Planning under Partial Observability by Classical Replanning: Theory and Experiments

Blai Bonet
Universidad Simón Bolívar
Caracas, Venezuela
bonet@ldc.usb.ve

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

Abstract

Planning with partial observability can be formulated as a non-deterministic search problem in belief space. The problem is harder than classical planning as keeping track of beliefs is harder than keeping track of states, and searching for action policies is harder than searching for action sequences. In this work, we develop a framework for partial observability that avoids these limitations and leads to a planner that scales up to larger problems. For this, the class of problems is restricted to those in which 1) the non-unary clauses representing the uncertainty about the initial situation are *invariant*, and 2) variables that are hidden in the initial situation do not appear in the body of conditional effects, which are all assumed to be deterministic. We show that such problems can be translated in linear time into equivalent fully observable non-deterministic planning problems, and that an slight extension of this translation renders the problem solvable by means of classical planners. The whole approach is sound and complete provided that in addition, the state-space is connected. Experiments are also reported.

1 Introduction

Planning problems with partial observability can be formulated as non-deterministic search problems in belief space [Bonet and Geffner, 2000]. This is the mainstream approach in both contingent [Hoffmann and Brafman, 2005; Bryce *et al.*, 2006; Cimatti *et al.*, 2004] and POMDP planning [Kaelbling *et al.*, 1999]. The difficulties for scaling up, in comparison to classical planners, arise from two reasons: keeping track of beliefs is harder than keeping track of states, and searching for action policies or trees, is harder than searching for action sequences. In general, there is no escape from these difficulties, and the challenge is to improve the belief representation and the heuristic or value functions over beliefs. Yet, another approach is to introduce restrictions on the problems to allow for more efficient solution methods.

In this work, we develop a framework for planning under partial observability that avoids the difficulties mentioned above. For this, the range of problems is restricted to those in which R1) the non-unary clauses representing the uncertainty about the initial situation are *invariant* (i.e. respected by all the actions), and R2) the fluents that are hidden in the initial situation do not appear in the body of conditional effects, all of which are assumed to be deterministic.

An special class of problems that fit these restrictions is the class of deterministic planning problems where the uncertainty is about the initial values of a set of multi-valued variables that are used in goals or action preconditions but not in the body of conditional effects. It follows from recent results by Albore *et al.* [2009] that contingent problems P of this form are *simple* in the following sense: they all have *bounded contingent width* (width 1 indeed), which means that they can be efficiently translated into equivalent *fully-observable* non-deterministic planning problems that require no beliefs.

In this work, we go beyond the general results of Albore *et al.*, and introduce a translation that is tailored to contingent problems that comply with R1 and R2 above, that is both more compact and convenient. The translation is more compact because it is *linear* rather than *quadratic*, and it is more convenient because a small modification of it renders the problem solvable by a *classical planner*. The whole approach is shown to be both *sound* and *complete*, provided that in addition to R1 and R2, the state space is connected.

The work is related to a number of proposals for planning with sensing aimed at scalability that do not address the contingent planning problem in its full generality. One thread of such work can be understood as assuming that action preconditions are made of two parts: hidden preconditions and known preconditions. For example, the action of going through a door may have two preconditions: that the agent is at the door and that the door is open. If the agent always knows its location; the first precondition would involve no uncertainty, while the second precondition may be hidden. The assumption in these proposals is that the hidden preconditions become *observable*, and hence known to be either true or false, in states where the non-hidden preconditions hold. Under this assumption, an effective greedy method for *on-line* planning is to: A) make the most convenient assumption about the values of the hidden variables, B) execute the plan that is obtained from the resulting *classical planning* problem, and C) revise the assumptions and replan, if during the execution, the observations refute the assumptions made. An instance of this general idea, used

in robot navigation tasks, goes under the name of 'freespace assumption planning' [Koenig *et al.*, 2003], where the hidden variables refer to the traversable status of cells in a grid. Variations on this basic idea appear in off-line planners that are aimed at solving the fully-observable non-deterministic problem [Ferguson *et al.*, 2005], in certain cases assuming that one value of the hidden (binary) variables is 'preferred' to the other possible value(s) [Likhachev and Stentz, 2009], and in the use of 'safe-assumptions' for simplifying the planning process that can be formally verified at run time [Albore and Bertoli, 2006]. The use of LRTA* for path-finding in partially-observable settings can also be seen from this perspective, as the algorithm requires the action cost $c(a, s)$ only when the agent is at s where the cost value is assumed to be observable. Last, the recent framework of *continual planning* [Brenner and Nebel, 2009] accommodates a language for expressing explicit assumptions of this form in partially-observable settings that involve more than one agent.

In our work, we exploit similar intuitions but adopt a more general formulation that *generalizes and makes precise the idea and scope of 'planning under optimism' in partially observable settings*. In particular, we cast our work in the general setting of domain-independent contingent planning, and establish conditions for a sound and complete approach.

The paper is organized as follows. We start with a general definition of partially observable planning and then consider the properties and methods for the class of problems that comply with the restrictions R1 and R2. We then define a planner based on these results and report experiments that illustrate the scope and effectiveness of the formulation.

2 Partially Observable Planning

We start with the general problem of planning with partial observability and deterministic actions.

2.1 Language

The language is a simple extension of STRIPS with conditional effects, negation, an uncertain initial situation, and sensing. More precisely, a partially observable problem is a tuple $P = \langle F, O, I, G, M \rangle$ where F stands for the fluent symbols, O for the actions, I is a set of *clauses* over F defining the initial situation, G is a set of F-literals defining the goal situation, and M represents the sensor model. An action a has preconditions given by a set of literals, and a set of conditional effects $a : C \rightarrow L$ where C is a set of literals and L is a literal. The sensor model M is a set of pairs (C, L) where C is a set of literals and L is a positive literal. The pair indicates that the truth value of L is *observable* when C is true. Each pair (C, L) can be understood as a sensor that is active when C is true, or as an information-gathering action with precondition C that reveals the boolean value of L. Unlike the other actions, however, these sensing actions trigger when their preconditions hold. We use $\neg L$ to denote the complement of L.

2.2 Policies

The general solution to a partially observable planning problem can be expressed in many forms, the most common of which are trees of possible executions, and functions mapping belief states into actions. Here, for convenience, we mix these two forms and let policies Π denote partial functions from belief states b into *action sequences*, under the restriction that the action sequence $\Pi(b)$ must be applicable in b.

Recall that a state s is a truth valuation over the fluents, a belief state b is a non-empty collection of states, and a formula A holds or is true in b if A is *known* to be true in b, meaning that A holds in every state $s \in b$. Likewise, an action a is applicable in b if the preconditions of a hold in b, and a maps b into the belief b' if a is applicable in b, and b' is the set of states that result from applying the action a to each state s in b. Similarly, an *action sequence* $\pi = a_0, \ldots, a_m$ is applicable in $b = b_0$ and results in the belief $b' = b_{m+1}$ if the action a_i maps b_i into b_{i+1} for $i = 0, \ldots, m$.

Finally, a policy Π *solves* the problem P iff all the executions that are possible according to Π given the initial belief state of P, reach a goal belief. The initial belief is the set of states that satisfy I, and the goal beliefs are the sets of states where G is true. An execution $b_0, b_1, \ldots, b_n$ is possible according to Π if b_0 is the initial belief state of P, b_{i+1} is the result of applying the action sequence $\Pi(b_i)$ to b_i, and b_{i+2} is a possible *successor belief* of b_{i+1}, $i = 0, 2, 4, \ldots$. The belief b' is a *successor* of b, written $b' \in Succ(b)$, if b' represents a maximal set of states in b that agree on the literals that are sensed in b. These literals are those appearing in sensors (C, L) in M such that C is true in b, and neither L nor $\neg L$ are true in b; we call these sensors the *active* sensors in b. As an illustration, if b consists of two states $s = \{p, q\}$ and $s' = \{p, \neg q\}$, then the sensor (C, L) with $C = p$ and $L = q$ is active in b, and its successors are $b_1 = \{s\}$ and $b'_1 = \{s'\}$, as C is true in b, and b_1 and b'_1 are the maximal sets of states in b that agree on the value of the sensed literal q.

3 Simple Partially Observable Planning

The definitions above are general and standard. Next, we focus on the restrictions and transformations that allow us to replace belief states by plain states. Let us recall that an *invariant* in a planning problem is a formula that is true in each possible initial state, and remains true in any state that can be reached from them with the available actions. For example, for an object o that can be picked up and dropped into a number of different positions l_i, $i = 1, \ldots, n$, formulas like $at(o, l_1) \vee \cdots \vee at(o, l_n) \vee hold(o)$, $\neg at(o, l_i) \vee \neg hold(o)$, and $\neg at(o, l_i) \vee \neg at(o, l_k)$, $i \neq k$, are all invariant. The first expresses that a set of literals is *exhaustive*, meaning that at least one of them must be true in every reachable state, while the other two, capture *mutual exclusivity*, meaning that at most one of them can be true in a reachable state. There are several algorithms for detecting such invariants [Fox and Long, 1998], and some of them [Helmert, 2009], focus exactly on the detection of sets of literals that are mutually exhaustive and exclusive, and which can be assumed to represent the different values of a multi-valued variable [Bäckström and Nebel, 1995]. Such invariants are usually denoted as expressions $oneof(x_1, \ldots, x_n)$ where the x_i's are atoms.

Interestingly, it turns out that when the *non-unary clauses* in I are all *invariant* (i.e., those expressing uncertainty) and

the effects of the actions do not depend on fluents that are *hidden* in I (these are the fluents p such that neither p nor $\neg p$ are known to be true in I), then the beliefs that need to be maintained in the planning process can be characterized in a simple form. We say that such problems are *simple:*

Definition 1. *A (deterministic) partially observable problem $P = \langle F, O, I, G, M\rangle$ is* ***simple*** *if the non-unary clauses in I are all* invariant*, and no hidden fluent appears in the body of a conditional effect.*

For example, if I is made up of the expression $oneof(x_1, x_2, x_3)$ and the literal $\neg x_1$, and first, the x_i literals do not appear in the body of a conditional effect, and second, the 'one-of' expression is an invariant, then the problem is *simple.* On the other hand, if I is encoded with the expressions $oneof(x_2, x_3)$ and $\neg x_1$, the problem would not be simple, even if the new formulas are logically equivalent to the old ones. The reason is that the information about the original one-of invariant is lost in the new encoding.

Notice that hidden fluents can appear in action preconditions and heads of conditional effects in simple problems, and hence, there is no requirement for hidden fluents to be invariant or static themselves: they can change. Similarly, the literals that are known to hold in I can change as well. What cannot change are the problem constraints captured by the non-unary clauses in I.

One-of expressions are normally used to encode multi-valued variables, and in the absence of other non-unary clauses in the initial situation, problems featuring such constructs express uncertainty only about the *initial value* of such variables. The simplicity of the belief representation often required for such problems follows from recent results in [Albore *et al.*, 2009], where partially observable problems P are mapped into equivalent non-deterministic planning problems $X(P)$ that are *fully-observable* and hence require no beliefs at all. Albore *et al.* show that the size of the translation $X(P)$ is $O(|P|^{w(P)+1})$, where $w(P)$ in the *contingent width* of P. The contingent width, like the notion of conformant width that it generalizes [Palacios and Geffner, 2009], represents the maximum number of hidden multi-valued variables that interact in the problem such that all of them are relevant to a goal, an action precondition, or an observable literal. Variables interact one with the other through action *conditional effects*, but do not interact at all through action *preconditions*, as in the latter case, the value of the variables must be known with certainty. Thus, our simple problems P have width $w(P) = 1$, and hence, they can be translated into equivalent fully-observable problems $X(P)$ of size $O(|P|^2)$. This thus means, among other things, that the complexity of the belief representation required for such problems is no more than *quadratic* in the size of P, rather than exponential as in the most general case.

By focusing on *simple problems* P, with multi-valued variables or not, we obtain a more efficient translation that is *linear* in P rather than *quadratic*, and which provides the basis for solving P using classical planners. The linear belief representation and update is granted by the following:

Theorem 2 (Simple Beliefs)**.** *Let P be a* simple *partially observable problem, and let I_u stand for the set of non-unary clauses (invariants) in the initial situation I of P. Then if b is a reachable belief state from I that makes a set R of literals true, then b is the set of states that satisfies R and I_u, and hence, it is fully characterized by the formula $R \cup I_u$.*

In other words, in simple problems, *we just need to keep track of the set of literals that are true*; the additional information needed is captured by the invariants that are given in I, which as invariants, do not change and hence do not have to be tracked. A relevant result is the following. We say that a literal L is known in a belief state b if L is known to be true in b or known to be false in b:

Proposition 3 (Monotonicity)**.** *If the literal L is known in a reachable belief state b over a simple problem P, and b' is a belief reachable from b, then L is known in b'.*

These results provide conditions under which (reachable) beliefs can be given a very simple representation with no information loss, in terms of literals that need to be maintained, and invariants that need to be detected in the initial situation.

3.1 Compiling beliefs away: $K'(P)$ Translation

The translation $K'(P)$ below, like the translation $X(P)$, captures partially observable problems P at the 'knowledge-level' [Petrick and Bacchus, 2002] by applying an extension of the K_0 translation for conformant problems [Palacios and Geffner, 2009]. The K_0 translation replaces each literal L by two fluents KL and $\neg KL$ that stand for whether L is known to be true or not, and is complete for problems with 0 conformant width.

Definition 4. *For a partially observable problem $P = \langle F, O, I, G, M\rangle$, $K'(P) = \langle F', O', I', G', M', D'\rangle$ is the* fully observable non-deterministic problem *where*

- $F' = \{KL, K\neg L \,:\, L \in F\}$,
- $I' = \{KL \,:\, L \in I\}$,
- $G' = \{KL \,:\, L \in G\}$,
- $M' = \{KC, \neg KL, \neg K\neg L \rightarrow KL | K\neg L : (C, L) \in M\}$,
- $D' = \{KC \supset KL \,:\, \textit{if } \neg C \vee L \textit{ invariant in } I\}$
- $O' = O$ *but with each precondition L for $a \in O$ replaced by KL, and each conditional effect $C \rightarrow L$ replaced by $KC \rightarrow KL$ and $\neg K\neg C \rightarrow \neg K\neg L$.*

The expressions KC and $\neg K\neg C$ for $C = L_1 \wedge L_2 \ldots$ are abbreviations of the formulas $KL_1 \wedge KL_2 \ldots$ and $\neg K\neg L_1 \wedge \neg K\neg L_2 \ldots$ respectively.

In $K'(P)$, the M' component stands for a set non-deterministic rules $KC, \neg KL, \neg K\neg L \rightarrow KL | K\neg L$, that capture the effects of the sensors (C, L) in P at the knowledge-level. Namely, such rules make L known (either true or false), when C is known to be true, and L is not known. Likewise, the D' component captures the invariants in I with the literals L replaced by KL.

The solution to the *fully-observable non-deterministic* problem $K'(P)$ can be expressed in a way analogous to the solutions to the *partially observable* problem P, but as a policy Π mapping *states* s over $K'(P)$ into action sequences $\Pi(s)$ that must be applicable in the state s. The criterion of applicability is the same as in classical planning, as the state s is fully known, and all the actions in $K'(P)$ have deterministic effects. The non-determinism in $K'(P)$ is the result of

the non-deterministic rules that represent the effect of sensors, and enters into the conditions that make the policy Π a *solution* to the problem $K'(P)$.

A policy Π *solves* $K'(P)$ iff all the executions that are possible according to Π in $K'(P)$ reach a goal state. A goal state is a state where the literals $KL \in G'$ are true. An execution $s_0, \ldots, s_n$ is possible according to Π if $s_0 = I'$ and s_{i+1} is the result of applying the action sequence $\Pi(s_i)$ to s_i, and s_{i+2} is a possible *successor state* of s_{i+1}, for $i = 0, 2, 4, \ldots$.

The set $Succ(s)$ of successor states of s is defined in terms of the non-deterministic rules in M' encoding sensing, and the clauses in D' encoding invariants. Let $r_i = A_i \rightarrow B_i|C_i$ for $i = 1, \ldots, n$, be the set of non-deterministic rules in M' such that A_i is true in s, and let $Succ_0(s)$ be the set of states s' that can be produced from s by applying action sequences $a_1, \ldots, a_n$, where a_i is an action with the single (deterministic) conditional effect $A_i \rightarrow B_i$, or the single conditional effect $A_i \rightarrow C_i$. Then $Succ(s)$ is defined as the set of states in $Succ_0(s)$, each one closed under the formulas in D'; i.e, $s'' \in Succ(s)$ iff $s' \in Succ_0(s)$ and s'' is s' extended with the literals KL entailed by the literals in s' and D'.

Provided with these definitions, the relationship between partially observable problems P and their fully-observable translations $K'(P)$ is captured by the following results. The expression s_b is used to denote the set of literals KL such that L is true in b, and b_s is used to denote the single reachable belief state b that makes a literal L true iff KL is in s (this belief state is unique in simple problems as a result of Thm. 2).

Theorem 5 (Soundness)**.** *Let Π be a policy that solves the fully-observable problem $K'(P)$ for a partially observable problem P. Then if Π maps s into the action sequence π, the function Π' that maps b_s into π is a policy that solves P.*

Theorem 6 (Completeness)**.** *Let Π be a policy that solves the partially observable problem P. Then if Π maps a reachable belief state b into the action sequence π, the function Π' that maps s_b into π is a policy that solves $K'(P)$.*

In other words, the partially observable problem P can be solved by solving the fully-observable translation $K'(P)$, and moreover, this is always possible if P is simple and solvable. We turn next to the problem of solving the translation $K'(P)$ using classical planners.

3.2 Classical $K(P)$ Translation

The action sequence π associated with the response $\Pi(b) = \pi$ for a belief b maps b into a new belief b'. Yet, if b' is not a goal belief, nor a belief where a sensor (C, L) becomes active, then $Succ(b') = \{b'\}$ and an equivalent policy Π' can be obtained by changing $\Pi(b)$ to the concatenation of π and π' where $\Pi(b') = \pi'$. This reduction just removes an 'idle pivot' belief in some executions, leaving the other executions untouched. If we apply this argument iteratively, we see that for completeness it suffices to focus on policies Π that map beliefs reachable from I and Π into goal beliefs or beliefs that activate a sensor (C, L). We call such policies *reduced*.

In the case of simple problems P and in light of the completeness theorem, *reduced policies* Π *over* P, translate into *reduced policies* Π *over* $K'(P)$ where $\Pi(s)$ is an action sequence that maps s into a state s' where the goal G' is true, or KC is true for a sensor (C, L) such that KL and $K\neg L$ are both false in s. This means that the action sequence $\Pi(s)$ can be conceived as a plan for the *classical planning problem* that results from $K'(P) = \langle F', O', I', G', M', D' \rangle$ when the initial situation is s, the new goal is the disjunction of the old goal G' and the various such KC's, and the components M' and D' are ignored. Any sequence $\Pi(s) = \pi$ obtained from a classical planner for that problem will achieve the goal or make a hidden fluent known. Since the number of fluents is finite, and known fluents remain known, any Π constructed in this manner, is guaranteed to solve $K'(P)$ and hence P, if in addition, the space is connected, a condition that ensures that all such classical planning problems have solutions.[1]

The shortcoming of policies derived in this manner is that they may drive the agent to look for information that is not necessary for achieving the goal. In order to compute policies Π that are always goal oriented, we make two refinements in the definition of the *classical planning problem* $K(P)$. First, the *goal* of $K(P)$ is set to the *goal* of $K'(P)$. Second, in order to make the goal achievable from any state even when information from sensors is needed, we translate the *invariants* and *sensors* in P into actions in $K(P)$. A sensor (C, L) is translated into two *deterministic actions*, one with preconditions KC, $\neg KL$ and $\neg K\neg L$, and effect KL, and the other with same preconditions but effect $K\neg L$. We denote these actions as $A(C, L)$ and $A(C, \neg L)$, and call them *assumptions*, as each makes an assumption about the truth value of the literal L whose value is observed when C is true.

Definition 7. *The classical problem $K(P)[s]$ is defined from P and the translation $K'(P) = \langle F', O', I', G', M', D' \rangle$ as $K(P)[s] = \langle F'', O'', I'', G'' \rangle$ where $F'' = F'$, $G'' = G'$, $I'' = s$, and O'' is O'* union *the actions $KC \rightarrow KL$ for the invariants $\neg C \vee L$ in P, and the actions $A(C, L)$ and $A(C, \neg L)$ for the sensors (C, L) in P.*

Since this classical problem is key in the computational account, let us illustrate it through an example. Let the invariant $oneof(y_1, \ldots, y_n, h)$ represent the possible locations y_i of an object, with h standing for the object being held, and let x_1, $\ldots$, x_n denote the possible agent locations. The agent location is known but the object location is not, except for h that is initially false. There are actions that allow the agent to move from one location to the next, and actions to pick up the object from a location if the object is known to be there. From a given location, the agent can sense whether the object is in that location or not, and the goal is to hold the object. The encoding P of this problem is direct with the one-of invariant in I, and the sensors (x_i, y_i) in M, $i = 1, \ldots, n$. The translation $K'(P)$ is direct as well with fluents Kx_i, $K\neg x_i$, Ky_i, and $K\neg y_i$, and pick-up actions with precondition Kx_i and Ky_i, and effects Kh and $K\neg y_i$. The translation $K(P)[s]$ extends $K'(P)$ with actions standing for the invariants, and actions standing for the sensors. The former feature actions with preconditions $K\neg h$ and $K\neg y_i$ for $i \neq k$, and effect Ky_k, while the latter feature actions (assumptions) $A(x_i, y_i)$ and $A(x_i, \neg y_i)$ for $1 \leq i \leq n$, both with preconditions Kx_i, $\neg Ky_i$, and $\neg K\neg y_i$, and effects Ky_i and $K\neg y_i$ respectively.

[1] The space is connected when for every pair of states reachable from a possible initial state, one is reachable from the other.

The classical planning problem $K(P)[s]$ for $s = I'$ is solvable if P is solvable and simple. Yet, the plan π obtained for $K(P)[s]$ cannot always be applied in one shot over $K'(P)$. This is because the plan may include actions that are not in $K'(P)$; i.e., those expressing invariants or assumptions. A suitable non-empty prefix of the plan, however, can always be applied. Let us say that a prefix π' of the action sequence $\Pi(s) = \pi$ *reveals an assumption* in π if π' achieves KC from s for an assumption $A(C, L)$ in π. Then:

Definition 8. *The* executable prefix *of a plan π for the problem $K(P)[s]$ is the shortest prefix of π that reveals an assumption in π, with the invariant actions removed. If π has no assumptions, it is π without the invariant actions.*

Notice that any plan π for the state s that includes an assumption $A(C, L)$ must include a prefix that achieves KC, as KC is a precondition of $A(C, L)$. Thus, some action before the assumption produces an observation (in P) that either *confirms* or *refutes* the assumption. The invariant actions are needed in $K(P)$ only after gathering the first observation, and hence can be dropped from the prefix. The following exploration and exploitation principle [Kearns and Singh, 2002] holds for executable prefixes that either must reach the goal or activate a sensor:

Proposition 9 (Exploration or Exploitation)**.** *Let P be a partially observable problem, b a reachable belief in P, and π a plan for $K(P)[s_b]$. Then, either b' is a goal belief or $|Succ(b')| > 1$ where b' is the result of applying the executable prefix π' of π in b.*

The main result of the paper, about the conditions under which partially observable problems P can be solved using classical planners, can be expressed by means of the following theorems, which require that the *non-unary clauses* in I, that express invariants, to be in *prime implicate form*:[2]

Theorem 10. *Let P be a p.o. problem with a* connected space *and the set of non-unary clauses in* prime implicate form. *Then Π is a (reduced) policy that solves $K'(P)$ **iff** for every state s reachable by Π in $K'(P)$, $\Pi(s) = \pi$ is the **executable prefix** of a classical plan for $K(P)[s]$.*

Theorem 11. *Let P be a* simple *p.o. problem with a* connected space *and the set of non-unary clauses in* prime implicate form. *Then Π is a (reduced) policy that solves P **iff** for every belief state b reachable by Π in P, $\Pi(b) = \pi$ is the **executable prefix** of a classical plan for $K(P)[s_b]$.*

The first result follows from the fact that any executable prefix achieves the goal of $K'(P)$ or makes a hidden literal known. Then, from monotonicity in $K'(P)$ (every know literal remains known) and connectedness that ensures that all relevant classical problems $K(P)[s]$ remain solvable, any policy defined from such plans will eventually reach the goal. The second result is a consequence of the soundness and completeness properties of $K'(P)$ for problems P that are *simple*.

3.3 K-Planner

The K-translation is the basis for an *on-line K-planner* for partially observable problems. Following Theorem 11, the K-planner constructs a policy Π that solves P in on-line fashion, setting the response $\Pi(b)$ for the current belief as the prefix π of the plan obtained with an off-the-shelf classical planner for the classical problem $K(P)[s_b]$. The K-planner is always sound, and it is complete if P is simple and connected. The K-planner solves an instance with a given hidden initial state if the single execution that results from the policy $\Pi(b)$ computed in this fashion reaches the goal. In the K-planner, the beliefs b over P are represented by the states over the translation $K'(P)$.

[2] A set of clauses $\mathcal{C}$ is in prime implicate form if any clause C entailed by $\mathcal{C}$ is a tautology or is subsumed by a clause C' in $\mathcal{C}$.

domain	N	solved	len.	calls	search
freespace	140	134	26.49	8.31	8.02
doors	90	69	132.96	85.17	11.98
wumpus	70	70	34.60	2.39	1.11
kill-wumpus	30	30	49.20	31.10	1.92
colored-balls	75	26	115.81	27.54	26.51
trail	140	133	22.77	8.18	7.54

Table 1: K-planner using FF. Columns stand for domain, number of instances, number of solved instances, and the average over solved instances for the length of the on-line plan, and the number of calls and search time invested by FF. Times are in seconds.

4 Experimental Results

We tested the K-planner on instances from several domains, all of which can be naturally encoded as simple partially observable problems. Several of these domains are beyond the reach of simple replanners as non-trivial beliefs that integrate the observations gathered need to be maintained. In the K-planner, this is achieved by keeping track of literals while maintaining the invariants. The domains are:

- **freespace**: the agent needs to reach a final position by moving through unknown terrain. Each cell of the grid is either blocked or clear. As the agent moves, it senses the status of adjacent cells.
- **doors**: the agent moves in a grid to reach a final position, yet is has to cross doors whose positions are only partially known.
- **wumpus**: the agent moves in a grid to reach a final position that contains a number of deadly *wumpuses* that must be avoided. As the agent moves, it smells the stench of nearby wumpuses.
- **kill-wumpus**: a variation of the above in which the agent must locate and kill the *single* wumpus in the grid.
- **colored-balls**: the agent navigates a grid to pick up and deliver balls of different colors to destinations that depend on the color of the ball. The positions and colors of the balls are unknown, but when the agent is at a position, he observes if there are balls in there, and if so, their colors.
- **trail**: the agent must follow a trail of stones til the end. The shape and length of the trail are not known. The agent cannot get off trail but can observe the presence of stones in the nearby cells.

Experiments on a total of 545 instances were performed using FF [Hoffmann and Nebel, 2001], on Linux machines with Xeon X5355 CPUs and limits of 1,800 seconds and 2Gb of RAM. Table 1 summarizes the results. For each domain, the table shows the total number of solved instances and the average length of the on-line plan, and the average number of calls and search time over the solved instances.[3] We do

[3] Search time is not the same as total time as the current implementation solves every problem $K(P)[s]$ that arises in an execution

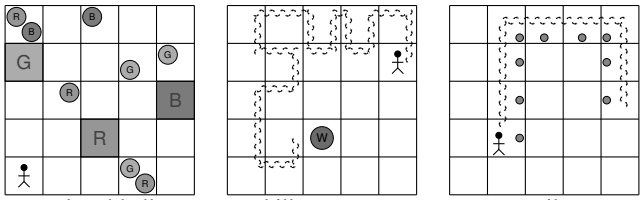

Figure 1: Some instances of colored-balls, kill-wumpus and trail with the initial state shown. The agent does not know, in colored-balls, the position and colors of the balls, in kill-wumpus, the position of the wumpus, and in trail, the positions of the stones. The plan found by K-planner for kill-wumpus and trail is shown.

not include comparisons with contingent planners because the size of the contingent plans for these domains is just too large. For instance, a 9×9 instance of colored balls solved by the K-planner features 18 balls of 4 possible colors, resulting in more than $(81 \cdot 4)^{18} = 1.54 \times 10^{45}$ possible initial states.

Some of the domains are well beyond the ability of simple replanners. For example, in kill-wumpus the agent must perform complex inferences to locate the wumpus. The invariants are 1) $\neg stench(p) \supset \neg wumpus(p')$ for $adj(p, p')$, 2) $\neg wumpus(p) \vee \neg wumpus(p')$ for $p \neq p'$, and 3) $stench(p) \supset \vee_{i=1}^{4} wumpus(p_i)$ where $\{p_i\}_{i=1}^{4}$ are the cells adjacent to p. Fig. 1 shows a 5×5 instance with the path traversed by the agent. Observe that when the agent reaches $(2, 2)$, it infers that the wumpus is at $(3, 2)$. This is obtained because at $(2, 2)$ there is a stench which, by invariant 3, implies that there is a wumpus at either $(1, 2)$, $(2, 3)$, $(2, 1)$ or $(3, 2)$. The inference follows then because the agent visited all such cells except $(3, 2)$.

5 Summary

We have laid out a logical and computational framework for planning with partial observability that exploits the use of classical planners. The formulation generalizes and makes precise the idea and the scope of 'planning under optimism' in partially observable settings. In particular, building on the general setting of domain-independent contingent planning, we established conditions under which the approach is sound and complete. A number of experiments have also been reported that illustrate the efficiency and expressive power of the proposed framework. The basic framework can be extended in a number of ways, including the use of actions with non-deterministic but observable effects. A limitation of the framework is that the idea of 'planning under optimism' is not always good; in particular, when backing up is impossible or not feasible. In the future, we would like to explore extensions that take such considerations into account. The planner and the examples will be made available through the web.

from scratch. This means that the PDDL of every such problem is parsed and preprocessed in each call. This is a considerable overhead that explains why FF fails on some of the instances, as 1,800 seconds is the bound on *total time*. Most of this overhead can be eliminated by implementing the K-planner *inside* the planner. This would make the K-planner more efficient but planner dependent.

Acknowledgmens

H. Geffner is partially supported by grants TIN2009-10232, MICINN, Spain, and EC-7PM-SpaceBook.

References

[Albore and Bertoli, 2006] A. Albore and P. Bertoli. Safe LTL assumption-based planning. In *Proc. ICAPS*, pages 192–202, 2006.

[Albore *et al.*, 2009] A. Albore, H. Palacios, and H. Geffner. A translation-based approach to contingent planning. In *Proc. IJCAI*, pages 1623–1628, 2009.

[Bäckström and Nebel, 1995] C. Bäckström and B. Nebel. Complexity results for SAS+ planning. *Comp. Intell.*, 11(4):625–655, 1995.

[Bonet and Geffner, 2000] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. ICAPS*, pages 52–61, 2000.

[Brenner and Nebel, 2009] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. *Aut. Agents and Multi. Sys.*, 19(3):297–331, 2009.

[Bryce *et al.*, 2006] D. Bryce, S. Kambhampati, and D. E. Smith. Planning graph heuristics for belief space search. *JAIR*, 26:35–99, 2006.

[Cimatti *et al.*, 2004] A. Cimatti, M. Roveri, and P. Bertoli. Conformant planning via symbolic model checking and heuristic search. *Art. Int.*, 159:127–206, 2004.

[Ferguson *et al.*, 2005] D. Ferguson, A. Stentz, and S. Thrun. PAO* for planning with hidden state. In *Proc. ICRA*, pages 2840–2847, 2005.

[Fox and Long, 1998] M. Fox and D. Long. The automatic inference of state invariants in TIM. *JAIR*, 9:367–421, 1998.

[Helmert, 2009] M. Helmert. Concise finite-domain representations for pddl planning tasks. *Art. Int.*, 173(5-6):503–535, 2009.

[Hoffmann and Brafman, 2005] J. Hoffmann and R. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS*, pages 71–80, 2005.

[Hoffmann and Nebel, 2001] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[Kaelbling *et al.*, 1999] L. P. Kaelbling, M. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Art. Int.*, 101:99–134, 1999.

[Kearns and Singh, 2002] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, 2002.

[Koenig *et al.*, 2003] S. Koenig, C. Tovey, and Y. Smirnov. Performance bounds for planning in unknown terrain. *Art. Int.*, 147(1-2):253–279, 2003.

[Likhachev and Stentz, 2009] M. Likhachev and A. Stentz. Probabilistic planning with clear preferences on missing information. *Art. Int.*, 173(5–6):696–721, 2009.

[Palacios and Geffner, 2009] H. Palacios and H. Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *JAIR*, 35:623–675, 2009.

[Petrick and Bacchus, 2002] R. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. ICAPS*, pages 212–222, 2002.