Completeness of Online Planners for Partially Observable Deterministic Tasks

Blai Bonet and **Gabriel Formica** and **Melecio Ponte**
Departamento de Computación
Universidad Simón Bolívar, Caracas, Venezuela
bonet@ldc.usb.ve, {gformica, mponte}@ac.labf.usb.ve

Abstract

Partially observable planning is one of the most general and useful models for dealing with complex problems. In recent years there have been significant progress on the development of planners for *deterministic models* that offer strong theoretical guarantees over certain subclasses of tasks. These guarantees however are difficult to establish as they often involve reasoning about features that are specific to the planner and subclass of tasks. In this paper we develop a formal framework for reasoning about online planning over deterministic tasks, identify a set of general conditions that are sufficient to guarantee completeness, and obtain novel and simple planners that are complete over non-trivial and interesting classes of tasks. Building on top state-of-the-art online planners, we implement some of our ideas and make a comparison with a state-of-the-art online planner.

Introduction

Partially observable planning is one of the most general and useful models for dealing with complex problems (Ghallab, Nau, and Traverso 2004; Geffner and Bonet 2013). In recent years there have been significant progress in the development of planners for such tasks (Bonet and Geffner 2014b; Maliah et al. 2014; Brafman and Shani 2012).

Planners can be classified as offline or online systems. In the offline approach, the planner computes a full contingent solution that solves the task by considering all the possible initial states, all the possible state transitions, and all the sensing signals that the agent may perceive. This approach seems appealing as a system confronted with such a task needs to compute a full contingent solution only once. However, the computational burden for computing (and storing) a complete solution are prohibitive in practice except for very small or simple problems (Rintanen 2004; Haslum and Jonsson 1999). In the online approach, the agent is interested in solving the task for the current, but unknown, initial state and thus only cares about the execution that unfolds as it interacts with the environment. The online approach is thus computationally feasible for a significantly broader class of tasks as the planner only needs to decide at each decision time what is the next action to apply given the observed execution.

A main target of recent research is the class of deterministic tasks in which transitions and sensing are both deterministic. Many of the current state-of-the-art solvers offer strong guarantees over restricted subclasses of deterministic tasks (Bonet and Geffner 2014b; Brafman and Shani 2012; Bonet and Geffner 2011), but the proofs and techniques used to establish such guarantees are often complex and specific to the combination of planner and subclass of problems.

In this paper we develop a general and formal framework for understanding and reasoning about online planning on partially observable *deterministic* tasks, define general and relevant properties for planners, and obtain a set of sufficient conditions that guarantee soundness and completeness. In doing so, we identify a class of problems for which a planner implementing a trivial action selection mechanism is guaranteed to be sound and complete; something that at first seems surprising as there is the common belief that the action selection problem is intractable even on relatively simple tasks.

The paper is organized as follows. In the next section we characterize the class of problems considered and recall basic notions about planning with partial information. Then, we give a brief but general account of offline and online planners, and formalize the notion of protocol, that norms the execution of online planners, and the notions of soundness and completeness. Online planners are then characterized over the dimensions of belief tracking and action selection, and formal results are presented. We finish by presenting three simple complete planners that extend the LW1 planner (Bonet and Geffner 2014b), some experimental results, and a brief summary.

Model and Executions

Problems and planners are characterized using state models, even though problems are presented to planners in the form of compact specifications over suitable languages.

A Partially observable deterministic (POD) planning task is a tuple $P = (S, A, S_{init}, S_G, f, O, \Omega)$ where:

1. S is a finite state space,
2. A is a finite set of actions, where $A(s) \subseteq A$ is the set of actions that are applicable at state $s \in S$,
3. $S_{init} \subseteq S$ is the set of possible initial states,
4. $S_G \subseteq S$ is the set of goal states,

5. $f : S \times A \to S$ is a *deterministic* transition function where $f(s, a)$ is the unique state that results after applying the action $a \in A(s)$ in the state $s \in S$,
6. O is a finite set of observation (tokens), and
7. $\Omega : S \times A \to O$ is a *deterministic* sensing model where $\Omega(s, a)$ is the token observed by the agent upon reaching the state s after applying the action a.

For simplicity we do not consider action costs but these can be easily fixed in the model.

The model is partially observable because the agent receives information about the system only through the observations. Under deterministic sensing, there is always a *unique* possible observation after executing an action, but since the underlying system state is unknown to the agent, the observation does not necessarily reveal the hidden state as other states may be also consistent with it. The agent thus needs to keep track of the possible states by maintaining a belief state. Belief states are sets of states associated with the *observable executions* (Geffner and Bonet 2013). Formally, an (observable) execution is a finite interleaved sequence $\langle a_0, o_0, a_1, o_1, \ldots \rangle$ of actions and observations. The belief state b_τ associated with the execution τ is inductively defined as

- $b_\tau = S_{init}$ for the empty execution $\tau = \langle\rangle$,
- $b_{\langle \tau, a \rangle} = \{f(s, a) : s \in b_\tau\}$, and
- $b_{\langle \tau, a, o \rangle} = \{s \in b_{\langle \tau, a \rangle} : \Omega(s, a) = o\}$.

Let us consider two executions τ and τ'. The execution that consists of τ followed by τ' is denoted by $\langle \tau, \tau' \rangle$. In such a case, we say that τ is a prefix of $\langle \tau, \tau' \rangle$ and that $\langle \tau, \tau' \rangle$ is an extension of τ (or that $\langle \tau, \tau' \rangle$ extends τ). We also write $\tau' \leq \tau$ when τ' is a prefix of τ, and $\tau' < \tau$ when $\tau' \leq \tau$ and $\tau' \neq \tau$.

An execution τ is admissible if: (1) $b_\tau \neq \emptyset$ and (2) for each prefix $\langle \tau', a \rangle$ of τ, the action a is applicable at $b_{\tau'}$. An action a is applicable at belief b iff $a \in A(s)$ for each state $s \in b$. All executions τ considered in this paper are implicitly assumed to be admissible.

An execution $\tau = \langle a_0, o_0, a_1, o_1, \ldots \rangle$ is possible for state s (or, briefly, that τ is for s) iff there is a state trajectory $\langle s_0, a_0, s_1, a_1 \ldots \rangle$ with $s_0 = s$ and that for $i \geq 0$: (1) the action a_i is applicable at $b_{\tau_{0:i}}$ where $\tau_{0:i}$ is the prefix of τ containing i pairs (a, o), (2) $s_{i+1} = f(s_i, a_i)$, and (3) $o_i = \Omega(s_{i+1}, a_i)$. By determinism such state trajectory is unique if it exists. We refer to it as the state trajectory induced by τ and seeded at s. We say that an execution τ is goal-reaching iff $b_\tau \subseteq S_G$. We denote by T_{init} the set made of the empty execution and all the executions that end in an observation and are for some initial state $s \in S_{init}$.

Offline vs. Online Planners

An offline planner is an algorithm that given task P, outputs a *valid plan* for P. A plan for P is a *partial nondeterministic function* $\pi : T_{init} \to 2^A \setminus \emptyset$ such that π is *downward closed* and $\pi(\tau) \subseteq A(s)$ for each $s \in b_\tau$ and $\tau \in Dom(\pi)$. Plan π is downward closed iff for each execution $\tau = \langle a_0, o_0, \ldots, a_{n-1}, o_{n-1} \rangle$, if π is defined on τ, then π is also defined on $\tau_{0:i}$ and $a_i \in \pi(\tau_{0:i})$ for $i = 0, 1, \ldots, n$. The plan π is valid iff it guarantees to reach a goal from any initial state (i.e., for each initial state $s \in S_{init}$ and each execution τ for s on which π is defined, there is an extension $\tau' = \langle \tau, \tau'' \rangle$ for s on which π is defined and $b_{\tau'} \subseteq S_G$).

An offline planner computes a complete plan for a given task whose size is often exponential in the number of states (Rintanen 2004). In many cases, however, the agent is typically interested in solving the task for a given initial but hidden state, and thus only interested in generating a single goal-reaching execution. Planners that guide the agent to generate such executions are called online planners.

Formally, an online planner π is an algorithm that on input $\langle P, \tau \rangle$, where P is a task and $\tau \in T_{init}$ is an execution for some initial state, outputs a non-empty subset $\pi(\tau) = \pi(P, \tau)$ of actions applicable at b_τ. An online planner is thus just an algorithm that generates applicable actions given an observable execution.

We formalize the utilization of the planner by the agent with the notion of an *online protocol*. An online protocol is a pair $L = (P, s)$ made of the task P and a hidden initial state $s \in S_{init}$. The protocol interacts with the online planner π as follows:[1]

1. Let $\lambda = \langle s \rangle$ be the empty state trajectory seeded at s
2. Let $\tau = \langle\rangle$ be the empty execution
3. While $b^\pi_\tau \not\subseteq S_G$ (i.e. b^π_τ is not a goal belief) do
4. Run π on input $\langle P, \tau \rangle$ to obtain set of actions $\pi(\tau)$
5. If $\pi(\tau)$ is empty, return FAILURE
6. Non-deterministically choose action $a \in \pi(\tau)$
7. Let state $s' := f(Last(\lambda), a)$ and token $o := \Omega(s', a)$
8. Update $\lambda := \langle \lambda, a, s' \rangle$ and $\tau := \langle \tau, a, o \rangle$.

The condition in the while loop refers to the belief b^π_τ that is computed by π in order to *approximate* the true belief b_τ. The protocol *fails* when $\pi(\tau)$ is empty, meaning the planner is not able to produce an applicable action, or *succeeds* when the agent (i.e. planner) is sure of reaching the goal. If b^π_τ is replaced by b_τ in the stopping condition, the protocol succeeds once a goal belief is reached, even if the agent is not aware of it, an odd situation when the planner models a fully autonomous agent.

Executions τ such that $b^\pi_\tau \subseteq S_G$ are called strong goal-reaching executions for π. The protocol is thus designed to succeed once a strong goal-reaching execution is generated.

A planner π is sound iff it generates admissible executions, and it is complete for a class $\mathcal{P}$ of tasks iff for every task $P \in \mathcal{P}$, the above process succeeds for every protocol $L = \langle P, s \rangle$ where $s \in S_{init}$.

Components of an Online Planner

We characterize general online planning in terms of two components, belief tracking and action selection.

[1] A protocol $L = (P, s)$ is a formalization of the *process* of executing the online planner π on the task P assuming that the hidden initial state is s. Other formalizations are possible. For example, a formalization in which the protocol is a stateless black box that receives the task P, the planner π, the ongoing state-trajectory λ, and the ongoing execution τ, and that outputs updated state-trajectory and execution after one action in $\pi(\tau)$ is selected and applied.

Belief Tracking

As a sound planner π must output actions that are applicable at each possible current state, the planner must calculate an approximation b^π_τ for b_τ for identifying applicable actions. We do not ask planners to compute b_τ exactly since this is computationally intractable (Bonet and Geffner 2014a). An approximation of belief b_τ is thus a belief b that contains b_τ because if an action a is applicable at b, then a is also applicable at b_τ. As the approximation becomes coarser, the set of candidate actions becomes smaller as each candidate action must be applicable at each state in the approximation. In the extreme case, a poor approximation may leave the planner with no action to choose from.

In our framework, the approximation b^π_τ is handled as an explicit subset of states. Planners however may represent approximations in different forms (Brafman and Shani 2016; Bonet and Geffner 2014b; Maliah et al. 2014; Brafman and Shani 2012; Albore, Palacios, and Geffner 2009).

Action Selection

Even exact knowledge of the set of applicable actions is not sufficient to guarantee completeness. In some tasks, an early bad choice of actions may cause the problem to become unsolvable. But the picture is worse because even in fully observable problems and complete connected state spaces, in which there are no dead ends or uncertainty, a bad choice of actions may cause the planner to loop on state space preventing it for generating a goal-reaching execution. For example, in a problem where the agent is initially at the middle of a $1 \times n$ grid and the goal is to reach one of the extremes, either position 1 or n, the agent may loop if it interleaves the actions to go left and to go right infinitely often.

We thus extend the scope of the framework by considering action selection mechanisms that return sequences of actions instead of single actions. Formally, we consider an action selection mechanism that receives the execution τ and the approximation $b = b^\pi_\tau$ computed by belief tracking, and generates a set $\pi(\tau, b)$ of action sequences with the constraint that the first action in each sequence is applicable at b. This general form of action selection is motivated by the way some planners work. Two remarks:

- If π is a planner that computes action sequences, we obtain a planner that returns applicable actions by defining $\pi(\tau) = \{First(\sigma) : \sigma \in \pi(\tau, b)\}$ where $b = b^\pi_\tau$.
- Planners that generate sequences usually select one sequence σ and "stick with σ" until exhausting σ or reaching an action in σ that is not applicable at the current belief approximation. Such planners can be fit into the model: the first time that an action is requested for execution τ, the planner generates and picks one sequence σ and outputs the first action a in σ. The protocol then generates the observation o associated to the action a and the planner is called again but now on the execution $\tau' = \langle \tau, a, o \rangle$. The planner then retrieves the last computed sequence σ, which was stored in a cache, and checks whether the next action a' in σ is applicable at $b^\pi_{\tau'}$. In the affirmative, the next action to execute is a'. Else, the planner discards the cached sequence and computes a new sequence from scratch using the new belief approximation $b^\pi_{\tau'}$. A planner that operates in this manner is called a *committed planner*.

Online planners that compute sequences usually implement action selection with *classical planners* (Bonet and Geffner 2014b; Maliah et al. 2014; Brafman and Shani 2012; Bonet and Geffner 2011; Albore, Palacios, and Geffner 2009; Palacios and Geffner 2009). Such online planner π, when called upon an execution τ, works by creating a *classical (planning) problem* that captures the knowledge in b^π_τ and the knowledge dynamics of the task (using suitable propositional encodings (Petrick and Bacchus 2002)), and that implements a limited form of inference over the encoding. A classical planner then computes a plan for the classical problem whose first action, by design, is guaranteed to be applicable at the approximation b^π_τ.

Formal Results

Thus far we have formalized planners and protocols, notions of soundness/completeness, and characterized online planners by the belief tracking and action selection components.

The completeness of a planner π depends on a delicate balance between the inferential and planning power in π. The inferential power is related to how well the belief tracking component approximates the beliefs b_τ for the observed executions τ, while the planning power is related to the quality of the action sequences σ generated by π.

We begin by defining properties about inference. An execution τ that is possible for some initial state and that ends in an action a is a *branch point* if there are two different observations o and o' such that the extensions $\langle \tau, o \rangle$ and $\langle \tau, o' \rangle$ are both possible; in such case, we say that o and o' are possible after τ. In deterministic tasks, if τ is an execution for initial state s then $|b_\tau| \leq |S_{init}|$, and the inequality is strict when τ contains a branch point; i.e. when τ contains a prefix that is a branch point. These facts are direct consequences of the determinism in transitions and sensing (Bonet 2009). Indeed, if $\{o_1, \ldots, o_m\} \subseteq O$ is the set of observations that are possible after a branch point τ, then $|b_\tau| = \sum_{1 \leq i \leq m} |b_{\langle \tau, o_i \rangle}|$ since $b_{\langle \tau, o_i \rangle}$ and $b_{\langle \tau, o_j \rangle}$ are disjoint for $i \neq j$. These remarks motivate the following:

Definition 1 (Exact Inference)**.** *Planner π implements exact inference iff $b^\pi_\tau = b_\tau$ for each execution τ for an initial state.*

Definition 2 (Monotone Inference)**.** *Planner π has monotone inference iff $|b^\pi_\tau| \leq |b^\pi_{\tau'}|$ for each initial state $s \in S_{init}$, each execution τ for s, and each prefix τ' of τ.*

Definition 3 (Asserting Inference)**.** *Let τ be an execution for an initial state s and τ' be a proper prefix of τ. Planner π has asserting inference for the pair (τ, τ') iff $|b^\pi_\tau| < |b^\pi_{\tau'}|$.*

Monotone inference says that the approximations generated by π along a fixed execution have non-increasing cardinalities, while asserting inference for (τ, τ') says that the inference implemented by π is *powerful enough* to rule out (in hindsight) at least one state from $b^\pi_{\tau'}$. The following result is direct from above definitions:

Lemma 4. *Let π be an online planner. If π implements exact inference, then π has monotone inference and asserting*

inference for all pairs (τ, τ') for which $\tau' < \tau$ and there is a branch point τ'' such that $\tau' \leq \tau'' \leq \tau$.

We now consider the planning component. A plan for state s is a sequence $\sigma = \langle a_0, \ldots, a_{n-1} \rangle$ of actions that induce a goal-reaching state trajectory seeded at s; i.e. a state trajectory $\langle s_0, a_0, \ldots, s_n \rangle$ with $s_0 = s$ and $s_n \in S_G$. A sequence σ is a *weak plan* for belief b iff it is a plan for at least one state s in b (Cimatti et al. 2003).

Definition 5 (Weak Planner)**.** *Planner π is a* weak planner *iff for each belief approximation $b = b^\pi_\tau$: (1) every sequence σ in $\pi(\tau, b)$ is a weak plan for b, and (2) $\pi(\tau, b)$ is non-empty whenever there is a weak plan for b.*

Definition 6 (Covering Planner)**.** *Planner π is covering iff $\pi(\tau)$ contains all the actions that are applicable at b_τ, where τ is any execution for an initial state s.*

We also need a property to link the inference and planning components. Let $\tau = \langle a_0, o_0, \ldots, a_n, o_n \rangle$ be an execution for initial state s, and let $\langle s_0, a_0, \ldots, s_{n+1} \rangle$ be the unique state trajectory seeded at s that is induced by τ. For a finite action sequence $\sigma = \langle a_{n+1}, a_{n+2}, \ldots \rangle$ such that a_{n+1} is applicable at b_τ, let us define the execution

$$\tau[\sigma] = \langle \tau, \ldots, a_{n+k}, o_{n+k} \rangle = \langle a_0, o_0, \ldots, a_{n+k}, o_{n+k} \rangle,$$

with induced state trajectory $\langle s_0, a_0, \ldots, s_{n+k+1} \rangle$, where k is the maximum integer, $1 \leq k \leq |\sigma|$, such that:

- a_i is applicable at b_{τ_i} where $\tau_i = \tau[\sigma]_{0:i}$ is the prefix of $\tau[\sigma]$ of length $2i$, for $i = 0, 1, \ldots, n+k$,
- $s_{i+1} = f(s_i, a_i)$ for $i = 0, 1, \ldots, n+k$, and
- $o_i = \Omega(s_{i+1}, a_i)$ for $i = 0, 1, \ldots, n+k$.

The idea is to extend the execution τusing the actions in σ as much as possible. The fact that τ can be extended with at least on action, i.e. $k \geq 1$, is direct by the assumption that a_{n+1} is applicable at b_τ. Clearly, (1) $\tau[\sigma]$ extends τ, (2) $\tau[\sigma]$ is possible for the initial state s, and (3) the state trajectory induced by $\tau[\sigma]$ extends the state trajectory induced by τ. The length of $\tau[\sigma]$ satisfies $|\tau[\sigma]| \leq |\tau| + 2|\sigma|$ with equality only when all actions in σ are used in $\tau[\sigma]$.

Definition 7 (Effective Planner)**.** *Planner π is effective iff for each execution τ for an initial state s, and each action sequence $\sigma \in \pi(\tau, b)$ where $b = b^\pi_\tau$, either $b^\pi_{\tau[\sigma]} \subseteq S_G$ or π has asserting inference for the pair $(\tau[\sigma], \tau)$.*

Intuitively, π is effective iff it either produces a sequence $\sigma \in \pi(\tau, b^\pi_\tau)$ that generates a strong goal-reaching execution, making the protocol to succeed, or the belief tracking component is able to discard at least one state that was previously deemed as possible after τ.

Finally, we define the target class of planning tasks as those that are connected, and then give sufficient conditions for completeness of online planners:

Definition 8 (Solvable Tasks)**.** *A planning task P is solvable (or goal connected) iff there is a plan for every state s in P.*

Theorem 9 (Main)**.** *Let P be a solvable task and let π be a sound and committed online planner for P. If π is a weak and effective planner and π has monotone inference, then π is complete for P.*

Proof. Suppose that π is not complete. Then there is a protocol $L = \langle P, s \rangle$ such that π either fails or loops. Failure is not possible since P is solvable and thus there is always a plan for every state $s \in b^\pi_\tau$; i.e. $\pi(\tau) \neq \emptyset$ since π is a weak planner. Therefore, π loops and generates an infinite execution $\tau = \langle a_0, o_0, a_1, o_1, \ldots \rangle$ for state s.

Since π is a committed planner, the actions in τ come from sequences σ computed by π. A new sequence σ is computed at the beginning and each time the previous computed sequence is exhausted or terminated early. We say that an index $i \geq 0$ is a *generation index* in τ if a new sequence σ for $b^\pi_{\tau_i}$ is computed, where τ_i denotes the prefix of τ of length $2i$.

We show below that for each generation index $i \geq 0$, either $b^\pi_{\tau_i}$ is a singleton or there is index $j > i$ such that $|b^\pi_{\tau_j}| < |b^\pi_{\tau_i}|$. Let us assume for the moment that this is true. Using π's monotonicity, there must be a generation index $i^* \geq 0$ for which the approximation $b = b^\pi_{\tau^*}$, with $\tau^* = \tau_{i^*}$, is a singleton $\{s^*\}$. At this point, since π is a weak planner and P is solvable, $\pi(\tau^*, b) \neq \emptyset$ and each $\sigma \in \pi(\tau^*, b)$ is a plan for s^*. Since π is a committed planner, the chosen σ is executed until exhaustion and the planner generates a strong goal-reaching execution contradicting the supposition.

We conclude the proof by showing that for each generation index $i \geq 0$, $b^\pi_{\tau_i}$ is a singleton or there is $j > i$ such that $|b^\pi_{\tau_j}| < |b^\pi_{\tau_i}|$. Suppose that $b = b^\pi_{\tau_i}$ is not a singleton and let $\sigma \in \pi(\tau_i, b)$. By definition of committed planner, there is index $j > i$ such that $\tau_j = \tau_i[\sigma]$ (i.e. a non-empty prefix of σ is executed). Since π is effective and τ is infinite, $\tau_i[\sigma]$ cannot be a strong goal-reaching execution. Therefore, π has asserting inference for the pair $(\tau_i[\sigma], \tau_i)$; i.e. $|b^\pi_{\tau_j}| < |b^\pi_{\tau_i}|$. □

Randomized Protocols

We now see an interesting result. If π implements exact inference and $\pi(\tau)$ contains all actions that are applicable at b_τ (i.e. π is a covering planner), then π is complete over a non-trivial set of tasks when the protocol is *randomized*. A protocol L is randomized when it makes a *randomized selection* of the action a in $\pi(\tau)$ (line 6 of the protocol); i.e., each action a in $\pi(\tau)$ has non-zero probability of being chosen by L. The tasks to consider are those that are strongly-solvable:

Definition 10. *A planning task P is strongly solvable iff for each execution τ for an initial state s, there is an extension $\tau' = \langle \tau, \tau'' \rangle$ for s that reaches the goal (i.e. $b_{\tau'} \subseteq S_G$).*

The solvable and strongly-solvable conditions are related but not equivalent. Solvability is about goal-connectness in state space while strong-solvability is about goal-connectness in *belief space*. There are tasks that are solvable but not strongly solvable, and vice versa. Since a random walk visits all states in a fully connected graph with probability 1, a random walk generates a goal-reaching execution with probability 1 on a strongly solvable task. Such a walk is obtained when a randomized protocol is combined with a covering planner:

Theorem 11. *Let P be a strongly solvable planning task and let π be a sound online planner for P. If π is a covering planner, then π generates a goal-reaching (as opposed to strong goal-reaching) execution with probability 1.*

This result does not say that a covering planner π is guaranteed to be complete for randomized protocols. The issue is that generating a goal-reaching execution is not enough for succeeding at the protocol. Additionally, just like a random walk may take a long time to reach a certain state in a connected graph (Koenig and Simmons 1996), generating a goal-reaching execution by above process may require a very large number of steps. We finish this section by noticing that the standard (and simple) notion of fairness over executions cannot be naively used to replace a randomized protocol with a "fair protocol" while preserving Theorem 11.

Complete Planners

The formal properties can be used to define planners that are complete for certain classes of problems. We consider three such classes and define one complete planner for each class.

The first class consists of tasks that are solvable and that have observable goals:

Definition 12. *A task P has observable goals iff for each execution τ for some initial state s, b_τ contains a goal state iff b_τ is a goal belief; i.e. $b_\tau \cap S_G \neq \emptyset$ iff $b_\tau \subseteq S_G$.*

This definition captures the intuition. If $o^* \in O$ is an observation with $\Omega(s,a) = o^*$ for all actions a iff $s \in S_G$, then upon reaching a goal state s and observing o^*, the other states in b_τ must be goal states as well. Conversely, for an observation $o \neq o^*$ and execution τ ending in o, every state s in b_τ cannot be a goal state.

Consider a belief tracking mechanism that is exact over a class $\mathcal{P}$ of solvable tasks with observable goals. Let π_{single} be a *committed planner* that implements exact belief tracking. When in need of an action for the belief b_τ associated with execution τ, π_{single} selects a state $s \in b_\tau$ and returns a plan σ for the classical problem $P[\tau, s]$, where $P[\tau, s]$ is constructed to satisfy the following conditions:

- There is a plan for $P[\tau, s]$ (as the task is solvable),
- Any plan for $P[\tau, s]$ is a plan for s,
- For any plan $\sigma = \langle a_0, \ldots, a_n \rangle$ for $P[\tau, s]$, there is execution $\tau' = \langle a_0, o_0, \ldots, a_n, o_n \rangle$ and state trajectory $\langle s_0, a_0, \ldots, a_n, s_{n+1} \rangle$ seeded at s and induced by τ' such that, for $i = 0, 1, \ldots, n$: (1) $a_i \in A(s_i)$, $s_{i+1} = f(s_i, a_i)$ and $o_i = \Omega(s_{i+1}, a_i)$, and (2) a_i is applicable at $b_{\langle \tau, \tau'_i \rangle}$ where τ'_i is the prefix of τ' of length $2i$.

In words, the classical task $P[\tau, s]$ must be solvable for each execution τ and state $s \in b_\tau$, and each plan σ for $P[\tau, s]$ must be consistent with an execution τ' for s. There are different ways to construct $P[\tau, s]$, and some of them correspond to simple modifications of existing planners.

Let us show that π_{single} is indeed complete. By assumption, the belief tracking implemented by π_{single} is exact and thus, by Lemma 4, π has monotone and asserting inference for the pair (τ, τ') whenever there is a branch point τ'' with $\tau' \leq \tau'' \leq \tau$. On the other hand, π_{single} is a committed and weak planner. Therefore, by Theorem 9, π_{single} is complete if it is an effective planner:

Lemma 13. *Let P be a planning task with observable goals. The planner π_{single} is effective over P.*

Proof. Let $\tau = \langle a_0, o_0, \ldots, a_n, o_n \rangle$ be an execution for an initial state s, and let $\sigma = \langle a_{n+1}, a_{n+2}, \ldots \rangle$ be a plan for $P[\tau, s']$ where $s' \in b_\tau$. We need to show that if $b_{\tau[\sigma]} \not\subseteq S_G$ then π has asserting inference for the pair $(\tau[\sigma], \tau)$.

If $|\tau[\sigma]| = |\tau| + 2|\sigma|$ (i.e. σ is fully used in $\tau[\sigma]$), the state s' in b_τ is mapped into a goal state in $b_{\tau[\sigma]}$ because σ is a plan for s'. Therefore, $b_{\tau[\sigma]} \subseteq S_G$ since goals are observable.

Consider the case $b_{\tau[\sigma]} \not\subseteq S_G$. Let $\tau[\sigma] = \langle a_0, o_0, \ldots, a_{n+k}, o_{n+k} \rangle$ where a_{n+k} is the last action in σ that appears in $\tau[\sigma]$; the action a_{n+k+1} in σ does not appear in $\tau[\sigma]$ because it is not applicable at $b_{\tau[\sigma]}$. By construction of $P[\tau, s']$, there is an execution $\tau' = \langle a_{n+1}, o'_{n+1}, \ldots \rangle$ and a state trajectory $\langle s'_{n+1}, a_{n+1}, \ldots \rangle$ that is seeded at s' and induced by τ' such that a_{n+j} is applicable at the belief $b_{\langle \tau, \tau'_j \rangle}$ where τ'_j is the prefix of τ' ending in o'_{n+j}, for $j = 1, 2, \ldots$. We then have two executions $\tau[\sigma]$ and $\langle \tau, \tau'_j \rangle$ such that

1. both executions end in observations: $\tau[\sigma]$ ends in o_{n+k} and $\langle \tau, \tau'_j \rangle$ ends in o'_{n+k},
2. both executions are different since a_{n+k+1} is applicable at $b_{\langle \tau, \tau'_j \rangle}$ but non applicable at $b_{\tau[\sigma]}$,
3. both executions have a non-empty *common prefix* (since the action a_{n+1} belongs to both) that ends in an observation.

Let τ'' be the largest common prefix of $\tau[\sigma]$ and $\langle \tau, \tau' \rangle$. Then, τ'' is a *branch point* of $\tau[\sigma]$ with $\tau < \tau'' < \tau[\sigma]$. By Lemma 4, π has asserting inference for the pair $(\tau[\sigma], \tau)$. □

Theorem 14. *Let P be a planning task that is solvable and with observable goals. The planner π_{single} is sound and complete for P.*

The planner π_{single} is quite similar to the "belief replanning" algorithm of Cassandra, Kaelbling, and Kurien (1996) but in the deterministic logical setting. Theorem 14 provides a strong completeness guarantee for π_{single} on solvable tasks with observable goals, while Cassandra, Kaelbling, and Kurien are not able to provide such guarantee.

For the second class of problems, we consider planning tasks that are strongly solvable, and a belief tracking component that is exact over such tasks. Let π_{all} be an online planner that implements exact belief tracking and that returns the set of all applicable actions at b_τ. Since π implements exact inference and is a covering planner, Theorem 11 gives

Theorem 15. *Let P be a strongly solvable planning task. The planner π_{all} is sound for P and complete under any randomized protocol L.*

Finally, for the last class of problems, we relax the assumption that π implements exact belief tracking by assuming that the tasks have observable goals:

Theorem 16. *Let P be a strongly solvable planning task with observable goals, and let π be a sound and covering planner for P. Then, π is complete for P under any randomized protocol L.*

Proof. By Theorem 11, the protocol and planner generate a goal-reaching execution τ. Since π is sound, b^π_τ contains a goal state. Since P has observable goals, $b^\pi_\tau \subseteq S_G$. □

Analysis and Variations of LW1

LW1 is an online planner for deterministic tasks that is complete on problems of *width* equal to 1 (Bonet and Geffner 2014b). This section briefly describes LW1, discusses its completeness with respect to the framework, and describes four variations that were implemented and tested.

Planning tasks in LW1 are specified with a language based on multivalued variables. A task $P = (V, I, A, G, W)$ is a tuple where V is a set of *state variables* X, each with a domain D_X, I is a valuation for all variables in V, G is a conjunction of X-literals (where an X-atom is a proposition of the form $X = x$, for some variable X and value $x \in D_X$, and an X-literal is an X-atom or its negation), A is a set of actions, and W specifies the sensing model. Each action $a \in A$ is given by its precondition $Pre(a)$ and a set of *conditional effects* of the form $a : C \rightarrow E$, where $Pre(a)$ and C are sets of X-literals, and E is a set of X-atoms (positive X-literals). The state space associated to problem P is the set of all complete valuations over the state variables.

The sensing model W consists of a collection of *observable variables* Y, each with domain D_Y, and boolean formulas $W_a(Y = y)$ over X-literals where a is an action, Y is an observable variable, and $y \in D_Y$ is a value for Y. There is no need to have formulas $W_a(Y = y)$ for each combination of a, Y and y. If there is no model for action a, the action has no observable effects. On the other hand, the existence of a formula $W_a(Y = y)$ entails that the atom $Y = y$ is observed after a is applied whenever the formula $W_a(Y = y)$ holds in the *resulting state*. There are some requirements that the sensing model needs to satisfy: (1) if $W_a(Y = y)$ is given for at least one value $y \in D_Y$, formulas $W_a(Y = y')$ for every other value $y' \in D_Y$ must be provided as well (else, the specification is incomplete), (2) the formula $W_a(Y = y) \wedge W_a(Y = y')$ must be inconsistent for every pair y, y' of different values for Y (else, the model is non-deterministic), and (3) the formula $\bigvee_{y \in D_Y} W_a(Y = y)$ must be valid (else, the observed value for Y may not be well defined).[2]

A crucial condition for completeness is that every formula $W_a(Y = y)$ must be in *positive DNF format* meaning that each sensing model must be a DNF formula without negative literals. This is a universal language in this setting as each boolean formula is equivalent to a DNF formula, and each negative literal $X \neq x$ is equivalent to $\bigvee_{x' \in D_X, x' \neq x} X = x'$. However, short formulas may suffer an exponential blow up when transformed into positive DNF format.

A task P induces a state model $S(P) = (S, A, S_{init}, S_G, f, O, \Omega)$ that we do not formalize here; see (Bonet and Geffner 2014b). Likewise, we do not formalize the notion of width but just say that the width $w(X)$ for variable X refers to the maximum number of non-determined variables that are relevant to X, while the width $w(P)$ of a task P is the maximum $w(X)$ over the variables X that appear in the goal, or in an action precondition.

Belief tracking is implemented at propositional level using K-literals of the form KL where L is an X-literal. The K-literals $KX = x$ and $KX \neq x$ are denoted by Kx and $K\bar{x}$ respectively. The interpretation of Kx (resp. $K\bar{x}$) is that the agent *knows* $X = x$ (resp. $X \neq x$). The knowledge dynamics is captured by replacing each effect $a : C \rightarrow E$ by *support rules* $a : KC \rightarrow KE$, where KC for $C = L_1 \wedge \cdots \wedge L_n$ is $KL_1 \wedge \cdots \wedge KL_n$, asserting that if the agent *knows* C then it would know E after the action a is applied. *Cancellation rules* of the form $a : \neg K\neg C \rightarrow \neg K\neg E$ are also required, where $\neg K\neg C$ stands for $\neg K\neg L_1 \wedge \cdots \wedge \neg K\neg L_n$. The cancellation rules assert that if the agent deems C as *possible* then it would deem E as possible after a. These rules combined with other rules called *action compilation*, plus the filtration described below, result in an exact factored belief tracking for tasks of width equal to 1 (Bonet and Geffner 2014b).

Beliefs in LW1 are represented as sets of K-literals; i.e., as the set of facts that are known to the agent. Given the belief $b^\pi_{\langle \tau, a \rangle}$, representing the belief $b_{\langle \tau, a \rangle}$, and upon observing o, the CNF theory $\Delta = b^\pi_{\langle \tau, a \rangle} \wedge D \wedge K_o$ is constructed and the new belief $b^\pi_{\langle \tau, a, o \rangle}$ is set to the collection of unit clauses in the closure of Δ by *unit resolution*. The clauses in D stand for deductive rules over the variables' domains, rules $Kx \Rightarrow K\bar{x}'$ for $X \in V$ and $x, x' \in D_X$ with $x \neq x'$, and rules $\bigwedge_{x' \in D_X, x' \neq x} K\bar{x}' \Rightarrow Kx$ for $X \in V$ and $x \in D_X$. The clauses in K_o encode the filtering entailed by the observation o. These are formulas $KC \Rightarrow K\neg L$ for all terms $C \wedge L$ in the DNFs $W_a(Y = y)$ such that $o \models Y \neq y$.

Completeness of LW1

LW1 guarantees completeness on tasks of width 1, and soundness on tasks of width greater than or equal to 1. As seen in experiments, LW1 seems to be complete on some tasks of width bigger than 1 (Bonet and Geffner 2014b).

The completeness over tasks of width 1 rests on two facts: an exact belief tracking and an action selection mechanism that conforms to a weak and effective planner. The action selection for a given execution τ is done by computing a plan for a classical task $H(P)$ whose encoding captures exact belief tracking on P and that permits the plan to make assumptions about the outcome of the observable variables. During plan execution, if one of the assumed observations does not materialize, the requirements on the form of the sensing model plus the width $w(P) = 1$ of the task (provably) imply that at least one piece of information is *learned by the agent*. This information is either that $X = x$ holds or $X = x$ is impossible for some *state variable* X and value $x \in D_X$. Thus, the belief state after the execution $\tau[\sigma]$ is either a goal belief, if σ is fully executed, or it has cardinality strictly smaller than the cardinality of b_τ for execution τ.

Variations on LW1

We build on top of LW1's implementation that provides the planner, protocol and benchmarks.

[2] LW1 also accepts preconditions $Pre(Y)$ for observable variables Y, but they are not considered here.

The first variation, denoted by LW1[WL], is essentially LW1 but where unit propagation is implemented using watched literals (Moskewicz et al. 2001; Qu 2006). LW1[WL] also improves other aspects that result in improved performance and provides the baseline for comparisons. LW1[WL] has thus the same completeness guarantees of LW1, being complete over tasks of width equal to 1.

The second variation, denoted by LW1[R], is obtained by using LW1[WL]'s belief tracking but replacing the action selection with a simple randomized strategy: if b_τ is a singleton, use the action selection in LW1[WL], else choose a random applicable action. We thus basically move the action selection performed by a randomized protocol inside LW1. By Theorem 16, LW1[R] is complete for strongly-solvable tasks with observable goals.

In the third variation, denoted by LW1[AC3], we enhance the inferential power of belief tracking on problems of width larger than 1 by replacing unit propagation with an inference done by arc consistency, implemented by AC3, on a CSP (Mackworth 1977; Russell and Norvig 2009). This variation only applies to problems in which all the hidden variables are *static*, meaning that the variables are not affected by the actions. On these problems, we construct a CSP with two types of variables: one variable X with domain D_X for each hidden state variable X, and one variable M_B for each sensing model $W_a(Y = y)$ that refers to a subset B of state variables and whose domain D_{M_B} consists of all joint valuations for the variables in B. The CSP has binary constraints that relates the variables X with the variables M_B, for $X \in B$, and that enforce the consistency of the valuations. Each time that an observation o such that $o \models Y \neq y$ is received after applying action a, all the joint valuation in the domain of M_B that make $W_a(Y = y)$ true are removed, and consistency is re-established with AC3. This is basically a restricted implementation of beam tracking (Bonet and Geffner 2014a).

The last variation, denoted by LW1[AC3R], uses the AC3 inference combined with randomized action selection. A characterization of the classes of tasks on which LW1[AC3] and LW1[AC3R] are complete is left for future work.

Experiments

The experiments were performed using FF (Hoffmann and Nebel 2001) as the underlying classical planner on an Amazon EC2 cluster consisting of Intel Xeon ES-2686 v4 CPUs running at 2.30 GHz and with limits of 8Gb of RAM and 3 hours of time. We compare the four variations of LW1 with results reported by Bonet and Geffner (2014b).

Table 1 shows the results for LW1[WL] and LW1[R] on the set of standard benchmarks. The table reports figures for number of problems in the benchmark (#sim), number of problems solved, avg. calls to the classical planner, avg. length of the executions found, avg. total time, avg. preprocessing time, and the avg. time invested in finding the execution without considering preprocessing time. Preprocessing time includes the time spent for parsing and grounding the input files, but most importantly, it includes the preprocessing time for the underlying classical planner across all the calls. As the underlying planner is used off-the-shelf, it parses and grounds the same domain each time it is called. On simple classical tasks, very often, preprocessing is responsible for the major fraction of runtime.

For lack of space we do not include the results of Bonet and Geffner (2014b), yet we observe that LW1[WL] improves with respect to LW1 on every domain except doors that has similar performance, and diagonal wumpus that has degraded performance. On the other hand, the experiments were performed on faster CPUs, but the improvements of one or more orders of magnitude in localize and rocksample cannot be explained by the differences in hardware. LW1[R] has in general better total time than LW1[WL] as it makes very few calls to the classical planner, except when the resulting executions are excessively long like in doors and wumpus. As predicted, LW1[R] remains complete on the tasks that comply with the conditions of Theorem 11. Medpks is a task with easily reachable dead-ends and LW1[R] solves no instance.

Table 2 shows results when an inference based on AC3 is used. We see a behavior similar to Table 1 in which the randomized planner is usually faster as it makes very few calls to the underlying classical planner. In the mines domain the coverage increases significantly as the inference provided by AC3 is stronger than unit propagation on this domain (Bonet and Geffner 2014a). LW1[AC3] fails to solve an instance of mines when the underlying planner fails to compute a weak plan. As LW1[AC3R] makes much fewer calls to the underlying planning, we do not observe this behaviour for LW1[AC3R] on mines. On the other hand, observe that LW1[AC3R] finds optimal executions in mines because all valid executions on mines have equal length.

Summary

We presented a formal framework for understanding and reasoning about online planning for deterministic tasks in which planners are characterized by their belief tracking and actions selection components, and in which their use by an agent is formalized with the notion of online protocol. We then defined formal properties about belief tracking, action selection, and the relation between the two, and obtained sufficient conditions for the completeness of planners. We implemented and tested four variations of the state-of-the-art LW1 online planner. In the experiments we observed that replacing unit propagation with a more powerful, but still tractable, AC3 algorithm increases the scope of tasks that can be solved. Likewise, we identified a simple randomized variation of LW1 that is complete, when combined with a suitable belief tracking component, on non-trivial and interesting classes of problems.

For the future, we want to study necessary conditions for completeness, extend results over non-deterministic tasks, and incorporate notions of landmarks in order to weaken the requirements for the action selection mechanism.

Acknowledgements. We thank the anonymous reviewers for useful comments and pointers to related work.

			LW1 using UP with watched literals (LW1[WL])						Randomized LW1[WL] (LW1[R])					
				average		avg. time in seconds				average		avg. time in seconds		
domain	problem	#sim	solved	calls	length	total	prep.	exec.	solved	calls	length	total	prep.	exec.
clog	7	12	12	2.25	17.75	0.14	0.12	0.02	12	1.00	24.58	0.14	0.12	0.01
clog	huge	3125	3125	6.97	45.59	5.91	5.37	0.54	3125	1.00	160.80	7.54	7.36	0.18
colorballs	9-5	1000	1000	64.65	127.15	342.40	335.65	6.75	1000	1.00	1458.74	17.18	13.14	4.04
colorballs	9-7	1000	1000	68.78	147.31	615.22	604.87	10.35	1000	1.00	1906.71	30.38	22.03	8.35
doors	17	1000	1000	62.16	119.67	446.25	441.38	4.87	1000	1.00	1357.29	31.35	22.26	9.09
doors	19	1000	1000	78.42	160.51	1058.41	1049.58	8.83	1000	1.00	1783.98	58.41	39.84	18.57
ebtcs	50	50	50	25.50	26.48	1.84	1.41	0.43	50	1.00	46.40	0.12	0.11	0.01
ebtcs	70	70	70	35.50	36.49	4.96	4.14	0.82	70	1.00	71.93	0.18	0.16	0.01
localize	15	134	134	8.34	15.00	4.04	3.79	0.25	134	1.00	416.22	0.82	0.60	0.21
localize	17	169	169	9.75	17.00	12.77	12.36	0.41	169	1.00	569.71	1.58	1.10	0.48
medpks	150	151	151	1.99	1.99	20.49	20.39	0.11	0	—	—	—	—	—
medpks	199	200	200	2.00	2.00	42.52	42.35	0.16	0	—	—	—	—	—
rocksample	8-12	1000	1000	6.00	96.60	1.78	0.82	0.96	1000	0.00	6494.78	0.40	0.21	0.19
rocksample	8-14	1000	1000	6.73	122.98	3.01	0.85	2.16	1000	0.00	7733.12	0.45	0.22	0.23
unix	3	28	28	17.00	46.54	1.88	1.59	0.29	28	1.00	406.39	0.30	0.26	0.04
unix	4	60	60	33.00	93.72	18.34	17.54	0.80	60	1.00	955.70	4.61	4.24	0.38
wumpus	5d	8	8	2.25	16.25	0.10	0.08	0.02	8	1.00	61.50	0.15	0.13	0.01
wumpus	10d	256	256	4.21	32.96	2.18	1.98	0.20	256	0.83	667.40	4.14	3.51	0.63
wumpus	15d	1000	1000	5.09	46.06	31.31	30.48	0.82	1000	0.77	1699.13	39.93	31.31	8.61
wumpus	20d	1000	1000	5.04	55.69	166.62	163.95	2.67	1000	0.86	3676.16	293.21	207.44	85.77
wumpus	25d	1000	1000	5.21	66.36	558.27	551.89	6.38	1000	0.70	5002.95	875.57	590.50	285.06
mines	4x3	100	34	2.71	14.00	1.14	1.10	0.04	34	0.00	14.00	0.13	0.13	0.00
mines	5x3	100	43	3.16	17.00	2.40	2.33	0.07	43	0.00	17.00	0.21	0.21	0.01
mines	4x4	100	35	3.91	18.00	8.91	8.75	0.16	35	0.00	18.00	0.77	0.75	0.02
mines	5x5	100	48	5.83	27.00	102.71	101.77	0.94	48	0.00	27.00	7.47	7.31	0.16
mines	6x6	100	37	8.51	38.00	599.53	594.15	5.38	37	0.00	38.00	52.24	51.51	0.73
mines	8x8	100	16	11.75	66.00	4003.97	3959.19	44.77	43	0.00	66.00	205.82	199.81	6.01
wumpus	5x5	100	100	9.95	13.65	0.79	0.59	0.20	100	0.00	132.66	0.24	0.21	0.04
wumpus	10x10	100	100	42.13	48.94	33.69	30.71	2.98	100	0.00	659.29	4.72	3.50	1.21
wumpus	15x15	100	100	105.04	118.35	769.96	754.05	15.90	100	0.00	1670.83	37.43	19.69	17.74

Table 1: Results for variations of LW1 where inference is done by unit propagation with watched literals and the action selection is done by either using a classical planner (LW1[WL]) or using a randomized criterion (LW1[R]). Dash (—) means that the planner solves no instance. Key columns are highlighted in gray.

			LW1 with AC3 (LW1[AC3])						Randomized LW1[AC3] (LW1[AC3R])					
				average		avg. time in seconds				average		avg. time in seconds		
domain	problem	#sim	solved	calls	length	total	prep.	exec.	solved	calls	length	total	prep.	exec.
doors	17	1000	1000	62.07	119.33	472.10	467.68	4.41	1000	1.00	1357.29	25.03	23.70	1.33
doors	19	1000	1000	78.34	161.97	1097.00	1089.35	7.65	1000	1.00	1783.98	44.58	42.51	2.07
ebtcs	50	50	50	25.50	26.48	1.88	1.47	0.42	50	1.00	46.40	0.09	0.09	0.01
ebtcs	70	70	70	35.50	36.49	5.41	4.60	0.81	70	1.00	71.93	0.14	0.13	0.01
mines	4x3	100	100	3.56	14.00	1.07	0.92	0.16	100	0.00	14.00	0.17	0.14	0.03
mines	5x3	100	100	3.78	17.00	2.43	2.17	0.27	100	0.00	17.00	0.29	0.23	0.06
mines	4x4	100	91	4.65	18.00	8.19	6.72	1.48	100	0.00	18.00	1.02	0.93	0.09
mines	5x5	100	98	5.66	27.00	145.47	85.28	60.19	100	0.00	27.00	11.23	10.79	0.43
mines	6x6	100	85	8.04	38.00	740.46	563.27	177.19	100	0.00	38.00	59.48	58.30	1.17
mines	8x8	100	30	10.60	66.00	4051.16	3602.81	448.35	100	0.00	66.00	194.57	189.81	4.76
wumpus	5x5	100	100	9.95	13.65	1.84	1.59	0.25	100	0.00	132.87	0.19	0.16	0.03
wumpus	10x10	100	100	42.13	48.94	128.66	125.21	3.45	100	0.00	659.71	3.79	2.86	0.93
wumpus	15x15	100	100	105.04	118.35	3733.90	3714.60	19.30	100	0.00	1671.56	41.68	27.85	13.82

Table 2: Results for variations of LW1 where inference is done by AC3 and the action selection is done by either using a classical planner (LW1[AC3]) or using a randomized criterion (LW1[AC3R]). Evaluation is done in problems where the hidden variables are static. Dash (—) means that the planner solves no instance. Key columns are highlighted in gray.

References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. 21st Int. Joint Conf. on Artificial Intelligence*, 1623–1628.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *Proc. 22nd Int. Joint Conf. on Artificial Intelligence*, 1936–1941.

Bonet, B., and Geffner, H. 2014a. Belief tracking for planning with sensing: Width, complexity and approximations. *Journal of Artificial Intelligence Research* 50:923–970.

Bonet, B., and Geffner, H. 2014b. Flexible and scalable partially observable planning with linear translations. In *Proc. 28th AAAI Conf. on Artificial Intelligence*, 2235–2241.

Bonet, B. 2009. Deterministic POMDPs revisited. In Bilmes, J., and Ng, A., eds., *Proc. 25th Conf. on Uncertainty in Artificial Intelligence*, 59–66. Montreal, Canada: AUAI Press.

Brafman, R. I., and Shani, G. 2012. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research* 1(45):565–600.

Brafman, R. I., and Shani, G. 2016. Online belief tracking using regression for contingent planning. *Artificial Intelligence* 241:131–152.

Cassandra, A. R.; Kaelbling, L. P.; and Kurien, J. 1996. Acting under uncertainty: Discrete bayesian model for mobile robot navigation. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robot and Systems*.

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

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

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In Biundo, S., and Fox, M., eds., *Proc. 5th European Conf. on Planning*, 308–318. Durham, UK: Springer: LNCS 1809.

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

Koenig, S., and Simmons, R. 1996. Easy and hard testbeds for real-time search algorithms. In *Proc. 13th Nat. Conf. on Artificial Intelligence*, 279–285.

Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8:99–118.

Maliah, S.; Brafman, R. I.; Karpas, E.; and Shani, G. 2014. Partially observable online contingent planning using landmark heuristics. In *Proc. 24nd Int. Conf. on Automated Planning and Scheduling*, 163–171.

Moskewicz, M. W.; Madigan, C. F.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient sat solver. In *Proc. 38th Annual Design Automation Conference*, 530–535. New York, USA: ACM.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35:623–675.

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proc. 6th Int. Conf. on Artificial Intelligence Planning Systems*, 212–222. Toulouse, France: AAAI Press.

Qu, S. 2006. Fast incremental unit propagation by unifying watched-literals and local repair. Master's thesis, Massachusetts Institute of Technology.

Rintanen, J. 2004. Complexity of planning with partial observability. In Zilberstein, S.; Koenig, S.; and Koehler, J., eds., *Proc. 14th Int. Conf. on Automated Planning and Scheduling*, 345–354. Whistler, Canada: AAAI Press.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition.