Computing Contingent Plans via Fully Observable Non-Deterministic Planning

Christian Muise and **Vaishak Belle** and **Sheila A. McIlraith**
Department of Computer Science
University of Toronto, Toronto, Canada.
{cjmuise,vaishak,sheila}@cs.toronto.edu

Abstract

Planning with sensing actions under partial observability is a computationally challenging problem that is fundamental to the realization of AI tasks in areas as diverse as robotics, game playing, and diagnostic problem solving. Recent work on generating plans for partially observable domains has advocated for online planning, claiming that offline plans are often too large to generate. Here we push the envelope on this challenging problem, proposing a technique for generating conditional (aka contingent) plans offline. The key to our planner's success is the reliance on state-of-the-art techniques for fully observable non-deterministic (FOND) planning. In particular, we use an existing compilation for converting a planning problem under partial observability and sensing to a FOND planning problem. With a modified FOND planner in hand, we are able to scale beyond previous techniques for generating conditional plans with solutions that are orders of magnitude smaller than previously possible in some domains.

1 Introduction

An agent *planning* to achieve a goal in a *partially observable* environment with *sensing* actions (PPOS) has the option of choosing how to act *online* by interleaving planning, sensing, and acting; or choosing how to act *offline* by generating a plan with decision points predicated on sensing outcomes. In this paper we investigate the latter. In particular, we examine the problem of generating conditional plans for planning problems with incomplete information about the initial state, deterministic actions, and sensing actions. In this work, we use the terms "offline planning" and "conditional planning" interchangeably to always refer to the offline generation of contingent plans; the online variant will be referred to as "online contingent planning." Our focus here is on a particular class of problems where the initial state specification includes a set of state constraints called *state invariants*. These are commonly used to model the behaviour of a device, or physical environments with which the agent is interacting. We further assume that uncertainty decreases monotonically, *i.e.* once a property of the world is known, it can change but cannot become unknown again.

There are merits and shortcomings to both online and offline planning in this context. Online contingent plans are generally easier to compute since integrating online sensing with planning eliminates the need to plan for a potentially exponential (in the size of relevant unknown facts) number of contingencies. In the absence of deadends, online contingent planning can be fast and effective. Recent advances include CLG and CLG+ (Albore, Palacios, and Geffner 2009; Albore and Geffner 2009), *K*-Planner (Bonet and Geffner 2011), and SDR (Brafman and Shani 2012).

In contrast, planning offline constructs conditional plans with decision points for sensing outcomes and guarantees that the goal will be achieved if it is possible to do so. The plan is larger than an online plan but has the merit that it is generalized to deal with alternative sensing outcomes. Indeed plan existence for conditional planning is 2-EXP-complete (Rintanen 2004; Baral, Kreinovich, and Trejo 2000). More importantly, because offline planners are able to search and deliberate, they have the capacity to avoid deadends, and also to support the generation of optimized high quality plans. Some early conditional planners were based on partial order planning (e.g., CNLP (Peot and Smith 1992), Cassandra (Pryor and Collins 1996)) and Graphplan (e.g., SGP (Weld, Anderson, and Smith 1998)). MBP (Bertoli et al. 2001) and BBSP (Rintanen 2004) are more recent BDD-based model checking planners. Planners based on heuristic search include Contingent-FF (Hoffmann and Brafman 2005), POND (Bryce, Kambhampati, and Smith 2006), and most recently CLG which has an offline variant (Albore, Palacios, and Geffner 2009). Finally, the conditional plan we consider is similar to the "execution structure" of Kuter et al.'s "conditionalized plan" (2007), but differs in construction and generality of what the plan represents. Experimental results reported in the literature appear to indicate that CLG represents the state of the art in terms of scalability of offline conditional planning.

In this paper we present PO-PRP, a conditional planner. PO-PRP plans achieve the goal for all consistent sequences of observations for which a solution exists. The key to our planner's success is its reliance on state-of-the-art techniques for fully observable non-deterministic (FOND) planning. In particular, we use an existing compilation by Bonet and Geffner (Bonet and Geffner 2011) (henceforth BG) for converting a PPOS problem to a FOND planning problem. All actions are treated as deterministic with the exception of sensing actions, which are encoded as non-deterministic actions. We then modify a state-of-the-art FOND planner, PRP (Muise, McIlraith, and Beck 2012), to compute strong cyclic plans in the form of policies, which we roll-out into plans represented as DAGs. We address a number of critical challenges, leading to a conditional planner that is able to scale beyond previous techniques for offline planning and that is able to compute solutions that are orders of magnitude smaller than state-of-the-art CLG in some domains.

2 Contingent Planning via FOND Planning

In this section, we formulate the PPOS problem, and then review the translation of PPOS problems to FOND ones.

Syntax and Interpretation. Following BG, we specify the problem in a STRIPS-like language, where actions can additionally have *conditional effects*. Formally, a PPOS domain is a tuple $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{F}$ is the set of fluent atoms, $\mathcal{A}$ is the set of actions, $\mathcal{O}$ is the set of observations, $\mathcal{I}$ is a set of clauses over $\mathcal{F}$ that determines the initial state, and $\mathcal{G}$ is a conjunction of atoms over $\mathcal{F}$ determining the goal condition. The specification is interpreted over a set of (world) *states*, which are essentially truth valuations to the atoms in $\mathcal{F}$. We say a literal l *holds* in a state s iff s assigns l to be true. This is extended for connectives in an obvious way. In particular, since $\mathcal{I}$ is a set of clauses, $\mathcal{I}$ would hold in a number of states; the *belief state b* is the set of all (world) states where $\mathcal{I}$ holds. By extension, we say a formula α holds in b iff α holds in every state $s \in b$.

For $a \in \mathcal{A}$, we let $\textsc{Pre}(a)$ be a conjunction of atoms to denote its preconditions, and $\textsc{Eff}(a)$ be a set of pairs $\langle c, l \rangle$ to capture its conditional effects. Observations $o \in \mathcal{O}$ are of the form $\langle c, l \rangle$ with the understanding that when c is true, o informs the agent about the truth of l. In practice, this is achieved by treating observations o as a separate category of actions which have c as the precondition, and l as the effect.[1] We say an action a is *applicable* in s iff $\textsc{Pre}(a)$ holds in s. Analogously, we say a is *applicable* in b iff a is applicable in every $s \in b$. On performing a in b, a *successor* belief state b' is defined by performing a in each $s \in b$. On performing an observation $o = \langle c, l \rangle$ in b, the successor belief state b' is the *maximal* set of states in b agreeing on l. By extension, a sequence $a_0 \cdot a_1 \cdots a_k$, possibly involving observations, is applicable in b if a_0 is applicable in b, resulting in a successor belief state b_1, and inductively, a_i is applicable in b_i, ultimately resulting in b_k. A belief state b' is said to be *reachable* from b if there is some sequence of actions and observations that when applied to b results in b'.

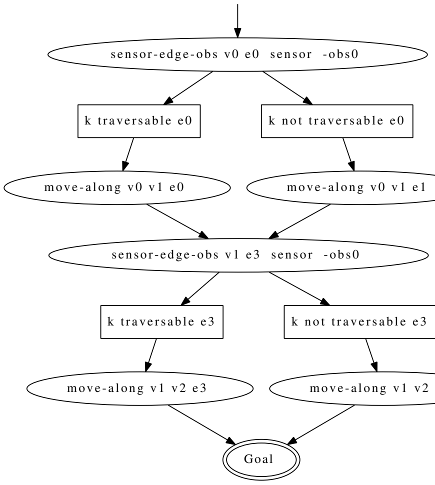

Figure 1: Example solution to the CTP. Circles represent the action to take or observation to be made while boxes contain edge labels indicating the sensing outcome that has occurred.

Solutions. Generally with PPOS problems, a solution is rarely a simple sequence of actions, because the agent may need to perform a number of sensing actions to determine aspects of the world based on which further actions can be taken. Therefore, a solution to a PPOS problem is a *policy* Π that is best viewed as a branching structure, usually called a *conditional plan*, induced by the outcomes of sensing actions (Geffner and Bonet 2013). Equivalently, one may view Π as a partial function from belief states to actions (Geffner and Bonet 2013). Such a function would then advise the subsequent action to be taken based on the actual observation. Finally, we say that Π *solves* the PPOS problem $\mathcal{P}$ iff the executions advised by Π are applicable in the belief state b for $\mathcal{P}$, and they result in belief states b^* where the goal condition $\mathcal{G}$ holds.

While conditional plans are usually *trees*, in this work we reduce the redundancy by representing many belief states as a single node in a DAG. Crucially, a node may correspond to a partial belief state, capturing many configurations of the agent's belief. As an example PPOS problem, consider the Canadian Traveller's Problem (CTP) where an agent must navigate a city with roads that may or may not be traversable (due to large amounts of snow fall). Sensing the status of a road can only be done when the agent is adjacent to it. If we consider the class of maps that have a pair of roads between every two adjacent locations in a chain, exactly one of which is traversable, then an obvious strategy presents itself: sense one of the two roads that leads to the next location; if it is traversable then take it; otherwise take the other road. Naive solutions to the problem are exponential in size (every new location has a new choice of two roads), but in this work we strive to generate plans such as in Figure 1.

Compiling Away Uncertainty. Computing successor belief states b', often called *belief tracking*, is non-trivial. Note, for example, that applying actions and observations over all world states (as needed for obtaining b') is clearly exponential in $|\mathcal{F}|$. In recent work, belief tracking is shown to be tractable against the *contingent width* of a PPOS problem, which is claimed to typically be small for most real-world domains (Albore, Palacios, and Geffner 2009). Thus, it follows that for many interesting domains, belief tracking is *quadratic* in $|\mathcal{F}|$. Nonetheless, even a quadratic fac-

[1] Our sensor model differs slightly from BG in that they assume sensing actions are triggered as soon as the preconditions hold. Our planner, on the other hand, chooses to apply sensing actions when needed, just like ordinary physical actions.

tor is computationally challenging in large domains. A second key result is that a PPOS problem can be translated into a FOND one (Albore, Palacios, and Geffner 2009). This supports the exploitation of state-of-the-art heuristic search techniques for classical planning. Of course, the quadratic belief tracking problem has a natural analogue in the translation, and so computational issues persists.

Interestingly, BG show that when we further restrict ourselves to the so-called *simple* PPOS problems, belief tracking is *linear* in $|\mathcal{F}|$. Informally, simple problems are motivated by the observation that many domains can be characterized by *state invariants*, and actions typically do not depend on fluents whose values are unknown. In precise terms, invariant clauses are clauses that hold in every world state (Helmert 2009), and often represent multivalued functional fluents. The second assumption in simple problems is that the body of conditional effects cannot mention *hidden* literals, that is, literals l such that $\mathcal{I} \not\models l$ and $\mathcal{I} \not\models \neg l$. Hidden literals, however, may appear in the preconditions of actions and may also appear in the effects of actions. Following BG, we have:

Definition 1. (Simple PPOS Problems.) A PPOS problem $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ is *simple* if the non-unary clauses in $\mathcal{I}$ are invariant, and no hidden fluent appears in the body of a conditional effect.

Most significantly, features of simple problems include:

- **Effective Characterization:** Suppose $\mathcal{I}^*$ are the non-unary clauses in $\mathcal{I}$ and b is the belief state for $\mathcal{I}$ in $\mathcal{P}$. If b^* is reachable from b and the literals in S are known to hold in b^*, then b^* is completely characterized by $\mathcal{I}^* \cup S$.
- **Monotonicity:** Suppose b' is reachable from b and l is known in b'. If b'' is reachable from b', then l is also known in b''.

From PPOS to FOND. Adapting the work of BG, we now present a translation from simple PPOS problems to FOND ones, where the non-deterministic actions are from the sensor model. The main idea is to replace every literal l with fluent atoms Kl and $K\neg l$, which denotes knowing that l is true vs. false.

Definition 2. Suppose $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ is a simple PPOS problem. We define $K'(\mathcal{P}) = \langle \mathcal{F}', \mathcal{A}', \mathcal{I}', \mathcal{G}' \rangle$ as a fully-observable non-deterministic planning problem where

1. $\mathcal{F}' = \bigcup_{l \in \mathcal{F}} \{Kl, K\neg l\}$;
2. $\mathcal{A}'$ is the set of actions $\mathcal{A}'_{\mathcal{A}} \cup \mathcal{A}'_{\mathcal{O}} \cup \mathcal{A}'_{\mathcal{V}}$, where,
   (a) $\mathcal{A}'_{\mathcal{A}}$: for every $a \in \mathcal{A}$, there is an action $a' \in \mathcal{A}'_{\mathcal{A}}$ such that if $l \in \textsc{Pre}(a)$ then $Kl \in \textsc{Pre}(a')$, and if $\langle c, l \rangle \in \textsc{Eff}(a)$ then $\{\langle Kc, Kl \rangle, \langle \neg K\neg c, \neg K\neg l \rangle\} \subseteq \textsc{Eff}(a')$;
   (b) $\mathcal{A}'_{\mathcal{O}}$: for $o = \langle c, l \rangle \in \mathcal{O}$, there is an action $a' \in \mathcal{A}'_{\mathcal{O}}$ such that $\textsc{Pre}(a') = Kc \wedge \neg Kl \wedge \neg K\neg l$ with two possible non-deterministic effects, $\textsc{Eff}_1(a') = \{\langle \top, Kl \rangle\}$ and $\textsc{Eff}_2(a') = \{\langle \top, K\neg l \rangle\}$;
   (c) $\mathcal{A}'_{\mathcal{V}}$: for every $(c \supset l) \in \mathcal{I}^*$, there is an action $a' \in \mathcal{A}'_{\mathcal{V}}$ such that $\textsc{Pre}(a') = Kc$ and $\textsc{Eff}(a') = \{\langle \top, Kl \rangle\}$;
3. $\mathcal{I}' = \{Kl \mid l \in \mathcal{I}\}$; and
4. $\mathcal{G}' = \{Kl \mid l \in \mathcal{G}\}$;

We use the notation Kc and $\neg K\neg c$ when $c = l_1 \wedge \ldots \wedge l_k$ to mean $(Kl_1 \wedge \ldots \wedge Kl_k)$ and $(\neg K\neg l_1 \wedge \ldots \wedge \neg K\neg l_k)$ respectively. The key differences between Definition 2 and BG are that (1) we use $\neg Kl \wedge \neg K\neg l$ as part of the precondition for a sensing action and (2) we assume that the agent is free to choose when a sensing action occurs. The former helps avoid inconsistent beliefs during search, and the latter is a restriction that can be removed easily if so desired.

The intuition is this: since unary clauses in $\mathcal{I}$ are clearly known, we index K to these literals in $\mathcal{I}'$; $\mathcal{A}'_{\mathcal{V}}$, then, represents the invariants in $\mathcal{I}$. $\mathcal{A}'_{\mathcal{A}}$ determines when the agent would know the effect of an action. Sensing actions are the only non-deterministic actions in the translated FOND problem, and these indicate that at execution time, the agent would either come to know that the literal is true or would come to know that the literal is false. In practice, on performing such a non-deterministic action, we will be tracking two possible successors for a world state corresponding to each outcome. Finally, the goal state is one where every literal in the original goal $\mathcal{G}$ is known to be true.

Analogous to the notion of a policy Π at the level of belief, the solution to a FOND problem is a policy (i.e. a partial function) Π' that maps (world) states to actions. Applicability and goal reachability are defined for the FOND problem also in an analogous manner. Finally, the translation is justified by means of a *soundness* and *completeness* result by BG: a policy Π' obtained for $K'(\mathcal{P})$ can be converted to a policy Π for $\mathcal{P}$, and vice versa.

3 Approach

The modified BG translation described in Section 2 provides a FOND encoding of our PPOS planning problem, $K'(\mathcal{P})$, in which sensing actions are encoded as non-deterministic actions. Our planner, PO-PRP, leverages state-of-the-art FOND planner PRP (Muise, McIlraith, and Beck 2012), extended with conditional effects (Muise, McIlraith, and Belle 2014) and is modified to address three critical challenges for dealing with PPOS problems. FOND planners are generally predicated on an assumption of *fairness* – if an action is executed infinitely many times, every non-deterministic outcome will occur infinitely often (Cimatti et al. 2003). A key challenge was addressing the fundamental violation of fairness by sensing actions. A second challenge was to suitably handle the computation of indirect effects of actions (ramifications) resulting from the state invariants. A third critical element was leveraging a technique used in PRP, strong cyclic detection, to produce compact conditional plans.

3.1 From PRP to PO-PRP

We review and elaborate upon key elements of PRP that are essential to PO-PRP, with particular focus on one aspect of the planner referred to as *strong cyclic detection* (SCD). Our interest in the SCD procedure of PRP stems from the fact that we can leverage it for computing a compact conditional plan given the policy PO-PRP generates (cf. Section 3.4).

Problem Specification. PO-PRP takes as input a simple PPOS problem $\mathcal{P}$ and outputs a conditional plan. The first step of PO-PRP is to apply the augmented BG translation described in Section 2 to $\mathcal{P}$, generating the associated FOND problem $K'(\mathcal{P})$. From this point, the problem specification is identical to that of PRP, which can be found in (Muise, McIlraith, and Beck 2012). The details are not necessary to the results that follow. The translated planning problem $K'(\mathcal{P})$ is input as a PDDL file to PO-PRP, which uses the SAS$^+$ representation (Helmert 2009). A significant property of PRP (likewise PO-PRP), is that states are represented as *partial states* – a subset of literals that is interpreted as a conjunctive formula, compactly representing a family of states. We exploit this compact representation to generate small-sized conditional plans (cf. Section 3.4).

General Approach. Cimatti et al. (2003) identify three types of plans for FOND problems: *weak*, *strong*, and *strong cyclic*. Intuitively, a *weak plan* corresponds to an "optimistic plan" that reaches the goal under at least one possible set of action outcomes of the actions in the plan. A *strong plan* corresponds to a "safe plan" and is a closed policy that achieves the goal in a finite number of steps while never visiting the same state twice. Often, however, weak plans are not acceptable and strong plans do not exist. As a viable alternative, a *strong cyclic plan* is a closed policy with the property that every reachable state will eventually reach the goal via the policy under an assumption of fairness.

PRP creates a strong cyclic plan in the form of a policy that maps states to actions. The representation of this policy, however, is non-standard. Rather than storing an explicit mapping of complete states to actions, PRP creates a set of *condition-action pairs* P, each of the form $\langle p, a\rangle \in P$ with p a partial state. In order to return a unique action for a given state s, there is a total ordering over the pairs that allows us to use P and simply return the action in the "most preferred" pair $\langle p, a\rangle$ such that $s \models p$ (ordered, for example wrt distance from goal). We will use $P(s)$ to designate the pair that is most preferred, and we say that P *handles* state s if P returns some pair. PRP's core algorithm is as follows:

1. Let $Open = \{s_0\}$ and $Cls = \emptyset$;
2. Select and move a state s from $Open$ to Cls such that,
   - (i) If $P(s) = \bot$, compute a classical plan for s and augment the policy with the result
   - (ii) If $P(s) = \langle p, a\rangle$ and $a \in \mathcal{A}_{\mathcal{O}}$, add to $Open$ the states $\{Prog(s, a, \textsc{Eff}_1(a)), Prog(s, a, \textsc{Eff}_2(a))\} \setminus Cls$;[2]
   - (iii) If $P(s) = \langle p, a\rangle$ and $a \notin \mathcal{A}_{\mathcal{O}}$, add to $Open$ the state $Prog(s, a, \textsc{Eff}(a))$ if it is not in Cls;
   - (iv) If $P(s) = \bot$, process s as a *deadend*;
3. If $Open$ is empty, return P. Else, repeat from step 2;

Note that step 2(i) is essentially computing a weak plan by way of solving a classical planning problem. In Section 3.3, we describe more precisely how this search procedure is modified to incorporate indirect effects of actions (ramifications) found in the translated PPOS problems.

The PRP planner has a number of components (described in (Muise et al., 2012)) that are key to the success of PO-PRP, including (1) deadend detection, generalization, and avoidance, (2) stopping conditions for weak plans, and (3) conditions for terminating the simulation of action effects (replacing steps 2(ii) and 2(iii)). It is this final phase that we discuss below.

Strong Cyclic Detection. PRP has the facility to "mark" certain pairs in the policy P to indicate that if $\langle p, a\rangle$ is marked, then P is a strong cyclic plan for every state s where $P(s) = \langle p, a\rangle$. This feature allows PRP to forgo steps 2(ii) and 2(iii) if the pair is marked – instead of expanding the state with every possible outcome, the state s is considered handled completely and the process continues. While reproducing the full strong cyclic detection (SCD) procedure is beyond the scope of this paper, later we do rely on one aspect of SCD for the export of conditional plans.

PRP will only allow $\langle p, a\rangle$ to be marked if P is guaranteed to return a marked pair $P(s') = \langle p', a'\rangle$ for every state s' that could be reached by the pair $\langle p, a\rangle$. In other words, we have the following property (following Definition 7 and Theorem 3 of (Muise, McIlraith, and Beck 2012)):

Proposition 1. The SCD procedure of PRP ensures that a pair $\langle p, a\rangle$ of policy P is marked only if for every state s such that $P(s) = \langle p, a\rangle$ and every non-deterministic effect $\textsc{Eff}$ of action a, the pair $P(Prog(s, a, \textsc{Eff}(a))) = \langle p', a'\rangle$ is marked or p' is a goal.

3.2 Violating Fairness with Sensing Actions

Following Cimatti et al. (2003), solutions to non-deterministic planning problems typically rely on an assumption of fairness with respect to non-deterministic actions: if an agent executes a non-deterministic action infinitely many times, every one of its outcomes will occur infinitely often. Unfortunately, in our current PPOS problems, the only non-deterministic actions are sensing actions and they are decidedly *unfair*. The outcome of a sensing action will reflect the state of the world, which is unchanging unless changed by an action. Fortunately, since we assume that the state of the world only changes as the result of the actions of the plan (i.e., there are no exogenous actions) then once a fluent has been sensed, the planner should not be compelled to execute the sensing action again because the value of the fluent is known.

This is realized by our problem encoding, $K'(\mathcal{P})$, together with the assumption of monotonicity of knowledge. Recall that a precondition of any sense action is that the outcome is currently not known and the effect is that one of the literal or its negation is known, violating the precondition for the sense action to be executed again. This together with monotonicity ensures that a particular ground sensing action can only ever be executed at most once during online execution. As a result, the space of solutions to the FOND planning problem, and subsequently the space of conditional plans, will *never contain a cycle with a sensing action*. This key property allows us to preserve the properties of our FOND planner, despite the inherent unfairness of sensing actions.

[2] $Prog(s, a, e)$ is the *progression* of s wrt a's effect e and is defined as usual for planning with conditional effects (Reiter 2001).

3.3 Computing Ramifications

When state invariants or state constraints are combined with an action theory, they result in indirect effects of actions – further consequences of an action that result from enforcing the truth of the state invariants. Understanding what effects should be generated and how is an instance of the *ramification problem*, a well-studied problem in Knowledge Representation (KR). A variety of solutions have been proposed for dealing with this issue, including compilation of these indirect effects or ramifications into further direct effects of actions, representing indirect effects as actions that are triggered by states, representing indirect effects as further axioms or derived predicates, etc. (e.g., (Pinto 1999; McIlraith and Scherl 2000; Strass and Thielscher 2013)).

Our BG inspired encoding, $K'(\mathcal{P})$, captures these indirect effects as a distinguished set of so-called invariant actions, $\mathcal{A}'_{\mathcal{V}}$. Intuitively, following the execution of a regular action, these invariant actions should be executed as bookkeeping actions to compute all indirect effects of the action. Unfortunately, if left unconstrained, this can lead to extensive wasted search effort in parts of the state space that will never be reached during execution, as well as the discovery of meaningless deadends that PO-PRP will try to avoid.

We elected to continue to compute the effects of invariants via the application of actions so that the heuristics PO-PRP uses to compute a weak plan would inform the computation of ramifications. Further, PRP's/PO-PRP's use of regression to compute condition-action pairs would become prohibitively complex were we instead to compile the ramifications into extra effects of actions. Inspired by KR research on the ramification problem, we modified the search procedure of PO-PRP to compute the indirect effects of actions by applying invariant actions at every node in the search until quiescence. That is, until no more invariant actions are applicable. To avoid state explosion, we enforce an arbitrary ordering over the application of invariant actions. Given the existing restrictions for simple domains, enforcing the order does not alter the state that the planner reaches once there are no more applicable invariant actions. This is because there is a unique interpretation in these simple domains, and because knowledge is assumed to accumulate monotonically.

3.4 Exporting a Conditional Plan

Like PRP, PO-PRP produces a policy in the form of condition-action pairs, which, in principle, could be used if the belief of the agent is maintained in its compiled form. Since this cannot be guaranteed, we convert the policy into a form more traditionally used for PPOS solutions: a conditional plan. The policy quite often will be far smaller than the conditional plan it implicitly represents, but there is merit to producing one: it helps in verifying PO-PRP's final solution, and it provides a plan in a form suitable for execution by an agent that does not maintain its belief in a compiled form (e.g., a simple controller).

While conditional plans are typically tree structures, PO-PRP conditional plans are constructed more compactly as DAGs. Nodes in the DAG correspond to either actions drawn from $\mathcal{A}'_{\mathcal{A}}$, or decision points – sensing actions drawn from $\mathcal{A}'_{\mathcal{O}}$ whose outgoing edges denote the possible observations. There are three additional distinguished node types: a single source node denoting the first action of the plan, deadend nodes where no plan exists, and goal nodes.

To convert PO-PRP's policy P to a conditional plan, we use an approach that essentially simulates P on every possible action outcome. Whenever we see a *repeated* state, we need not continue expanding the conditional plan. Intuitively, the procedure is as follows:

1. Let $Open = \{s_0\}$, $Cls = \emptyset$, and $\langle N, E\rangle = \langle \{s_0\}, \emptyset\rangle$;
2. Select and move a state s from $Open$ to Cls such that,
   (i) Let $P(s) = \langle p, a\rangle$ and if $a \in \mathcal{A}'_{\mathcal{O}}$ let $Succ = \{Prog(s, a, \text{Eff}_1(a)), Prog(s, a, \text{Eff}_2(a))\}$ (otherwise let $Succ = \{Prog(s, a, \text{Eff}(a))\}$);
   (ii) Add $Succ \setminus Cls$ to N and $Open$;
   (iii) Add (s, s') to E for every s' in $Succ$;
3. If $Open$ is empty, return $\langle N, E\rangle$. Else repeat from 2.

We further modify the DAG $\langle N, E\rangle$ so that every state $s \in N$ is labelled with the action a from the pair $P(s) = \langle p, a\rangle$, and every exit edge from a state labelled with an action $a \in \mathcal{A}'_{\mathcal{O}}$ is labelled with the appropriate observation outcome. Additionally, we reduce the graph by merging any state labelled with an invariant action into its successor.

The similarity to PRP's general approach (described above) is not coincidental: both approaches operate by enumerating the reachable states of P. To avoid a full state explosion of reachable states we can appeal to PO-PRP's SCD procedure and create a more compact conditional plan. The key is to replace every state $Prog(s, a, \text{Eff}_i(a))$ in $Succ$ on line 2(i) with p whenever $P(Prog(s, a, \text{Eff}_i(a))) = \langle p, a\rangle$ is marked as strong cyclic. This means that the closed list Cls will contain both complete and partial states of the FOND encoding, $K'(\mathcal{P})$. The partial states p contain only the relevant information for a strong cyclic solution to exist with P, and this allows us to reuse parts of the conditional plan for all states s such that $P(s) = \langle p, a\rangle$. As an example, consider the CTP problem from earlier. The states computed on the modified line 2(i) would be partial states that retain the belief about future roads, but forgo knowledge about roads observed in the past. In this way, the agent can use the same conditional plan regardless of how it has travelled so far.

Theorem 1. Given a simple PPOS problem $\mathcal{P}$, the conditional plan produced by PO-PRP is sound.

This follows from Proposition 1 and the soundness and completeness of the BG translation from $\mathcal{P}$ to $K'(\mathcal{P})$; the soundness of the partial policy produced by PO-PRP with respect to the class of FOND problems represented by $K'(\mathcal{P})$; and by the correctness of the transformation of the partial policy with respect to $K'(\mathcal{P})$ into a compact conditional plan with respect to the original PPOS problem.

Theorem 2. Given a simple PPOS problem $\mathcal{P}$, PO-PRP will return a conditional plan if one exists.

This similarly follows from the soundness and completeness of the BG translation of $\mathcal{P}$ into a FOND problem, together with the property, inherited from PRP, that if a strong cyclic solution exists for $K'(\mathcal{P})$, then PO-PRP will compute it and export it as a conditional plan with respect to $\mathcal{P}$.

Problem	Time (seconds)			Size (actions + sensing)			PO-PRP Policy Size		
	CLG	PO-PRP$^-$	PO-PRP	CLG	PO-PRP$^-$	PO-PRP	all	strong	used
cballs-4-1	0.20	0.03	**0.02**	343	365	**261**	150	24	125
cballs-4-2	19.86	**0.6**	0.67	22354	18643	**13887**	812	46	507
cballs-4-3	1693.02	**87.03**	171.28	1247512	899442	**671988**	3753	81	1689
cballs-10-1	211.66	1.63	**1.57**	4829	5096	**4170**	1139	20	815
cballs-10-2	T	M	M	T	M	M	-	-	-
ctp-ch-1	**0.00**	**0.00**	**0.00**	5	5	**4**	10	10	9
ctp-ch-5	0.02	0.01	**0.00**	125	125	**16**	52	52	37
ctp-ch-10	2.2	0.08	**0.02**	4093	4093	**31**	131	127	72
ctp-ch-15	133.24	2.79	**0.07**	131069	131069	**46**	233	227	107
ctp-ch-20	T	M	**0.22**	T	M	**61**	361	352	142
doors-5	0.12	**0.00**	0.01	169	174	**82**	55	18	55
doors-7	3.50	**0.04**	**0.04**	2492	2603	**1295**	123	16	114
doors-9	187.60	**1.07**	**1.07**	50961	53942	**28442**	194	16	182
doors-11	T	M	M	T	M	M	-	-	-
wumpus-5	0.44	**0.14**	0.16	854	441	**233**	706	160	331
wumpus-7	9.28	**1.14**	1.54	7423	1428	**770**	2974	241	992
wumpus-10	1379.62	**7.56**	11.17	362615	4693	**2669**	8122	281	2482
wumpus-15	T	**51.06**	86.16	T	25775	**15628**	21342	304	8241
wumpus-20	T	M	M	T	M	M	-	-	-

Table 1: Comparing compilation time and conditional plan size for CLG and PO-PRP. Also listed are statistics on the size of policy PO-PRP generates. Bold represents the best performance, while T and M represent time limit or memory exceeded.

4 Evaluation

We compared PO-PRP with the state of the art in offline conditional planning, CLG (Albore, Palacios, and Geffner 2009), to assess both the efficiency of the solving process and succinctness of the generated plans. We measure the efficiency in terms of the time it takes to compute a complete solution, and the succinctness in terms of the generated conditional plan. All experiments were conducted on a Linux desktop with a 3.4GHz processor, with time / memory limits of 1hr / 2GB respectively. The times listed here do not include parsing, but this portion of the solving process never exceeded 3 seconds in the problems tested.[3]

We consider four domains that fall under the category of simple domains with partial observability and sensing actions: Coloured Balls (cballs), Canadian Traveller's Problem (ctp-ch), Doors (doors), and Wumpus World (wumpus). The cballs domain involves navigating a known maze to sense for coloured balls and placing them in the correct bin (there is uncertainty in the ball locations and the colour of the balls). The ctp-ch is the variant of the Canadian Traveller's Problem introduced earlier in the paper where the map consists of a chain of locations connected with a pair of roads (one of which is safe to travel on). The doors domain requires the agent to find the unknown position of a door in a long wall before moving on to another wall. Finally, the classic wumpus domain requires the agent to navigate through a maze of wumpus monsters and pits (which can be sensed in a neighbouring location) in order to retrieve a bag of gold. Aside from ctp-ch, we retrieved all problems from the benchmark set that is included with the online contingent planner of BG, *K*-Planner (Bonet and Geffner 2011).[4] All problems, example plans, and PO-PRP source is available online at,

`http://www.haz.ca/research/poprp/`

When measuring the size of the conditional plan, we additionally considered the size of the conditional plan that is produced when we disable the SCD procedure. In such cases, the conditional plan is almost always a tree, as every belief state reachable by the policy is distinct. We use PO-PRP$^-$ to designate the use of PO-PRP with SCD disabled when exporting the conditional plan.

To further assess the succinctness of the partial policy that PO-PRP generates, we include the size of the policy prior to computing the conditional plan. Recall that the policy PO-PRP produces is a mapping of partial belief states to an action – it thus has the potential to be far more compact than a traditional mapping from belief states to actions. Additionally, we list the number of pairs in the policy that were marked as strong cyclic as well as the number of pairs that were used during the construction of the conditional plan.

Table 1 shows the results for a selection of problems from the four domains considered here. The sizes reported for CLG differ from the original published results, as we include the number of branching points (i.e., sensing actions) in our

[3] The conversion to SAS$^+$ rarely created multi-valued variables; so we restricted the invariant synthesis of the parser to 3 seconds.

[4] `https://code.google.com/p/cp2fsc-and-replanner/`

calculation. Invariant actions are suppressed from the conditional plan for both CLG and PO-PRP, and they do not appear in the totals. For the 'Policy Size' column, **all** refers to the full policy size (i.e., the number of condition-action pairs), **strong** refers to the number of pairs marked strong cyclic, and **used** refers to those pairs that were used in generating the conditional plan with PO-PRP. The **used** column is almost identical for PO-PRP^{-}, and is thus omitted.

We found that PO-PRP consistently outperformed CLG in both time to compute a solution and in the size of the final conditional plan. For some domains the size is comparable (e.g., doors and cballs), but the runtime shows a dramatic improvement. This is due in large part to the generalized policy that PRP produces internally, which is used to enumerate the reachable belief states. CLG, in contrast, must continually recompute a weak plan for every new belief state.

The improvement in conditional plan size for PO-PRP over CLG is promising, and it is even more striking when we consider the representation size of the policy. The policy is not smaller than the conditional plan in every case (see for example ctp-ch and wumpus domains), but the policy can be smaller than the conditional plan produced by orders of magnitude. The conditional plan is much more succinct when many of the pairs in the policy are marked as strong cyclic because of the approach we use to compile the conditional plan (cf. Section 3.4). This is most evident in the ctp-ch domain where PO-PRP is able to compute the optimal conditional plan very quickly.

We conclude by noting the potential for further improvement. The SCD procedure of PRP, and subsequently PO-PRP, is merely a sufficient condition. With stronger techniques to mark more of the policy as strong cyclic, we expect the conditional plans produced to be more compact. Additionally, the difference between the **all** and **used** columns gives an indication of the amount of redundant or unnecessary information in the policy. This suggests room for improvement in the final policy that PO-PRP produces.

5 Concluding Remarks

Planning under partially observability is a computationally challenging problem applicable to a number of AI endeavours. We focused on solving a compelling class of PPOS problems where the initial state specification includes a set of state constraints and where uncertainty about the state monotonically decreases. We demonstrated how PPOS problems can be solved offline by exploiting and extending a modern FOND planner. In contrast to the commonly held belief that offline planning for PPOS is impractical, PO-PRP can produce conditional plans several orders of magnitude faster and smaller than the best planners. In our view, offline planners come with significant advantages, such as deadend detection, which is critical in certain settings. Our contribution includes novel techniques for solving PPOS problems and producing a compact plan; it opens the door to a new variety of offline planning techniques.

There are many avenues for future work, including generalizing the framework to effectively handle a larger class of problems, and characterizing problems that can be given succinct policy representations (*e.g.* the ctp-ch domain).

Acknowledgements

We gratefully acknowledge funding from the Ontario Ministry of Innovation and the Natural Sciences and Engineering Research Council of Canada. We also would like to thank the anonymous reviewers for their insightful feedback, and Blai Bonet for making the source of *K*-Planner both publicly available and straightforward to use / extend.

References

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

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1623–1628.

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122(1-2):241–267.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *17th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2001, 473–478.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 1936–1941.

Brafman, R. I., and Shani, G. 2012. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research* 45:565–600.

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

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1):35–84.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan and Claypool Publishers.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173(5-6):503–535.

Hoffmann, J., and Brafman, R. I. 2005. Contingent planning via heuristic forward search witn implicit belief states. In *15th International Conference on Automated Planning and Scheduling (ICAPS)*, 71–80.

Kuter, U.; Nau, D.; Reisner, E.; and Goldman, R. 2007. Conditionalization: Adapting forward-chaining planners to partially observable environments. In *ICAPS 2007—workshop on planning and execution for real-world systems*.

McIlraith, S., and Scherl, R. B. 2000. What sensing tells us: Towards a formal theory of testing for dynamical systems. In

17th National Conference on Artificial Intelligence(AAAI), 483–490.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *22nd International Conference on Automated Planning and Scheduling (ICAPS)*, The 22nd International Conference on Automated Planning and Scheduling.

Muise, C.; McIlraith, S.; and Belle, V. 2014. Non-deterministic planning with conditional effects. In *24th International Conference on Automated Planning and Scheduling*.

Peot, M., and Smith, D. 1992. Conditional nonlinear planning. In *International Conference on AI Planning and Scheduling (AIPS)*, 189–197.

Pinto, J. 1999. Compiling ramification constraints into effect axioms. *Computational Intelligence* 15:280–307.

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

Reiter, R. 2001. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. The MIT Press.

Rintanen, J. 2004. Complexity of planning with partial observability. In *14th International Conference on Automated Planning and Scheduling (ICAPS)*, 345–354.

Strass, H., and Thielscher, M. 2013. A general first-order solution to the ramification problem with cycles. *Journal of Applied Logic* 11(3):289–308.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending graphplan to handle uncertainty and sensing actions. In *15th National Conference on Artificial intelligence (AAAI)*, 897–904.