Using the Causal Graph to enhance Translations to solve Contingent Planning Problems

Ignasi Andrés and Leliane Nunes de Barros
Department of Computer Science
IME-USP, Brazil
{ignasi,leliane } @ime.usp.br

***Abstract*—Planning with partial observation, an area called *contingent planning*, is a complex and challenging problem since it requires to keep track of belief states to search for a contingent plan of actions. Recent approaches considers the agent's knowledge about the world to compile a contingent planning problem into a full observable planning problem, described in an epistemic logic language, and then use an efficcient full observable planner to solve the translated problem. In this paper we use the concept of relevance and causality to propose a new translation based in a structure called Causal Graph that can improve the belief tracking task of contingent Planning problems described in a more general planning language, in particular problems envolving actions with uncertainty on its conditional effects.**

I. INTRODUCTION

Planning is the process of generating an organized set of actions that can achieve a desired goal. Actions are selected by anticipating their expected effects. Automated planning is the area of Artificial Intelligence (AI) that studies this deliberation process. Classical planning problems make several restrictive assumptions: complete information about the environment, deterministic actions, single agent, discrete time and reachability goal. Contingent Planning (CP) [1], relaxes the assumptions of complete information and deterministic actions, and adds sensing actions which requires reasoning about *belief states*.

The conceptual model of contingent planning is as follows. Initially, the agent must consider all possible initial states (the belief state denoted by B_0). Then the agent chooses an action a that can be executed in B_0 which produces a new belief state, according with the expected result of a. If there is hidden information in the current location, the agent can perform an observation o and the result of o will be used to update the agent's beliefs (also producing new belief states). This repeats until the goal is reached or the agent detects a dead-end state (a state from which it is not possible to reach the goal). Note that an action is applicable in a belief state B_i if its preconditions hold in all physical states, i.e. $\forall s \in B_i$. The task of computing the next belief state after every action or observation is called *belief tracking*. This process can be implemented by an AND/OR search (OR for action choices; AND for the observation results) in a belief state space, whose solution is a subtree, branching on non-deterministic actions and observations.

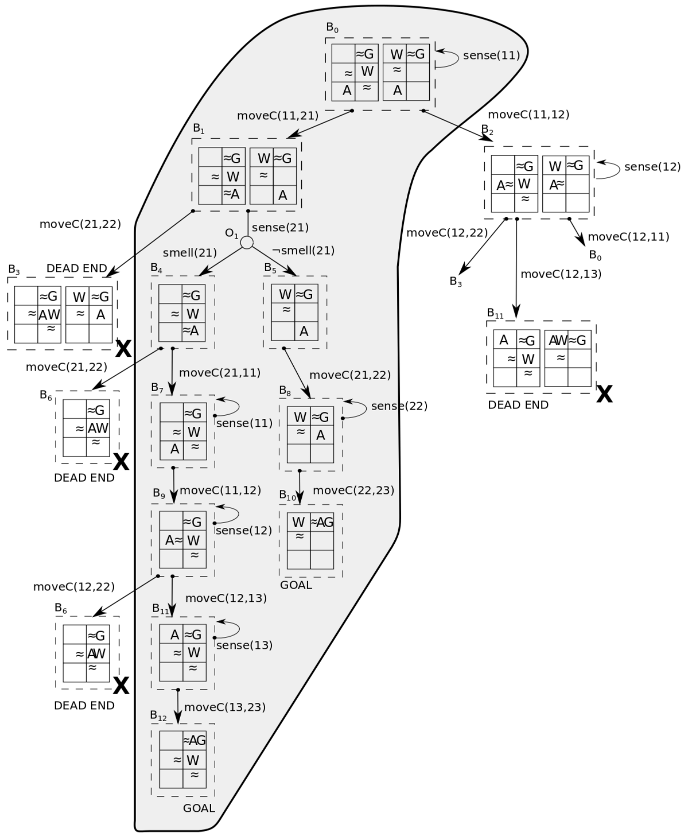

Fig. 1. Contingent planning in the Wumpus World. The agent starts at location (1,1) and must grab the gold at (2,3). Belief states B_i are represented by dashed boxes. Arrows indicate the result of actions or observations. The gray region indicates the solution.

Figure 1 shows a complete AND/OR search tree for a small instance of the Wumpus World, a classical domain where the agent's goal is to get the gold and not to be killed by a monster, called Wumpus. The agent, starting at location (1,1), can move in any direction and sense if there is a smell that indicates the presence of the Wumpus in an adjacent cell; if there is no smell, all the adjacent cells are safe, so the agent can move to one of them; belief states where the agent and the Wumpus are in the same location are dead-ends. The uncertainty in this problem lies in the position of the Wumpus that is unknown in the initial state. In Figure 1, B_i is an OR node and O_j is an AND node; the initial belief state B_0 is the set of states consistent with $wumpus\_at(1,3) \vee wumpus\_at(2,2)$; and the subtree in the grey region is the solution.

For large belief state spaces the AND/OR search can be intractable, i.e. 2-EXP-Complete in the worst case [2]. However, in the last few years, significant progress on the solutions of CP problems has been achieved due to approaches based on translating problems with incomplete information into full observable non-deterministic problems [3] [4]. The conceptual model for the translated problem is a bit different since it is no longer a search in the belief state space but a search in the epistemic state space. In a epistemic state space the agent reasons about what he knows, making explicit what he knows to be true or false, and thus ruling out the uncertainty. E.g., in the translated problem $K(P)$, the initial state s_0 must be consistent with $\neg K wumpus\_at(1,3) \wedge \neg K \neg wumpus\_at(1,3) \wedge ...$ where K is the epistemic operator [5]. One limitation of the translation approaches is that they can only solve efficiently a class of CP problems called *simple*, where there is no uncertainty in the conditional effects. Problems with uncertainty in the conditional effects are called *complex* CP problems. Closed-Loop Greedy Planner (CLG) [4] and Partial Observability-Planner for Relevant Policies (PO-PRP) [6], considered the state-of-the-art planners for contingent planning, can fail to solve complex CP problems.

The search in Figure 1 was generated with complex conditional actions, but a search using simple conditional actions would not be able to apply as many actions but would not generate the four dead-states marked with "X".

In this work we propose a new translation technique, sound and complete for a wider range of CP problems, that is able to analyse conditional effects involving uncertainty, using a structure called the *Causal Graph* to produce a scalable and more informed translation. We also introduce a planner that uses this translation and is capable to solve most of the contingent planning benchmark problems.

This paper is organized as follows: in Section II we formally define contingent planning problems followed by definitions of relevance and *Causal Graph*. Then in Section III we introduce our new contingent planner, named *Comp2BT*, and its modified translation method. We present some empirical results in Section IV and a brief discussion in Section VI.

II. BACKGROUND

A. Contingent Planning Problem

Automated planning in general uses a formal language to describe states [1]. One of the simplest and popular language for classical planning is STRIPS [7], that can be extended to represent a contingent planning domain.

A contingent planning problem in STRIPS is a tuple $P = \langle F, I, A, O, G \rangle$ that extends classical planning such that actions can be deterministic or non-deterministic and the agent has partial observation of the world. More precisely: F is a set of fluent symbols of the problem; I is a set of clauses over F that defines the initial situation, partially known; the non-unary clauses in I are all invariant (i.e. the set of axioms D); A is a set of deterministic or non-deterministic actions; O is a set of observations and G is a conjunction of atoms over F that defines the planning goal.

A state is a truth valuation to the atoms in F. A literal l is a fluent in F, or its negation. A literal l holds in a state s iff s assigns l to be true. An action $a \in A$, is a triple $a = \langle Prec(a), Add(a), Del(a) \rangle$ where $Prec$, consists of a set of literals that must be true in the belief state where the action is applied, i.e. $\forall s \in B, Prec(a) \subseteq s$; $Add(a)$ is a set of fluents from F that become true after executing the action and $Del(a)$ is a set of fluents also from F that become false after executing the action, the result of applying an action a in state s is given by $RESULT(s,a) = (s \setminus Del(a)) \cup Add(a)$. The belief state B' resulting of applying an action a in a belief state B is given by $B' = \{s' : s' = RESULT(s,a) \ \forall s \in B\}$.

An action $a \in A$ can also include a set of *conditional effects* $Eff(a)$, where each effect $e \in Eff(a)$ is given in the form of implications $\{C(e) \rightarrow L(e)\}$, where $C(e)$ is a set of literals and $L(e)$ is a literal. This conditional effect means that, in the moment of executing action a, if the conjunction of literals in C holds, then L will be true in the next state. The formula to calculate the next state after applying an action with conditional effects is $RESULT(s,a) = (s \setminus Del(a)) \cup Add(a) \cup L(e)$ when the conditions of the conditional effect holds. For simplicity and without loss of generality, we can write actions as $a = \langle Prec(a), Eff(a) \rangle$ where $Add(a), Del(a)$ can be transformed into conditional effects of the form $true \rightarrow Add(a)$ and $true \rightarrow neg(Del(a))$ where $neg(Del(a))$ returns the negated atoms of $Del(a)$.

A non-deterministic effect is in the form $\{C \rightarrow L_1(e)|...|L_n(e)\}$, meaning that only one of L_i will be true at the next state.

Since I is a set of clauses, I holds in a set of states, i.e. the belief state B_0. Non-unary clauses in the initial state are called axioms, and we denote this set by D. This set of axioms D represent facts are true in every reachable state in spite of the uncertainty. For example, in the Wumpus world showed in Figure 1, an invariant represent the fact that there is only one Wumpus either at position (1,3) or at position (2,2).

Observations can also have preconditions, and are in the form $o = \langle C, L \rangle$ meaning that when C (body) is true, o uncover the truth value of a positive literal L (head). On performing an observation $o = \langle C, L \rangle$, the successor belief state B' is the set of states from B in which L holds: $B' = \{s | s \in B \text{ and } L \in s\}$. After executing an observation the belief state size will decrease. A sequence of actions (and observations) $a_0, a_1, ..., a_n$ is applicable in B if a_0 is applicable in B and results in a belief state B_1, and inductively, a_i is applicable in B_i resulting in B_n. A belief state B' is reachable from B if there is a sequence of actions (and observations) that when applied in B results in B'. In deterministic contingent problems, the actions are deterministic and the only source of non-determinism are the observations.

The process of updating the beliefs considering the observations and axioms, is given by $B_o = UPDATE(B', o) = UNIT(B' \cup o \cup D)$. Where $UNIT(s)$ is the operation that gives the unit literals in the logic closure of s. And o is the observation and D is the non-unary clauses in the initial state or the axioms.

When planning with partial information and sensing, the solution is not a sequence of actions but a tree structure. The solution must consider every result of an observation. Thus, a solution for a contingent planning problem P is a contingent plan, i.e. a *policy* π [4] that maps belief states B into actions $a(n) \in A$. In a solution, tree nodes are labeled with belief states and edges with actions. A node n has two children if the action $a(n)$ is an observation. We say that Π solves a contingent problem P, iff every execution of π is applicable and finishes in a belief state B' where the goal G holds.

B. Simple Contingent Problems

A planner that solves Contingent Problems has to keep track of the beliefs to select the next best actions. As it is showed in Figure 1, this is done by applying the action on every physical state in the belief state B, and obtaining the new belief state B'. This task can be intractable in the worst case, exponential in the number of fluents of the problem $|F|$.

But in certain problems, when the non-unary clauses in I (i.e. the axioms) are all *invariant*, i.e. a formula that is true in each possible initial state and remains true in any reachable state [8], and the effects of the actions do not depend on fluents that are hidden in I (fluents p such that $I \nvDash p$ and $I \nvDash \neg p$), computation of the new belief state b' can be characterized in a simple form. This is what happens in *contingent simple problems* [9], i.e. problems when the conditional effects don't involve uncertainty.

Formally a *simple contingent planning problem* is:

Definition 1: [9] A contingent planning problem $P = \langle F, I, A, O, G \rangle$ is simple if no hidden fluents appear in the body of a conditional effect.

However hidden fluents can appear in the preconditions of actions and in the heads of conditional effects. Since preconditions must hold in a belief state b before applying an action, they do not carry uncertainty. As an example, consider the action showed in Figure 2 (top). Note that the precondition only includes the negation of the Wumpus location, so it can be applied only in safe locations. Simple contingent problems have interesting properties:

- (Belief representation) For a simple contingent problem, let D be the set of non-unary clauses in I (invariants). If b is a reachable belief state from I, and R is the set of literals that are known to hold in b, b is fully characterized by the formula $D \cup R$.
- (Monotonicity) If the literal L is known in a reachable belief state b of a contingent simple problem, and b' is a reachable belief state from b, then L is known in b.

According to this properties, belief tracking for simple contingent problems is linear in $|F|$, because actions effects do not depend on fluents whose values are unknown.

C. Complex Contingent Problems

In opposition to the simple contingent planning problems, we formally define a *complex contingent planning problem*:

```
Action: move (L1, L2):
  Precond:  adj(L1,L2), at(L1), ¬wumpus_at(L2)
  Effect:   ¬at(L1), at(L2)
Action: moveC (L1, L2):
   Precond: adj(L1,L2), at(L1), alive
   Effect:   ¬at(L1), at(L2), (wumpus_at(L2) → ¬alive)
```

Fig. 2. Move action for the Wumpus world problem. (top) Action $move$ is a simple action; (bottom) Action $moveC$ is an action with uncertainty in the conditional effect.

Definition 2: A contingent planning problem $P = \langle F, I, A, O, G \rangle$ is complex if hidden fluents appear in the body of a conditional effect.

As an example, consider the $moveC$ action showed in Figure 2 (bottom). Notice that the predicate $wumpus\_at(L2)$ in the conditional effect can be unknown but the action can be applied. In these kind of problems, the belief tracking cannot be fully represented only by the invariants and the literals known to hold in the belief state B. When applying action $moveC$, there is not enough information since the belief representation $D \cup R$ can not be used to compute the next belief state and the agent must consider all possible worlds. Note that, in deterministic contingent planning, the property of monotonicity is still valid for complex contingent planning problems.

D. Relevance and Causal Graph

Given a contingent planning problem $P = \langle F, I, A, O, G \rangle$, we can formalize some concepts as concepts as *cause* and *relevance* [10], to understand how uncertainty is transmitted. For a literal L, whether observable or not, we define a literal X as being the *immediate cause* (of uncertainty) of L iff $L \neq X$ and X is in the conditions C of a conditional effect $C \rightarrow L$.

The notion of *causal relevancy* is given by the transitive closure of the immediate cause relation. We define X as being causally relevant to L in P as follows: (1) if $L = X$, X is the immediate cause of L or (2) X is causally relevant to Y that is causally relevant to L. Finally, we define X to be *evidentially relevant* to L if L is causally relevant to X and X is an observable literal.

With the definitions above we can construct the *Causal Graph* that we will use to perform belief tracking on complex CP problems. The graph we consider is constructed in a slightly different way from the causal graph in [10], since it takes into account also the preconditions, making it similar to the *FD-Causal Graph* [11]. The *Causal Graph* is a directed graph with literals as nodes and a directed edge $X \rightarrow Y$ if X is causally relevant to Y.

Definition 3 (Causal Graph): Let P be a Contingent Planning Problem given by $P = \langle F, I, A, O, G \rangle$. The causal graph of P, $Cg(P)$ is a DAG (V, E) where $V = F$, containing an arc (X, Y) iff $X \neq Y$ and there is an effect $C \rightarrow Y$ and $X \in C$, or an action a with an effect $C \rightarrow Y$ such that $X \in Prec(a)$.

In Figure 3 we can see an example of a Causal Graph considering only the complex action $moveC$ of Figure 2 (bottom).

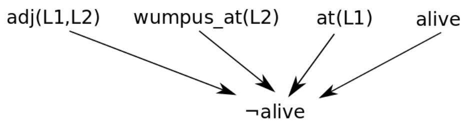

Fig. 3. Example of causal graph for $\neg alive$ for the action moveC showed in Figure 2 (bottom).

III. THE COMP2BT PLANNER

The planner COMP2BT (*Compact Compilation Belief Tracking*), is based in a two layer architecture: a layer with the translation, and a layer with the Full Observable Non-Deterministic (FOND) planner that solves the translated full observable planning problem. The aim of COMP2BT is to first make the translation (compile) and then reduce the belief state when calculating the next state using the deductive actions and axioms (compact). These deductive axioms enhance a basic translation and make it able to perform belief tracking in complex contingent planning problems. The translation layer receives as an input a contingent planning problem P and returns the translated $K(P)$ full observable version of the problem.

The underlying planner used in the experiments is a modified FF planner [12], with some differences respect to the original. First, the planner branches and recursively calls itself after every observation considering both outcomes of the observation to construct the solution. The second modification is that after selecting an action for expansion, the state s obtained is closed under the deductive axioms, meaning that the agent can reduce the belief state obtaining knowledge, thanks to the effects of the actions (*action compilation*) and the observations (*enhanced observations* see section III-C). The deductive actions are axioms modeled as actions that can be selected by the planner and expanded with no cost after executing an action, to reflect the process of deduct in belief tracking. This is the most critical difference since part of the effects is translated as axioms that have to be maintained all the time during the execution. This underlying classical planner is strongly based in the modified FF planner used by the CLG planner.

A. Translations

The translation K of a problem P is the epistemic version of the problem P. It means that the agent reasons about the knowledge of the fluents in P, instead of reasoning about the world. The meaning of the operator K is to indicate if the value of the fluent L is known. For example, the epistemic fluent KL means that the value of L is known to be true, and $K\neg L$ that L is known to be false.

The translation used in this work is based on the translation X_0 for the planner $LW1$ [13], and is similar to the translation K_0 for conformant problems [3].

Definition 4 (Translation): For a contingent problem $P = \langle F, I, A, O, G\rangle$, the translation $K_0(P)$ outputs a planning problem with full observation $K_0(P) = \langle F', I', A', G'\rangle$ and a set of axioms D' (named deductive rules):

- $F' = \{KL, K\neg L \mid L \in F\}$;
- $I' = \{KL \mid L \in I\}$;
- $G' = \{KL \mid L \in G\}$;
- A' is the set of actions $A'_A \cup O'$ where:
  1) A'_A : for every action $a \in A$, there is an action $a' \in A'_A$ where every precondition L of a, is replaced by KL, and for every conditional effect $C \rightarrow L$ of the action a, a' contains two effects $KC \rightarrow KL$ and $\neg K\neg C \rightarrow \neg K\neg L$;
  2) O' : for every observation $o : C \rightarrow L \in O$, there is a non-deterministic action $o' \in O'$ such that every precondition C of o, is replaced by $KC, \neg KL, \neg K\neg L$, and two non-deterministic effects KL and $K\neg L$;
- D' : for every $(L \supset L) \in D$ there is an axiom $K\neg C \Longrightarrow KL$, and $KL \Longrightarrow K\neg C$

In Definition 4, C is a set of literals $C = L_1, L_2, ..., L_n$, the expressions KC and $\neg K\neg C$ are abbreviations for $KL_1, KL_2, ..., KL_n$ and $\neg K\neg L_1, \neg K\neg L_2, ..., \neg K\neg L_n$ respectively. Notice that the observations have preconditions preempting the planner to choose an observation that has already been made. The deductive rules or axioms enforce the exclusivity and exhaustiveness of the uncertain variables. Sensing actions are the only non-deterministic actions in the translated problem.

The translation $K_0(P)$ is linear as it introduces no assumption or *tags* and is complete only for simple problems. To overcome this limitation, we use the Action Compilation extension proposed by [13], with some extra extensions: before performing the translation, we create the causal graph for all actions and make an analysis based on it to detect effects with uncertainty that can lead the agent towards a state with no applicable applicable (i.e. *dead end*). After that, the planner adds some deductive rules to the result of a translation $K_0(P)$.

B. Use of Causal Graph for avoiding dead-ends

Recall that a *dead-end* is a state that can not reach the goal. A *dead-end belief state* is a belief state that contains a physical *dead-end* state [14]. In a contingent planning problem, to compute the next belief state, some complex effects can lead the agent to a dead-end belief state, because uncertainty can be propagated through the uncertainty on the conditional effects. This is shown in Figure 1 by the belief states marked with an "X".

The COMP2BT planner deals with dead-end belief states by translating the original action to a new version where the effects that can lead the agent to a dead-end are removed and transformed into new axioms and actions have some additional preconditions to avoid entering these states. This is done using the *Causal Graph*.

Definition 5 (Dead-end effect): If action a contains a complex conditional effect $C \rightarrow L$, and in the causal graph there are no directed edges exiting from L, the effect is called a *Dead-end effect* because it can lead to a dead-end belief state.

To avoid the uncertain effects of *dead-end effects*, we translate the action a containing the *dead-end effect* as:

- The *dead-end effect* $C \rightarrow L$ is translated as an axiom $KC \rightarrow KL$.
- We include in the precondition of the translated action the condition $K\neg C$.

If there is an complex effect $C \rightarrow L$ and L has no directed arrows from it, it means that L does not appear in the precondition of any action or effect. No appearing in the precondition of any action means that either L is a goal condition or a literal that negates a literal $\neg L$ appearing in a precondition. To solve the first case, the translation can include a *dummy* goal action, that has the goal conditions G as preconditions. And in the second case, since the literal does not appear in any precondition or condition it means that eliminating the effect does not cause the agent to avoid reaching the goal. But since the effect is gone from the translation, the agent may try to apply the action in states where the conditional effect should be applied. To avoid these cases, the action should include in its preconditions, the conditions of the complex effect negated.

C. Enhancing observations

When an observation is made, the agent should be able to deduct that some non-observable literals are no longer known to be true or false, even if they were uncertain.

With the evidential relevance we express that a observable literal L is relevant for a literal X that appears in the body of an effect $C \rightarrow L$. But this is not sufficient to deduct information about X from L, because some other literals can be relevant to the same literal observed. As an example consider a $1 \times N$ grid where a robot can move to an adjacent cell. If the agent performs an observation, and detects a wall at the left, the agent must deduct then that he is in the leftmost location, and similarly when the agent senses a wall in the right, the agent must know that he reached the goal. But if the agent does not sense a wall at the right, it cannot deduct in which position is, using only the evidential relevance, because more than one position can yield the same observation. In fact, the agent does know that he is not at the goal position. We extend the definition of evidential relevance to reflect this:

Definition 6 (Inversed Evidential Relevance): X is inversed evidentially relevant to Y if X is an observable literal, and Y is causally relevant to the complement of X, $\neg X$.

Using this definition, we can add some axioms to the translation $K_0(P)$ to be considered after every observation in order to perform a better belief tracking in complex contingent planning problems:

Definition 7 (Enhanced Observations): If X is an observable literal, and X is inversed evidential relevant to Y, when performing an observation that yields the literal X, the agent also knows $\neg Y$.

IV. EXPERIMENTS

We tested the proposed algorithm over a selection of contingent domains, comparing it with the CLG planner [4] and PO-PRP planner [6], both considered the state-of-the-art of contingent planning. The efficiency of the planner is given by the time it takes to compute a complete plan, and the quality is based on the total number of actions in the solution. All experiments were performed in a Linux machine with an 1.33 Ghz processor. The times were limited to one hour for each one of the instances of every problem and to 1GB RAM memory.

The benchmark domains used in the experiments are: Color Balls (*cballs*), Wumpus World, Doors, Localize and Clog [4]. They are all considered as simple contingent planning problems, except for the Localize domain and the modified versions of Wumpus and Doors (as explained below) in which actions contain complex conditional effects. These domains are included in the CLG Benchmark package, except for the modified Wumpus and Doors. Color Balls involves searching for a set of balls of different colors and throwing them in the correct garbage can. Doors is a domain where an agent must move in a grid with hidden doors. Localize consists in a robot that must locate itself in a known map. And Clog is a logistic problem.

To demonstrate how the translation step of COMP2BT overcomes the disadvantages of other translations, we also used a variation with complex conditional actions for the Doors and Wumpus domains [15]. In the Wumpus-D (Wumpus with dead-ends) the safe preconditions are removed, and it is added an special effect that makes the agent die if it enters in a cell that is not safe (i.e. it contains a Wumpus or a pit) creating a dead-end. In the new Doors-D domain, we added an effect that makes the robot break if it tries to enter through a door that is not opened, removing the *opened* precondition. These new variations are more challenging because they involve complex conditional actions.

Table I shows the comparison of the three planners CLG, PO-PRP and COMP2BT: For simple contingent planning problems (instances from wumpus-5 to doors-11), in general PO-PRP obtains the best results followed closely by COMP2BT, even if in some instances PO-PRP runs out of memory. For those problems, PO-PRP and COMP2BT are orders of magnitude better than CLG. Due to the fact that PO-PRP builds partial policies instead of trees, its plans have a better quality. A partial policy is a policy defined only for some *relevant* literals of the belief state, instead of a policy defined for all literals.

In domains with complex conditional actions, COMP2BT outperforms both PO-PRP and CLG, being capable of solve more problems in less time.

V. DISCUSSION

a) Linear translation: One reason for COMP2BT to outperform CLG is that it uses a quadratic translation based on tags and merges. Roughly speaking, this means that it considers every possible initial state, and includes in the translation new literals KL/t meaning that the literal L is known to be true if in the initial state t was true. Since in the CLG planner there is a tag for every possible initial state, the translated problem increases quadratically in size, and actions have more effects. As an example, a problem with 10 fluents translated with a linear translation can increase to

TABLE I

TIMES IN SECONDS OBTAINED FOR THE THREE DIFFERENT PLANNERS (CLG, COMP2BT AND PO-PRP) TO COMPUTE THE FULL CONTINGENT PLANS. *T* STANDS FOR TIME-OUT AND *MO* FOR MEMORY OUT.

	CLG		**COMP2BT**		**PO-PRP**	
Problem	**Time**	**Size**	**Time**	**Size**	**Time**	**Size**
wumpus-5	1.28	754	0.229	435	0.48	197
wumpus-7	30.52	6552	2.968	4115	3.5	631
wumpus-10	T		76.809	63297	24.36	1471
wumpus-15	T		T		162.6	7787
cballs4-1	0.62	295	0.098	270	0.08	271
cballs4-2	60.84	20050	5.712	15149	5.54	12360
cballs4-3	T		448.889	892590	MO	
cballs10-1	337	4445	12.325	4799	5.02	3849
cballs10-2	T		T		MO	
clog-7	0.28	210	0.24	190	0.24	93
clog-huge	490.46	37718	39.985	28972	7.76	15799
doors-7	18.76	2153	0.745	2241	0.682	1282
doors-9	1294.64	46024	19.128	46656	18.3	23897
doors-11	T		659.907	1208947	MO	
wumpus-d5	T		0.259	491	1.48	253
wumpus-d7	T		3.494	4589	17.86	947
wumpus-d10	T		88.562	69708	336.66	2497
doors-d5	T		0.084	173	0.14	107
doors-d7	T		1.142	2584	1.06	1304
doors-d9	T		31.401	53217	28.62	22570
doors-d11	T		MO		MO	
localize-5	0.4	115	0.094	106	2139.76	8399
localize-7	2.58	241	0.27	239	T	
localize-9	11.8	409	0.737	416	T	
localize-11	180.68	617	2.246	688	T	
localize-13	MO	MO	4.948	969	T	

$2 \times 10 = 20$ fluents. But with a quadratic translation it can grow to $(2 \times 10)^2 = 400$ fluents. The CLG planner takes more time updating belief states because it has to consider a larger number of epistemic fluents. Comp2BT and PO-PRP use a linear translation that results in a smaller translated problem with less literals. Both planners are better than CLG in all benchmarks. PO-PRP and COMP2BT use linear translation and get better results in most domains.

b) Detecting complex effect dead-ends: In domains with effects leading to dead-ends, like *wumpus-d* and *doors-d*, COMP2BT can detect this kind of complex conditional actions and translate them in a version with no dead-ends, which effectively reduces the branching factor. However CLG fails to deal with this problems. CLG may fail when the local search (hill climbing) fails and then it has to restart with a complete heuristic search. Surprisingly PO-PRP can solve them with worse times than COMP2BT.

c) Dealing with complex effects: In the *localize* domain actions are modeled with conditional actions. In this case to keep track of beliefs during the search it is harder. CLG can solve easily this kind of problems since it considers all possible initial states. But since PO-PRP uses a linear translation, it cannot deal with the complex actions, and fails. Compt2BT uses a translation were actions yield more information and observations are enhanced with axioms. Compt2BT can solve this kind of problems in a better time than both CLG and PO-PRP.

VI. CONCLUSIONS AND FUTURE WORK

In sum, belief tracking for planning with partial observation is an open and challenging problem in Automated Planning. In this paper we have shown some causality relations among variables in a contingent planning problem that can be exploited by a planner. We have shown a translation that is correct and uses information concerning these relations to keep track of beliefs while being linear in size. We tested the proposed planner COMP2BT on 7 different contingent planning domains. The results show that our planner can solve more problems than the two planners considered the state-of-the-art for contingent planning, presenting also better times (up to 2 orders of magnitude faster than CLG). Some challenges we would like to address in the future is to use the causal graph to detect traps of dead-ends, while keeping the translation small.

ACKNOWLEDGMENT

The first author is supported by funding from the CAPES Foundation within the Ministry of Education, Brazil. The second author is supported by the grant #2015/01587-0, Sao Paulo Research Foundation (FAPESP), and the CNPq . Also we would like to thank Christian Muise for the help he provided to use the PO-PRP planner.

REFERENCES

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: theory & practice*. Access Online via Elsevier, 2004.

[2] J. Rintanen, "Complexity of planning with partial observability." in *ICAPS*, 2004, pp. 345–354.

[3] H. Palacios and H. Geffner, "Compiling uncertainty away in conformant planning problems with bounded width," *Journal of Artificial Intelligence Research*, vol. 35, no. 2, p. 623, 2009.

[4] A. Albore, H. Palacios, and H. Geffner, "A translation-based approach to contingent planning." in *IJCAI*, 2009, pp. 1623–1628.

[5] J. Hintikka, *Knowledge and belief: an introduction to the logic of the two notions*, ser. Contemporary philosophy. Cornell University Press, 1962.

[6] C. Muise, V. Belle, and S. A. McIlraith, "Computing contingent plans via fully observable non-deterministic planning," 2014.

[7] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3, pp. 189–208, 1972.

[8] M. Helmert, "Concise finite-domain representations for pddl planning tasks," *Artificial Intelligence*, vol. 173, no. 5, pp. 503–535, 2009.

[9] B. Bonet and H. Geffner, "Planning under partial observability by classical replanning: Theory and experiments," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, 2011, p. 1936.

[10] ——, "Belief tracking for planning with sensing: Width, complexity and approximations," *Journal of Artificial Intelligence Research*, pp. 923–970, 2014.

[11] M. Helmert, "The Fast Downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[12] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[13] B. Bonet and H. Geffner, "Flexible and scalable partially observable planning with linear translations," in *Proc. 28th AAAI Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 2014.

[14] A. Albore and H. Geffner, "Acting in partially observable environments when achievement of the goal cannot be guaranteed," in *Proc. of ICAPS Workshop on Planning and Plan Execution for Real-World Systems*, 2009.

[15] R. I. Brafman and G. Shani, "A multi-path compilation approach to contingent planning." in *AAAI*, 2012.