FF with the heuristic model $H(P)$ from this state, closed with the literals ML for the true KL literals, is like the previous one but without action (1). Then CLG selects `inspect` to be executed, which given the hidden state for this execution, results in $K\neg s$, and the state

$$\{K\neg s, Ks/i, K\neg s/h, Ki/i, Kh/h, \neg K\neg i, \ldots\}$$

which closed under the K-rules $Ks/i \wedge K\neg s \rightarrow K\neg i_0$ and $K\neg i_0 \rightarrow Kh$, yields the goal Kh.

[2] A difference with the standard encoding of the problem is that we make 'having illness i' a precondition of the action 'medicate i' rather than a condition, and eliminate the effect 'dead' when 'not having illness i'. For the Wumpus problem below we make a similar assumption, that for moving into a cell, the cell must be safe; i.e., it must contain no wumpus nor pit, rather than having the effect 'dead' when it is not safe. This move from conditions to preconditions, however, can be accounted for automatically using techniques from classical planning.

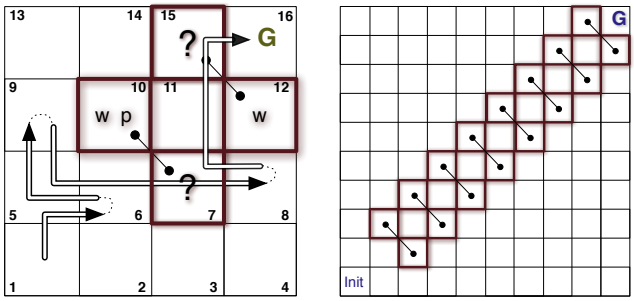

Figure 1: Wumpus domain: An execution in Wumpus-4 on the left, and an illustration of Wumpus-n on the right for $n = 10$.

Wumpus

Figure 1 shows two versions of the Wumpus world, modified from (Russell and Norvig 1994), where an agent at the bottom-left corner is supposed to grab the gold at the opposite corner. The agent can move only into safe cells that do not contain either a Wumpus (W) or a Pit (P). If the locations of the W's and P's is completely unknown, then the problem has no contingent solution, as the W's and P's may block all the routes to the goal. In order to make solvable instances, we chose to select pairs of cells (shown in the figures as linked by segments) such that one of the two cells in each pair is safe. Aside from the action of moving to an adjacent safe cell, the agent has two sensing actions in each cell: feel-breeze and feel-stench. These actions uncover the value of a boolean that is true iff there is a Pit or Wumpus respectively, in an adjacent cell.

The figure on the right shows the instance Wumpus-n for $n = 10$. For $n = 7$, the full contingent plan found by CLG contains 6552 actions. CLG cannot build a full contingent plan for $n = 10$, although as we will see, it can generate executions leading to the goal with more than 50 actions in less than 2 seconds. One execution for Wumpus-4 is shown in the left figure. As we have said above, for this problem, the agent knows that one of the cells 10 or 7 is safe, and that one of the cells 15 or 12 is safe as well. The unsafe cells, may contain a Wumpus, a Pit, or both. For the execution, the hidden state has a Wumpus at cells 10 and 12, and a Pit at cell 10. The agent starts at cell 1 and heads first to cell 6 where it first does feel-breeze and then feel-stench. These moves are not best though, as an observation of a breeze or a stench in cell 6, does not determine which of the two cells 10 or 6 is unsafe. From the given hidden state, the two actions result in positives, and the agent heads then to cell 9 where it does a feel-breeze again. Obtaining again a positive, it concludes that there is a Pit at cell 10, that cell 10 is then not safe, and therefore that cell 7 must be safe, from which the stench at cell 6 requires a Wumpus at 10 as well. From cell 9, the agent heads to cell 8 where it does a feel-breeze, which is false. It then does a feel-stench which is true, inferring then that there is a Wumpus at cell 12, and therefore that cell 15 is safe. It then heads to cell 16, avoiding cell 12, and there grabs the gold.

Let us emphasize that this behavior is *not* the result of a *replanner* that makes optimistic assumptions about the environment, except when it is not 'safe', refining its beliefs along the way. The heuristic model $H(P)$ from which the classical planner obtains the heuristics, consider *all possible contingencies*, albeit, in an implicit manner, by dealing with the conformant relaxation of the problem. This probably explains why, in the execution above, the agent ends up going first to cell 6 rather than to 9 or 3, which in this case, would make more sense. On the other hand, a greedy replanner may go first to cell 14 or cell 8 for checking whether the passage to the goal is safe or not. In the 'best case' this is an option, but notice that if one of the sensors returns a positive, it will have to back up to 9 or 3 respectively, to see where the breeze or stench is coming from.

Experimental Results

We tested CLG over a broad range of problems comparing it with Contingent-FF. Contingent-FF appears to scale up better than other contingent planners such as POND, while planners such as MBP use a different modeling language. We used Contigent-FF with two options (with or without helpful actions) and report the best option for each instance. The experiments are obtained on a Linux machine running at 2.33 GHz with 1.8Gb of RAM.Some of the the benchmarks are taken from the Contingent-FF distribution, like *ebtcs-*x, *elog-*x, *medpks-*x, and *unix-*x. We also consider some new domains:

- *colorballs-n-*x is the problem of disposing of x colored balls with a single gripper, in a $n \times n$ grid. The locations and color of the balls are not known, but can be observed when the agent and ball are in the same cell. There are four colors $c_1, \ldots, c_4$, and balls of each color must be disposed in a different corner.
- *ncolorballs-n-*x is a different representation of a similar problem where each cell in the grid may contain a ball of any x color (i.e. the grid may contain up to n^2 balls). The parameter x in this case indicates number of possible colors.
- *doors-*n is the problem of moving from the left most column c_1 of a square $n \times n$ to the right most column c_n, for an odd n. In the even columns $(c_2, c_4, c_6, \ldots)$ there is a hidden door that can be detected by a sensing action from an adjacent cell. The agent starts in the center of column c_1 and must end up in the center of column c_n.
- *localize-*n involves a grid $n \times n$ with parallel corridors, and an agent whose initial location is unknown that must reach a designated destination. For this, it has four sensors for detecting the presence of adjacent walls in each one of the four directions.
- *wumpus-*n involves a grid $n \times n$ with possible locations for Wumpus and Pits, that must be avoided for reaching a goal, as explained above.

Table 1 displays the ability of CLG to build *full contingent plans* which compares favorably with Contingent-FF. In almost all cases, only CLG scales up to the larger problems, with the length of the longest branches or executions being roughly similar. The total number of actions is also

	Contingent FF			'Contingent' CLG		
problem	time	depth	nacts	time	depth	nacts
ebtcs-70	69,66	70	139	16,52	71	209
elog-7	0,06	23	223	0,12	22	210
eloghuge	M			423,35	73	38894
medpks-70 *	1098,44	71	140	15,4	72	141
medpks-99 *	T			89,03	101	199
unix-4	222,65	179	238	92	181	240
colorballs-3-2	1,86	38	2781	1,31	34	2641
colorballs-3-3	MA			47,35	48	60924
colorballs-3-4	T			1349,1	61	1329235
colorballs-4-1	0,27	45	277	0,25	52	295
colorballs-4-2	35,88	67	18739	35,67	74	20050
colorballs-4-3	T			T		
colorballs-5-1	1,72	91	611	1,15	65	586
colorballs-5-2	909,22	143	71157	534	107	72817
colorballs-5-3	T			T		
colorballs-7-1	41,97	317	1826	32,72	137	1629
colorballs-7-2	T			T		
colorballs-9-1	T			264,03	197	3385
colorballs-9-2	T			T		
localize-5 *	9,8	35	188	0,45	22	137
localize-7 *	MC			4,77	31	314
localize-9 *	MC			28,97	44	602
localize-11 *	MC			M		

Table 1: CLG vs. Contingent-FF in the computation of full contingent plans: 'time' is search time in secs. 'depth' is number of actions in longest branch (execution), 'nacts' stands for the total number of actions in all branches (execs), 'M' means memory out, 'T' time out of 45m, 'MA' *too many actions*, and 'MC' *too many clauses*. In instances marked with '*', best results for Contingent-FF obtained without the *helpful actions* option.

similar, even if Contingent-FF builds a *graph*, where executions leading to the same belief state end up in the same node, while CLG builds a *tree*, as no duplicate detection is done.

Table 2 provides the same evaluation for CLG but in domains that Contingent-FF reports, incorrectly, as unsolvable. After checking with Hoffmann and Brafman it appears that this is bug that results from simplifications made in Contingent-FF that are not easy to fix.

Table 3 shows results of a sample of 100 executions of CLG in each of the instances. All the sample executions end up in the goal with the number of actions shown in the fourth column. The last column shows how focused is the search for the next action to apply: a ratio of one means that the selection of the next action picks up the first (helpful) action considered; similarly, a ratio of 2 means that 2 actions were considered in the local search, and so on. The results show that the search is very focused, except in *ncolorballs.*

The results also show that for many problems for which CLG could not build a full contingent plan, due to its size, CLG generates executions that lead effectively to the goal. These problems include colorballs-7-2, doors-11, ncolorballs-8-2, wumpus-10, and wumpus-15.

The first two columns of Table 3 show the time consumed in the translation from P to the execution and heuristic models $X(P)$ and $H(P)$, as well as the size of the resulting PDDL file in MBytes. These times, as it can be seen, can grow up to a several minutes and the size of the resulting PDDL files can grow up to several Mbytes. Right now,

	'Contingent' CLG		
problem	time	depth	nacts
doors-7	15,91	51	2153
doors-9	1801,65	95	46024
doors-11	T		
ncolorballs-3-3	68,39	57	25421
ncolorballs-3-4	298,21	62	81765
ncolorballs-4-2	T		
wumpus-5	0,63	41	636
wumpus-7	22,12	57	6552
wumpus-10	T		

Table 2: Construction of full contingent plans in domains reported (incorrectly) 'unsolvable' by Contingent-FF

	CLG in execution mode				
	Translation		Time	#acts	nodes /acts
problem	time	size	avg / max	avg / max	avg
colorballs-4-1	0,2	0,4	0,01 / 0,06	13,84 / 52	1,86
colorballs-4-2	0,3	0,7	0,05 / 0,12	25,67 / 60	2,08
colorballs-4-3	0,5	1,1	0,15 / 0,35	37,23 / 71	2,24
colorballs-5-1	0,5	1,0	0,03 / 0,15	13,1 / 59	1,43
colorballs-5-2	1,2	2,0	0,18 / 0,76	26,45 / 72	1,78
colorballs-5-3	1,9	2,9	0,45 / 1,18	34,24 / 82	1,94
colorballs-6-4	8,8	9,3	3,13 / 7,94	47,46 / 106	2,04
colorballs-7-1	3,3	5,0	0,24 / 0,87	15,06 / 53	1,23
colorballs-7-2	7,3	9,8	1,41 / 3,97	25,92 / 81	1,79
colorballs-9-1	13,6	17,39	1,06 / 2,11	15,93 / 39	1,19
doors-9	6,5	9,0	1,01 / 2,65	25,91 / 69	1,07
doors-11	20,6	22,2	3,36 / 9,36	31,11 / 78	1,06
doors-13	56,6	49,7	T		
ebtcs-70	8,1	13,7	0,65 / 3,29	4,62 / 18	1
elog-huge	1,2	1,7	0,52 / 1,47	43,92 / 72	1,9
localize-9	5,5	9,8	1,11 / 1,99	28,21 / 44	1,17
localize-11	12,9	22,2	M		
medpks-70	6,2	11,0	0,55 / 2,91	4,23 / 16	1
medpks-99	16,7	26,0	1,71 / 10,84	4,21 / 18	1
ncolorballs-3-4	0,4	1,0	0,08 / 0,3	31,44 / 56	2,61
ncolorballs-4-2	0,8	2,2	1,27 / 3,8	106,5 / 171	5,46
ncolorballs-7-2	14,5	35,6	522,5 / 1744	484,5 / 800	12,03
ncolorballs-8-2	33,3	72,7	681 / 809,09	510,5 / 540	7,79
ncolorballs-9-2	66,1	141,7	T		
unix-4	29	87,7	4,44 / 19,75	7,68 / 40	2,08
wumpus-7	1	1,6	0,16 / 0,31	35,78 / 51	1,54
wumpus-10	7,3	5,4	1,78 / 2,88	55,33 / 71	1,99
wumpus-15	68,8	22,4	13,08 / 20,67	88,7 / 111	1,89
wumpus-20	383,4	63,5	T		

Table 3: CLG in execution mode over 100 sample runs: 'translation time' and 'size' is time and size (in MB) for getting $X(P)$ and $H(P)$ from P. 'time avg/max' is avg/max execution time over sampled execs, '#acts avg/max' is avg/max number of actions in sample execs, and 'node/acts' is avg. ratio of nodes expanded in execution by number of actions executed. 'T' stands for timeout of 35m on loading the PDDL file or searching. Times are in seconds.

however, a main bottleneck in the current implementation of CLG is loading these files into FF, a process that can takes even more time in the largest instances. We do not report these times though because they are not relevant; in the next version of CLG we will bypass the use of external PDDL files and write this information directly into FF's data structures.

Discussion

We have built on recent ideas that allow delete and precondition-free contingent problems to be converted into conformant problems, and conformant problems into classical ones, for mapping the action selection problem in contingent planning into an action selection problem in classical planning. The resulting Closed-Loop Greedy Planner CLG has been tested over the standard benchmarks in contingent planning and over problems with exponential size plans, scaling up in both settings, producing behaviors that lead to the goal in spite of the lack of explicit contingent plans, and solving a broad range of problems, some of which appear to be beyond the scope of current planners. These behaviors are not the result of a replanner that makes optimistic assumptions about the environment, but of an encoding, that due to its conformant nature, take all possible contingencies into account.

The most immediate open problem is a theoretical characterization of the conditions under which this formulation is complete. Some illustrative questions in this regard are: 1) under what conditions is the execution model $X(P)$ complete, ensuring that solutions to the contingent problem P translate into solutions for $X(P)$; 2) under what conditions is the heuristic model $H(P)$ complete, ensuring that solutions to $X(P)$ translate into solutions to $H(P)$ by combining the actions in all the branches in a sequence; 3) under what conditions, can these complete models be polynomial in size, so that the deductive inference that must be carried out from the observations remains polynomial as well? In the proposed formulation, everything is polynomial but completeness is not ensured, and in particular, the justification of the deductive rules for both the K and M-literals is empirical. The conformant width of a literal, as defined by Palacios and Geffner, measures the size of the tags required to 'hit' the clauses in the initial situation whose literals are all relevant to L. Most likely this notion of width will prove relevant for answering some of these questions, provided that it is suitable extended to take into account observables as well.

Acknowledgments

To be included.

References

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2006. Strong planning under partial observability. *Artif. Intell.* 170(4-5):337–384.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, 52–61. AAAI Press.

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Planning graph heuristics for belief space search. *Journal of AI Research* 26:35–99.

Een, N., and Sörensson, N. 2003. An Extensive SAT-solver. In *Proceedings of SAT*.

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *Proc. ECP-99, Lect. Notes in AI Vol 1809*. Springer.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS 2005*.

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. AAAI-06*.

Palacios, H., and Geffner, H. 2007. From conformant into classical planning: Efficient translations that may be complete too. In *Proc. ICAPS-07*.

Pearl, J. 1988. Embracing causality in default reasoning. *Artificial Intelligence* 35:259–271.

Peot, M., and Smith, D. E. 1992. Conditional nonlinear planning. In Hendler, J., ed., *Proc. 1st Int. Conf. on AI Planning Systems*, 189–197.

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. AIPS'02*, 212–221.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of AI Research* 4:287–339.

Rintanen, J. 2004. Complexity of planning with partial observability. In *Proc. ICAPS-2004*, 345–354.

Russell, S., and Norvig, P. 1994. *Artificial Intelligence: A Modern Approach*. Prentice Hall.