

A Translation-based Approach to Contingent Planning

Alexandre Albore

Universitat Pompeu Fabra
Barcelona, Spain
alexandre.albore@upf.edu

Héctor Palacios

Universidad Simón Bolívar
Caracas, Venezuela
hlp@ldc.usb.ve

Héctor Geffner

ICREA & Universitat Pompeu Fabra
Barcelona, Spain
hector.geffner@upf.edu

Abstract

The problem of planning in the presence of sensing has been addressed in recent years as a non-deterministic search problem in *belief space*. In this work, we use ideas advanced recently for compiling conformant problems into classical ones for introducing a different approach where contingent problems P are mapped into non-deterministic problems $X(P)$ in *state space*. We also identify a *contingent width* parameter, and show that for problems P with bounded contingent width, the translation is sound, polynomial, and complete. We then solve $X(P)$ by using a relaxation $X^+(P)$ that is a classical planning problem. The formulation is tested experimentally over contingent benchmarks where it is shown to yield a planner that scales up better than existing contingent planners.

1 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 non-deterministic (AND/OR) search problem in belief space [Bonet and Geffner, 2000; Hoffmann and Brafman, 2005; Bertoli *et al.*, 2006; Bryce *et al.*, 2006].

In this work, we introduce a different approach to contingent planning that uses and extends ideas advanced recently for compiling conformant problems into classical planning problems [Palacios and Geffner, 2007]. In this approach, a contingent problem P , which is a non-deterministic search problem in *belief space*, is compiled into a non-deterministic problem $X(P)$ in *state space* whose literals represent the beliefs over P . We assume that the problem P involves uncertainty in the initial situation only and that actions are all deterministic. In such a case, as we will see, a straightforward sound and complete compilation is feasible by tagging each of the fluents L in P with the possible initial states of

P . This compilation, however, is linear in the number of possible initial states that is exponential in the number of fluents. We show nonetheless that even in such cases, a sound, complete, and polynomial translation $X(P)$ is possible, provided that the problem P has *bounded contingent width*, and show that the contingent width of almost all existing benchmarks is 1; a result that parallels the one reported by Palacios and Geffner for conformant planning. We then show how the non-deterministic but fully observable problem $X(P)$ can be solved using a suitable relaxation $X^+(P)$ that is a classical planning problem.

The paper is organized as follows. We present the contingent problem P , the conformant translation $K(P)$, the new contingent translation $X(P)$, and the conditions under which it is sound, complete and polynomial. We then introduce the relaxation $X^+(P)$ and a contingent planner CLG that uses both $X(P)$ and $X^+(P)$. We finally present empirical results and a summary.

2 Contingent 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, O for the actions, I is a set of *clauses* over F defining the initial situation,¹ and $G \subseteq F$ is the goal situation. 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. Sensing actions a uncover the truth value of a positive literal L and are denoted as $obs(L)$. Sensing actions may have preconditions but for simplicity we assume that they do not have any other effects. We refer to the conditional effects $C \rightarrow L$ of an action a as the *rules* associated with a , and write them often as $a : C \rightarrow L$. We use $\neg L$ to refer to the complement of L .

3 Conformant Translation $K(P)$

The translation $K(P) = K_{T,M}(P)$ for conformant problems P due to P&G [Palacios and Geffner, 2007] involves two parameters: a set of *tags* T and a set of *merges* M . A tag t is a

¹No closed world assumption about I is made throughout the presentation, yet as it is standard, the actual planner assumes $\neg L$ in I if the positive literal L is not mentioned at all in I .

set (conjunction) of literals in P whose status in the initial situation I is not known, and a merge $m \in M$ is a collection of tags t_1, \dots, t_n that stands for the DNF formula $t_1 \vee \dots \vee t_n$. Tags are assumed to represent consistent assumptions about I , i.e. $I \not\models \neg t$, and merges, disjunction of assumptions that are valid in I ; i.e. $I \models t_1 \vee \dots \vee t_n$.

The fluents in $K_{T,M}(P)$, 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 is 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”, expressing that L is known unconditionally) when KL/t' has been obtained for each tag t' in a merge $m \in M$. Formally,

Definition 1. For a conformant problem $P = \langle F, O, I, G \rangle$, the classical problem $K_{T,M}(P) = \langle F', O', I', G' \rangle$ is:

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

with t ranging over T and 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 set M is taken to be collections M_L of merges for each literal L . When $C = L_1, \dots, L_n$, the expressions KC/t and $\neg K\neg C/t$ are abbreviations for $KL_1/t, \dots, KL_n/t$ and $\neg K\neg L_1/t, \dots, \neg K\neg L_n/t$ respectively. Similarly, KL stands for KL/t_0 where t_0 is the “empty tag”. The empty tag is assumed in all sets T and is not mentioned explicitly. Notice that a rule $a : C \rightarrow L$ in P gets mapped into “support rules” $a : KC/t \rightarrow KL/t$ and “cancellation rules” $a : \neg K\neg C/t \rightarrow \neg K\neg L/t$; the former “adds” KL/t when the condition C is known in t , the latter undercuts the persistence of $K\neg L/t$ except when (a literal in) C is known to be false in t .

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 that can be obtained by just dropping the merge actions. On the other hand, the *complexity* and *completeness* of the translation depend on the choice of tags T and merges M . The $K_i(P)$ translation, where i is a non-negative integer, is a special case of the $K_{T,M}(P)$ translation where the tags t are restricted to contain at most i literals. $K_i(P)$ is exponential in i and complete for problems with *conformant width* less than or equal to i . The planner T_0 feeds the $K_1(P)$ translation into the classical FF planner [Hoffmann and Nebel, 2001], and was the winning entry in the Conformant Track of the 2006 IPC [Bonet and Givan, 2006].

4 Conformant Translation $X(P)$

The translation of a contingent problem P into an equivalent classical problem P' is not possible as the problems have different solution form. The solution to a contingent problem P

is a contingent plan that can be regarded as a *policy tree* Π : a tree T with a function A mapping the internal nodes n of T into actions $a(n)$ in P .² Roughly, a node n in the tree has two children when the action $a = a(n)$ is a sensing action, so that the different (complete) branches stand for the different possible executions. A policy tree Π solves the problem P when the executions associated with each of the branches in Π are all feasible and end up in belief states where the goals are true. The formalization of this notion is well known and requires the specification of the belief state $b(n)$ associated with each of the nodes in the tree, that we omit for lack of space [Bonet and Geffner, 2000].

Contingent problems P cannot be translated into classical problems but can be translated into *non-deterministic fully observable problems* $X(P)$: these are problems where the states are observable but some actions have non-deterministic effects. It is simple to notice that (strong) solutions to such problems are also policy trees; the difference being that belief states $b(n)$ can be replaced by plain states $s(n)$ over $X(P)$.

The translation $X(P) = X_{T,M}(P)$ can be defined as the translation $K_{T,M}(P')$ of the *conformant fragment* P' of P (i.e. P without the sensing actions) extended with two components: a suitable encoding of the *sensing actions*, expressed as non-deterministic actions, and two *deductive rules* expressed as actions that extend the conformant merges, reflecting that tags may be inferred to be false when observations are gathered. We will see that for suitable choices of tags and merges, the translation $X_{T,M}(P)$ can be shown to be complete too.

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

$$obs(L) : \neg KL \wedge \neg K\neg L \rightarrow KL \mid K\neg L. \quad (1)$$

These are *physical actions* that result into either KL or $K\neg L$ when neither KL nor $K\neg L$ hold, thus capturing the effect of observing the truth value of L at the knowledge-level [Petrick and Bacchus, 2002].

The additional deductive rules encoded as actions with single conditional effects are:

1. **Contingent Merge:** $\bigwedge_{t \in m, m \in M_L} (KL/t \vee K\neg t) \rightarrow KL$
2. **Tag Refutation:** $KL/t \wedge K\neg L \rightarrow K\neg t$

Contingent Merge (CM) is a generalization that subsumes the conformant *Merge* actions in the conformant translation by replacing every literal KL/t by the disjunction $KL/t \vee K\neg t$. This is because in the contingent setting, for L to be known, it suffices to have L true given the tags t that have not been refuted by the observations. The tags t that are refuted by the observations are obtained from the second rule, *Tag Refutation (TR)*, as the tags that predict a literal L which is known to be false. In these rules, Kt and $K\neg t$ refer to new atoms added to $X(P) = X_{T,M}(P)$ for all $t \in T$ (except the empty tag).

²Most contingent planners build graphs rather than trees, yet this distinction is irrelevant from a theoretical point of view.

Definition 2. For a contingent problem P , $X_{T,M}(P)$ is the non-deterministic, fully observable problem given by translation $K_{T,M}(P')$ of the conformant fragment P' of P , extended with the non-deterministic actions (I) and the deductive actions **CM** and **TR**.

As an example, consider a medical-like contingent problem P with $I = \{\neg s\}$, $G = \{h\}$, actions a and b with effect h and preconditions d and $\neg d$ respectively, action c with effect $d \rightarrow s$, and the sensing action $obs(s)$ with no preconditions. A contingent plan for this problem is

$$\Pi = \{c, obs(s), \text{ if true } a \text{ else } b\}$$

that can be understood as a policy tree with two complete branches with action/observation sequences $\pi_1 = [c, o^+(s), a]$ and $\pi_2 = [c, o^-(s), b]$, where $o^+(x)$ and $o^-(x)$ stand for the observations x and $\neg x$ respectively. In the translation $X_{T,M}(P)$ with tags $t_1 = \{d\}$ and $t_2 = \{\neg d\}$, and merge $m = \{t_1, t_2\}$ for d and $\neg d$, this plan would work as follows. The action c in the first branch yields the literal $K\neg s/t_2$, while $o^+(s)$ “yields” the literal Ks . From **TR**, $K\neg t_2$ follows, which along with Kd/t_1 (that is initially true and persists) permits to obtain Kd from **CM** for $L = d$, so that action a can be applied and the goal Kh obtained. The second branch works in a similar way.

The example shows that the solution tree Π for P solves also the translation $X_{T,M}(P)$ for suitable tags and merges, provided that *deductive actions* (merges and tag refutations) are interleaved. This is the basis for the notions of soundness and completeness below.

5 Properties of $X(P)$

For convenience, we consider policy trees Π^* for $X(P)$ where the labels $a(n)$ associated with the internal nodes of Π^* can stand for an action a in P followed by a (possibly empty) sequence of deductive actions (merges and tag refutations). We write then $a(n) = a^*$ and prevent deductive actions from appearing anywhere else in Π^* . Any policy tree for $X(P)$ can be written in this way. With this notation, two transformations on policies can be defined: *dropping* the deductive actions from Π^* replaces the “actions” $a(n) = a^*$ by the actions $a(n) = a$ over all nodes in Π^* , and vice versa, *adding* deductive actions in a policy Π for P , replaces the actions $a(n) = a$ by $a(n) = a^*$ for suitable sequences a^* headed by a . The soundness and completeness of $X(P)$ can be then defined as follows:

Definition 3. A translation $X(P)$ is sound if for any policy tree Π^* that solves $X(P)$, the policy tree Π obtained from Π^* by dropping the deductive actions solves P .

Definition 4. A translation $X(P)$ is complete if any policy tree Π that solves P can be extended into a policy tree Π^* that solves $X(P)$ by adding some deductive actions to Π .

The first result that can be established is soundness:

Theorem 5 (Soundness). The translation $X(P)$ is sound.

The conditions that ensure completeness are more subtle and are considered below. A simple complete translation however can be obtained from the general $X_{T,M}(P)$ scheme

as in conformant planning, by letting the set of tags T represent the set S_0 of all the possible initial states of P and by having a single merge m for each precondition and goal literal in P , $m = S_0$. We call the resulting translation $X_{S_0}(P)$:

Theorem 6 (Completeness of X_{S_0}). The translation $X_{S_0}(P)$ is sound and complete.

This translation is exponential in the number of uncertain fluents in the initial situation of P , in the worst case. Still, where this number is small enough, the translation can be quite effective. Before addressing how these non-deterministic but fully-observable problems $X(P)$ are solved, we consider complete translations that may be compact.

5.1 Covering Translations

As in P&G, the merges $m = \{t_1, \dots, t_n\}$ that are needed for completeness in $X(P)$ are the ones in which each tag t_i subsumes the clauses in I that are relevant to the precondition and goal literals of P . Clearly, the merges $m = S_0$ achieve this, yet more compact merges will often do as well. We will say that a merge $m = \{t_1, \dots, t_n\}$ with tags t_i all consistent with I , *satisfies* a set of clauses \mathcal{C} iff for each t_i in m and each clause c in \mathcal{C} , there is at least one literal L' in c entailed by t_i and I . For characterizing the set of clauses $C_I^o(L)$ that are relevant to a given literal L , we assume like P&G that I is in prime implicate form [Marquis, 2000], meaning that I includes only the inclusion-minimal clauses it entails, along with the tautologies $L \vee \neg L$ when neither L nor $\neg L$ are in I .

Definition 7. For a literal L in P , with I in prime implicate form, $C_I^o(L)$ is the set of non-unary clauses $c \in I$ such that each $L_i \in c$ is relevant to an observable literal L' in $O(L)$ or to L .

The notion of relevance is the same as the one in P&G,³ and $O(L)$ stands for a set of *observables*: literals L' such that either $obs(L')$ or $obs(\neg L')$ is an action in P . The condition for completeness can then be expressed as follows:

Definition 8. $X_{T,M}(P)$ is a covering translation for a contingent problem P if for each precondition and goal literal L in P such that $C_I^o(L)$ is non-empty, M contains a merge m that satisfies $C_I^o(L)$.

Theorem 9. Covering translations $X_{T,M}(P)$ are complete.

The use of the set $O(L)$ of observables in Definition 7 is the main difference with the corresponding result for conformant planning. This set can be taken to comprise the set of all observables, yet a smaller set does as well: it suffices to take $O(L)$ to be the observables that may affect L ,⁴ and this corresponds to the unique minimal set of observables L' that obey the following condition: L' is in $O(L)$ if there is a clause c in I with a literal L_1 that is relevant to L' and a literal L_2 that is relevant to L'' , where $L'' \in O(L)$ or $L'' = L$.

³A literal L is relevant to L' in P , written $L \rightarrow L'$, if $L = L'$, if $L \rightarrow L''$ and $L'' \rightarrow L'$, if $a : C \rightarrow L'$ in P and L in C , or if $\neg L \rightarrow \neg L'$.

⁴The literals that affect L in the contingent setting are not the same as the literals that may affect L in the conformant setting, which are fully characterized by the notion of relevance. Yet relevance expresses “causal relevance” and in the contingent setting there is “evidential relevance” as well.

As an illustration, consider the problem P with initial situation $I = \{xor(x_1, \dots, x_n)\}$, goal $G = \{y\}$, actions a_i with precondition x_i and effect y , and sensing actions $obs(x_i)$, with $i = 1, \dots, n$. The translation $X_{T,M}(P)$ with tags $t_i = \{x_i\}$ and merges $m = \{t_1, \dots, t_n\}$ for each precondition x_i can be shown to be covering, and hence complete. Indeed, $C_I^o(L)$ for the goal $L = y$ is empty ($O(y)$ is empty), while for the preconditions $L = x_i$, $C_I^o(L)$ contains all the prime implicates that follow from $xor(x_1, \dots, x_n)$ ($O(x_i)$ includes all the observables), all of which are satisfied by each t_i in m .

5.2 Width and X_i Translation

Covering translations require the translation of the CNF formula $C_I^o(L)$ that encodes the uncertainty in I relevant to L , into a DNF formula entailed by I (the merge for L) that satisfies $C_I^o(L)$. Such DNF formula can be computed as follows:

Definition 10. *The cover $c(\mathcal{C})$ of a set of clauses \mathcal{C} given an initial situation I , is the collection of all minimal sets of literals S consistent with I such that S contains a literal of each clause in \mathcal{C} .*

The cover $c(\mathcal{C})$ satisfies \mathcal{C} and can be computed in polynomial time if $|\mathcal{C}|$ is bounded. From the completeness of covering translations, it follows that a complete translation $X_{T,M}(P)$ can be constructed in polynomial time if the size $|C_I^o(L)|$ for all preconditions and goals L in P is bounded. Unfortunately, this condition rarely holds, yet there is a weaker sufficient condition that does: namely, it is often possible to find a subset \mathcal{C} of clauses that are either in $C_I^o(L)$ or are tautologies $L \vee \neg L$ for L or $\neg L$ mentioned in $C_I^o(L)$, we call them *Taut*, such that $c(\mathcal{C})$ satisfies $C_I^o(L)$. The *contingent width* of the literal L is defined in terms of the cardinality of such sets.

Definition 11. *The contingent width of a literal L in P , written $w(L)$, is the size of the smallest (cardinality-wise) set of clauses \mathcal{C} in $C_I^o(L) \cup Taut$ such that $c(\mathcal{C})$ satisfies $C_I^o(L)$.*

A consequence of this definition is that the width of a literal lies in the interval $0 \leq w(L) \leq n$, where n is the number of fluents in P whose status in the initial situation is not known. The width of a problem is the width of the precondition or the goal literal with maximum width:

Definition 12 (Width). *The contingent width of a contingent problem P , written as $w(P)$, is $w(P) = \max_L w(L)$, where L ranges over the precondition and goal literals in P .*

The translation $X_i(P)$ for a non-negative integer i is a special case of the general translation $X_{T,M}(P)$, that for a fixed i , is sound, polynomial in i , and complete if $w(P) \leq i$.

Definition 13. *The set of merges for a precondition or goal literal L in $X_i(P)$ is empty if $C_I^o(L)$ is empty, else it contains a single merge $m = c(\mathcal{C})$ if \mathcal{C} is a set of at most i clauses in $C_I^o(L) \cup Taut$ such that $c(\mathcal{C})$ satisfies $C_I^o(L)$, and is the collection of all such merges for each set \mathcal{C} of i -clauses in $C_I^o(L) \cup Taut$ otherwise.*

The tags in $X_i(P)$ are that ones appearing in the merges along with the empty tag. The crucial property is that

Theorem 14. *For a fixed i , the translation $X_i(P)$ is sound, polynomial, and if $w(P) \leq i$, covering and complete.*

The translation $X(P)$ above with $I = \{xor(x_1, \dots, x_n)\}$ is an instance of the $X_1(P)$ translation. Like in conformant planning, the *contingent width of almost all contingent benchmarks* turns out to be 1, and hence for such problems the $X_1(P)$ translation is complete. It is not necessary however for $X_1(P)$ to be complete for being useful: if $X_1(P)$ is solvable, it can produce a solution to P . The incompleteness of $X_1(P)$ just means that there is no *guarantee* that all solutions to P can be obtained in this way.

6 Solving $X(P)$

The computational payoff of the translation $X(P)$ is that it enables us to compute with states represented by sets of literals rather than with beliefs represented by sets of states. Nonetheless, the solution of non-deterministic fully observable problem like $X(P)$ is not trivial. Fortunately, however, $X(P)$ is a problem of a special type that can be solved using classical planning techniques. The key result is the following. Let $X^+(P)$ stand for the *relaxation* of $X(P)$ where *deletes*, *preconditions*, and actions with *non-deterministic* effects are dropped. $X^+(P)$ is thus a *classical planning problem*.

Theorem 15. *If $X(P)$ is a covering translation and Π^* is a policy tree that solves $X(P)$, then there is a classical plan π for the relaxation $X^+(P)$ that only uses the deterministic action in Π^* .*

That is, as in classical planning, we can use the relaxation $X^+(P)$ that can be solved in polynomial time to get estimates of the size of the plans that are needed for solving $X(P)$. The result applies to covering translations $X(P)$ but not to arbitrary non-deterministic problems, which can be rendered unsolvable when the non-deterministic actions are dropped.

In the CLG planner, the relaxation $X^+(P)$ is strengthened in two ways. First, rather than using $X^+(P)$, we use an *stronger relaxation* $X^+(P_c)$ obtained from a problem P_c that is *equivalent* to P and where each precondition L of an action a in P is copied as a condition of all the effects associated with a . The result is that the “wishful thinking” in $X^+(P_c)$ about the applicability of actions does not translate into “wishful thinking” about the action effects. This is crucial for the heuristics obtained from $X^+(P_c)$ to be well informed. In Contingent-FF, a similar transformation is used where preconditions in P are moved in as conditions [Hoffmann and Brafman, 2005].

Second, rather than making the preconditions of an action a in $X^+(P_c)$ empty, we add literals ML for each precondition L of a in P expressing roughly that L must be known in *some* branch of the contingent plan. The ML literals are added by a suitable relaxation of the sensing actions $obs(L)$

$$obs(L) : \neg KL \wedge \neg K\neg L \rightarrow ML \wedge M\neg L \wedge o(L) \quad (2)$$

that makes *both* ML and $M\neg L$ *true* after observing the truth value of L , and by a set of deductive actions similar to the ones used for the KL literals:

1. **M-CM** $\bigwedge_{t' \in m, t' \neq t} M\neg t' \rightarrow Mt$
2. **M-TR** $KL/t \wedge o(L) \rightarrow M\neg t$
3. **M-K:** $KL \rightarrow ML$

with $o(L)$ representing an atom that is true when the action $obs(L)$ has been done. Moreover, for each a in P with effects $a : L_1, L_2, \dots \rightarrow L$, the rule $a : ML_1, ML_2, \dots \rightarrow ML$ is added as well.

The classical planning problem that results from these extensions is what we call the *heuristic model* $H(P)$:

Definition 16. $H(P)$ is the classical planning problem given by the relaxation $X^+(P_c)$ extended with the ML preconditions, the encoding (2) of the sensing actions, the actions **M-CM**, **M-TR**, and **M-K**, and the rules $a : MC \rightarrow ML$ for rules $a : C \rightarrow L$ in P .

Notice that the sensing actions $obs(L)$ in the form (2) are part of this classical model $H(P)$ where they are needed for achieving the ML preconditions of other actions. The ML literals in the model $H(P)$ arise from the observations or the KL literals, and get propagated by the M actions and rules.

As an illustration of the heuristic model, if a is an action with precondition L and effect $L_1 \rightarrow L_2$ in P , then a will be an action in $H(P)$ with precondition ML and effects $ML_1 \rightarrow ML_2$ and $KL/t, KL_1/t \rightarrow KL_2/t$ for all tags t in $H(P)$.

7 The CLG Planner

The *Closed-Loop Greedy (CLG)* planner uses the $X_1(P)$ translation, called now the *execution model*, for keeping track of beliefs, and the *heuristic model* $H(P)$ for selecting the actions to do next in closed-loop fashion. Starting with the initial state s of $X(P) = X_1(P)$, an action sequence π is selected for application in s , and the loop resumes from the state s' that results, until s' is a goal state in $X(P)$. The action sequence π is obtained using a modified version of the classical FF planner. In FF, a single enforced hill climbing (EHC) step is a local search that results in an action sequence π that maps a state s into a state s' with a better heuristic value h_{FF} . In this local search, the classical model is used for doing the state progression, and its delete-relaxation for computing the relaxed plans. In CLG, the same search strategy is adopted, with the execution model $X(P)$ used for the progression, and the heuristic model $H(P)$ used for computing the relaxed plans. In addition, in order to avoid the consideration of non-deterministic actions in the local search, whenever a “local plan” π that ends in a sensing action $obs(L)$ is being considered, the action sequence π is returned without further evaluation. Notice that for an action to be considered into the local plan, the action must have been found to be “helpful” according to FF’s criterion.

The CLG planner can be used *on-line* or *off-line*. In the first case, the non-deterministic actions $obs(L)$ in $X(P)$ are applied by selecting one of outcomes randomly; in the second, both outcomes are considered. In both cases, CLG is invoked recursively on the resulting states. The on-line mode is used for capturing single executions, while the off-line mode is used for constructing *full contingent plans*.

CLG is implemented on top of a revised version of FF that loads a single PDDL file where convenient flags are used for distinguishing the $X(P)$ and $H(P)$ models. In addition, when feeding a state s into $H(P)$ for computing a relaxed plan, the state s is extended with the ML literals (ML is

added to s iff KL is in s), and similarly, after applying each action sequence π in $X(P)$, the resulting state s' is closed under the deductive K -actions. In CLG, a few other M and K “deductive” actions are included, such as one that exploits static disjunctions $L_1 \vee \dots \vee L_n$ in P for deriving KL_i from $K \neg L_j$ for all other literals $L_j \neq L_i$.

8 Experimental Results

We tested CLG over a broad range of problems comparing it with Contingent-FF and Pond 2.2 [Bryce *et al.*, 2006]. We used Contingent-FF with two options (with or without helpful actions), reporting the best option for each instance. Pond was run with the A^* search algorithm. The experiments are obtained on a Linux machine running at 2.33 GHz with 2Gb of RAM with a cutoff of 45 mn or 1.8Gb of memory.

Some of the benchmarks are taken from the Contingent-FF distribution, like *ebtcs*, *elog*, *medpks*, and *unix*; others are more challenging problems from our own: *cballs-n-x*, about disposing of x balls with unknown location and colour into boxes according to their colour; *doors-n*, about moving in a $n \times n$ grid with hidden doors, *localize-n*, about robot localization in a known map, and *wumpus-n*, a variation of the Wumpus world. Last, *clog* is a variation of *elog* where all action conditions have been moved out as preconditions. In this suite of problems, only two domains have contingent width greater than 1: *cballs*, and *wumpus*.⁵

problem	Contingent FF		Pond		CLG	
	time	#acts	time	#acts	time	#acts
ebtcs-50	11,96	99	6,02	99	9,37	149
ebtcs-70	69,66	139	29,82	139	34,37	209
elog-7	0,06	223	1,10	212	0,18	210
eloghuge		M		M	240,14	38894
medpks-70	1098,44	140		T	9,89	141
medpks-99		T		T	28,77	199
unix-2	0,09	48	2,18	48	0,57	50
unix-4	222,65	238		M	120,72	240
cballs-4-1	0,27	277	0,98	102	0,35	295
cballs-4-2	35,88	18739	40,92	1897	18,83	20050
cballs-4-3		T	1063,11	28008	1537,99	1136920
cballs-10-1		T		M	415,73	4445
cballs-10-2		T		M		T
localize-5	9,8	188		T	0,72	137
localize-7		MC		T	3,80	314
localize-9		MC		T	17,96	602
localize-11		MC		T		M
clog-7		E	1,12	212	0,17	210
clog-huge		E		M	157,94	37718
doors-7		E	21,48	2159	10,60	2153
doors-9		E	1432,34	44082	1042,96	46024
doors-11		E		T		T
wumpus-5		E	5,58	587	1,76	732
wumpus-7		E	703,54	11673	89,32	10681
wumpus-10		E		M		T

Table 1: Full Contingent Plans: Contingent-FF vs Pond vs CLG. Figures shown are total times in seconds, and total number of actions in solution. ‘M’, ‘T’, ‘MC’ and ‘E’ refer to memory out, time out, too many clauses, and buggy response, respectively.

Table 1 displays the ability of CLG to build *full contingent plans* which compares favorably with the other two planners. Times reported stand for total time, and in the case of CLG,

⁵The planner and problem encodings are available at <http://www.upf.edu/pdi/dtecn/alexandre.albore/clg.html>

they include the translation time. The quality of the plans appears to be comparable (except for *cballs*). Domains marked with an 'E', are reported as unsolvable by Contingent-FF without any search. After checking with Hoffmann and Brafman, it appears that this is due to a bug that results from simplifications made in Contingent-FF that are not easy to fix.

problem	CLG in Execution Mode				
	time	size	avg / max	avg / max	ratio
ebtcs-70	7,7	12,9	0,62 / 3,48	12,5 / 71	1,0
elog-huge	1,1	1,7	0,63 / 1,43	44,25 / 59	1,8
medpks-99	13,7	21,6	0,54 / 3,30	15,1 / 100	1,0
medpks-150	48,7	65,0	4,26 / 28,60	20,8 / 150	1,0
unix-4	27,9	87,2	10,88 / 70,03	27,0 / 181	1,9
cballs-9-1	20,9	16,5	1,21 / 7,80	33,7 / 197	1,6
cballs-9-2	56,4	33,7	4,84 / 25,70	57,1 / 288	1,6
cballs-9-3	113,7	51,4	46,26 / 122,19	76,3 / 367	1,7
clog-huge	1,0	1,6	0,44 / 0,71	48,2 / 69	2,6
doors-9	6,0	8,1	0,72 / 1,46	34,2 / 80	1,1
doors-11	19,4	20,0	2,58 / 6,37	42,8 / 120	1,1
doors-13	52,4	44,7	6,78 / 20,68	53,0 / 178	1,1
doors-15	127,5	90,2		MP	
localize-9	4,6	8,9	0,37 / 0,60	21,3 / 34	1,0
localize-11	12,1	20,4		M	
wumpus-7	2,4	4,5	0,42 / 0,56	38,5 / 46	1,6
wumpus-10	12,1	16,8	3,26 / 5,59	57,5 / 86	1,8
wumpus-15	101,1	80,7		M	

Table 2: CLG in Execution Mode: Averages over 50 samples. Figures shown are time and size of translation, avg and max search time for execution, avg and max number of actions in execution, and ratio of nodes per actions executed in local EHC search. 'M' stands for memory out, and 'MP' for too many predicates. Times in seconds.

Table 2 shows average results for CLG used in execution mode over a sample of 50 random executions for each problem. The first two columns of Table 2 show the time consumed in the translation from P into $X(P)$ and $H(P)$, as well as the size of the resulting PDDL file in MBytes. All the 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, and more precisely, the number of nodes expanded in the local EHC search vs. the number of actions executed. Indeed, a ratio of one means a very focused search. The results show that CLG in execution mode can solve problems for which building a full contingent plan is not feasible due to its size. For example, the largest *door* instance solved in off-line mode is *doors-9* that results in a policy tree with more than 46 000 actions; in on-line mode CLG solves *doors-13* that is much larger. The same is true for other domains like *cballs* and *wumpus*.

9 Discussion

We have extended the translation-based approach to conformant planning introduced by Palacios and Geffner, to planning with sensing. In both settings, the translation maps search problems in belief space into search problems in state space with complete translations being exponential in a width parameter that is 1 for most benchmarks. We have also tested these ideas empirically by formulating a contingent planner CLG that uses the new translation along with a suitable relaxation, and have found that the planner scales up better than existing contingent planners, and that when used in on-line

mode, it can scale up to problems for which the construction of full contingent plans is not feasible. Two advantages of the translation-based approach are that it results in compact belief representations that are often complete, and classical plan relaxations that provide useful heuristics.

Acknowledgments

The authors would like to thank D. Bryce and J. Hoffmann for the helpful exchanges regarding Pond and Contingent-FF. This work was partially funded by grant TIN2006-15387-C03-03, from MEC (Spain).

References

- [Bertoli *et al.*, 2006] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Strong planning under partial observability. *Artificial Intelligence*, 170(4-5):337–384, 2006.
- [Bonet and Geffner, 2000] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-00*, 2000.
- [Bonet and Givan, 2006] B. Bonet and B. Givan. Results of the conformant track of the 5th Int. Planning Competition. At <http://www ldc.usb.ve/~bonet/ipc5/docs/results-conformant.pdf>, 2006.
- [Bryce *et al.*, 2006] D. Bryce, S. Kambhampati, and D. E. Smith. Planning graph heuristics for belief space search. *Journal of AI Research*, 26:35–99, 2006.
- [Haslum and Jonsson, 1999] P. Haslum and P. Jonsson. Some results on the complexity of planning with incomplete information. In *Proc. of ECP-99, Lect. Notes in AI Vol 1809*. Springer, 1999.
- [Hoffmann and Brafman, 2005] J. Hoffmann and R. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proc. of ICAPS-05*, 2005.
- [Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of AI Research*, 14:253–302, 2001.
- [Marquis, 2000] P. Marquis. Consequence finding algorithms. In D. Gabbay and Ph. Smets, editors, *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, volume 5, pages 41–145. Kluwer, 2000.
- [Palacios and Geffner, 2007] H. Palacios and H. Geffner. From conformant into classical planning: Efficient translations that may be complete too. In *Proc. ICAPS-07*, 2007.
- [Peot and Smith, 1992] M. Peot and D. E. Smith. Conditional nonlinear planning. In James Hendler, editor, *Proc. of 1st Int. Conference on AI Planning Systems*, 1992.
- [Petrick and Bacchus, 2002] R. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of AIPS-02*, 2002.
- [Pryor and Collins, 1996] L. Pryor and G. Collins. Planning for contingencies: A decision-based approach. *Journal of AI Research*, 4:287–339, 1996.
- [Rintanen, 2004] J. Rintanen. Complexity of planning with partial observability. In *Proc. of ICAPS-04*, 2004.