Journal of Artificial Intelligence Research 50 (2014) 923–970 Submitted 06/14; published 08/14

Belief Tracking for Planning with Sensing: Width, Complexity and Approximations

Blai Bonet BONET@LDC.USB.VE
Departamento de Computación
Universidad Simón Bolívar
Caracas, Venezuela

Hector Geffner HECTOR.GEFFNER@UPF.EDU
ICREA & Universitat Pompeu Fabra
Roc Boronat 138
08018 Barcelona, Spain

Abstract

We consider the problem of belief tracking in a planning setting where states are valuations over a set of variables that are partially observable, and beliefs stand for the sets of states that are possible. While the problem is intractable in the worst case, it has been recently shown that in *deterministic* conformant and contingent problems, belief tracking is exponential in a width parameter that is often bounded and small. In this work, we extend these results in two ways. First, we introduce a width notion that applies to *non-deterministic* problems as well, develop a factored belief tracking algorithm that is exponential in the problem width, and show how it applies to existing benchmarks. Second, we introduce a meaningful, powerful, and sound approximation scheme, *beam tracking*, that is exponential in a smaller parameter, the problem *causal width*, and has much broader applicability. We illustrate the value of this algorithm over large instances of problems such as Battleship, Minesweeper, and Wumpus, where it yields state-of-the-art performance in real-time.

1. Introduction

Planning with incomplete information can be formulated as a search problem in belief space where two issues need to be addressed: keeping track of beliefs, and searching for a goal belief (Bonet & Geffner, 2000). While the two tasks are intractable in the worst case over compact representations, this is the approach adopted in most recent conformant and contingent planners where beliefs are handled using SAT, regression techniques, or logical normal forms such as CNF, DNF, and OBDDs, and the search for goal beliefs is guided by domain-independent heuristics (Bertoli, Cimatti, Roveri, & Traverso, 2001; Hoffmann & Brafman, 2006; Bryce, Kambhampati, & Smith, 2006; To, Pontelli, & Son, 2011; Shani & Brafman, 2011; Brafman & Shani, 2012).

Recently, the complexity of belief tracking in *deterministic* conformant and contingent planning has been shown to be exponential in a problem *width* parameter that is often bounded and small (Palacios & Geffner, 2009; Albore, Palacios, & Geffner, 2009). The bound follows from a family of translations developed for compiling planning problems over beliefs into planning problems over states. The translations are exponential in the problem width, and for deterministic conformant problems result in problems that can be solved by classical planners.

The difficulty in extending the results of Palacios, Albore, and Geffner to the non-deterministic setting is a consequence of the special role played by the initial situation in deterministic problems. In such a case, all uncertainty, and in particular, the uncertainty about observations, action preconditions, and goals, which is the one that matters in a complete planner, is the result of the uncertainty about the initial situation. In the non-deterministic setting, on the other hand, uncertainty is produced dynamically as a result of the application of non-deterministic actions. Moreover, while an uncertain initial situation can always be modeled by a fully known initial situation and a dummy non-deterministic action, the opposite transformation is not as simple. Indeed, non-deterministic effects can be compiled into deterministic effects that are conditional on the value of hidden variables, but the number of hidden variables required must grow then with the planning horizon (Weld, Anderson, & Smith, 1998; Albore, Ramirez, & Geffner, 2010).

The aim of this work is the study of the computational complexity of belief tracking in terms of novel width parameters that apply to both deterministic and non-deterministic planning problems, and the formulation of practical approximate belief tracking algorithms that can be efficient and effective even for problems with large width. We will achieve this by considering two *decomposition schemes* for belief tracking, and three algorithms based on these decompositions. More precisely, we introduce:

1. A *width* notion for planning that is in close correspondence with the notion introduced by Palacios, Albore, and Geffner but which applies to non-deterministic problems as well.

2. A first belief tracking algorithm, *factored belief tracking*, that is sound and complete for both deterministic and non-deterministic problems P, and runs in time and space exponential in the problem width $w(P)$. The algorithm is based on a *decomposition* of the problem P into projected subproblems P_X, one for every goal and precondition variable X, each one including the variables that are *relevant* to X.

3. A second belief tracking algorithm, *causal belief tracking*, that is based on an alternative decomposition scheme, where subproblems P_X are defined for every goal, precondition, and *observable* variable X, each one including the variables that are *causally relevant* to X. The algorithm is sound and complete for a large and meaningful class of problems, and while it is still *time exponential* in the problem width, it is *space exponential* in the *causal width* of the problem that is often much smaller.

4. A final belief tracking algorithm, *beam tracking* that is a sound but incomplete approximation of causal belief tracking, and is often practical enough, even in problems with large widths, as it runs in time and space that are exponential in the problem *causal width.*

The power of the last algorithm, *beam tracking*, will be shown empirically over large instances of problems such as Minesweeper, Battleship, and Wumpus, where state-of-the-art performance is obtained in real-time by combining the belief tracking algorithm with simple heuristics for action selection.[1]

The organization of the paper follows this structure, preceded by an overview of the relevant notation and background, and followed by a description of the experiments, a discussion of related work, and a summary. The paper integrates results from two conference papers (Bonet & Geffner, 2012b, 2013), providing proofs and additional details. The work is related to other proposals for tractable forms of belief tracking in logical and probabilistic frameworks (Doucet, Freitas, Murphy, & Russell, 2000; Amir & Russell, 2003), yet there are two key differences. One is that we start with an *exact* account that is used to determine with *certainty* whether the goal has been achieved or an action is applicable. The second is that belief tracking accounts in *planning* do not have to be *complete over all formulas*. In order to have a sound and complete planner, only the beliefs over observations, action preconditions, and goals are required. This is important because observations, action preconditions, and goals are given, and the structure of the actions, sensors, and goals can be exploited to track those beliefs more efficiently. This observation is implicit in 'lazy' belief tracking schemes for planning with incomplete information that appeal to SAT-solvers (Hoffmann & Brafman, 2006) or regression (Shani & Brafman, 2011). We'll say more about related work in Section 12.

2. Model

The model for *planning with sensing* is a simple extension of the model for conformant planning where a goal is to be achieved with certainty in spite of uncertainty in the initial situation or action effects (Goldman & Boddy, 1996; Smith & Weld, 1998). The model for *conformant planning* is characterized by a tuple $\mathcal{S} = \langle S, S_0, S_G, A, F \rangle$ where

- S is a finite state space,
- S_0 is a non-empty set of possible initial states, $S_0 \subseteq S$,
- S_G is a non-empty set of goal states, $S_G \subseteq S$,
- A is a set of actions with $A(s)$ denoting the sets of actions applicable in $s \in S$, and
- F is a non-deterministic state-transition function such that $F(a, s)$ denotes the non-empty set of possible successor states that follow action a in s, for $a \in A(s)$.

A solution to a conformant model is an action sequence that maps each possible initial state into a goal state. More precisely, $\pi = \langle a_0, \ldots, a_{n-1} \rangle$ is a conformant plan if for each possible sequence of states $s_0, s_1, \ldots, s_n$ such that $s_0 \in S_0$ and $s_{i+1} \in F(a_i, s_i)$, $i = 0, \ldots, n-1$, action a_i is applicable in s_i and s_n is a goal state.

Conformant planning can be cast as a path finding problem over *beliefs*, defined as the sets of states that are deemed possible at any time point (Bonet & Geffner, 2000). The initial belief b_0 is S_0, and the belief b_a that results from an action a in a belief state b is:

$$b_a = \{ s' \,|\, \text{there is a } s \in b \text{ such that } s' \in F(a, s) \} \,, \tag{1}$$

1. A real-time animation of the algorithm for several instances of Minesweeper can be seen in `https://www.youtube.com/watch?v=U98ow4n87RA`, while all the source code and graphical interfaces can be obtained at `http://code.google.com/p/belief-tracking`.

where the action a is applicable in b if it is applicable in each state s in b. In this formulation, a conformant plan is an action sequence that maps the initial belief b_0 into a goal belief b_G; i.e., a set of goal states.

Contingent planning or *planning with sensing* is planning with both uncertainty and feedback. The model for contingent planning is the model for conformant planning extended with a *sensor model*. A sensor model is a function $O(s, a)$ mapping state-action pairs into observations tokens o. The expression $o \in O(s, a)$ means that token o is a possible observation when s is the true state of the system and a is the last action done. The observed token o provides partial information about the true but possibly hidden system state as the same token may be possible in different states. If two different tokens o_1 and o_2 belong to $O(s, a)$, it means that either one can be observed in s when a is the last action. Sensing is *deterministic* or *noiseless* when $O(s, a)$ contains one token, else it is *non-deterministic* or *noisy*. The contingent model is similar to POMDPs (Kaelbling, Littman, & Cassandra, 1999) but with uncertainty encoded through sets of states rather than probability distributions.

Executions in the contingent setting are sequences $\langle a_0, o_0, a_1, o_1, \ldots \rangle$ of pairs of actions a_i and observations o_i. If $b = b_i$ is the belief state when the action a_i is applied and o_i is the token that is observed, then the belief b_a after the action $a = a_i$ is given by (1), and the belief $b_{i+1} = b_a^o$ that follows from observing the token o is:

$$b_a^o = \{s \mid s \in b_a \text{ and } o \in O(s, a)\}\,. \tag{2}$$

An execution $\langle a_0, o_0, a_1, o_1, \ldots \rangle$ is *possible* if starting from the initial belief b_0, each action a_i is applicable in the belief b_i (i.e., $a_i \in A(s)$ for all $s \in b_i$), for $i \geq 0$, and each belief b_i is not empty.

In *off-line* contingent planning, an action selection strategy is sought that ensures that *all possible executions* end up in a goal belief. In *on-line* contingent planning, an action selection strategy is sought that ensures that the *single execution* that results from the interaction with the real system or simulator, ends up in a goal belief. In both cases, the action selection strategy can be expressed as a *partial* function π over beliefs, called a *policy*, such that $\pi(b)$ is the action to do in belief b. The function is partial because it has to be defined only over the initial belief b_0 and some non-goal beliefs b; namely, those that can be reached with π from b_0 in off-line planning, and those that have been reached with π from b_0 in on-line planning.

3. Language

Syntactically, conformant problems can be expressed in *compact form* through a set of *state variables*, which for convenience we assume to be *multi-valued*.[2] More precisely, a conformant planning problem is a tuple $P = \langle V, I, A, G \rangle$ where V stands for the problem variables X, each one with a finite and discrete domain D_X, I is a set of clauses over the V-literals defining the initial situation, A is a set of actions, and G is a set of V-literals defining the goal. Every action a has a precondition $Pre(a)$ given by a set of V-literals, and

2. Multi-valued variables can be compiled into boolean variables but the compilation affects the syntactic structure of the problem. In principle, such a structure could be recovered from boolean encodings but this would result in a more complex formulation.

a set of conditional effects $C \rightarrow E_1 | \ldots | E_n$ where C and each E_i are sets (conjunctions) of V-literals. The conditional effect is *non-deterministic* if $n > 1$; else $n = 1$ and the effect is deterministic.

A problem $P = \langle V, I, A, G \rangle$ defines a conformant model $\mathcal{S}(P) = \langle S, S_0, S_G, A, F \rangle$, where S is the set of possible valuations over the variables in V, S_0 and S_G are the sets of valuations that satisfy I and G respectively, $A(s)$ is the set of operators whose preconditions are true in s, and $F(a, s)$ is the non-deterministic transition function that results from collecting the successor states that may follow from a by selecting one head E_i from each conditional effect $C \rightarrow E_1 | \ldots | E_n$ whose body C is true in s.[3]

Contingent problems can be described by extending the syntactic description of conformant problems with a compact encoding of the *sensor model*. For this, we assume a set V' of observable multi-valued variables Y, not necessarily disjoint with the state variables V (i.e., some state variables may be observable), and formulas $W_a(Y = y)$ over the state variables, for each action a and each possible value y of each observable variable Y. The formula $W_a(Y = y)$ implicitly encodes the states over which the observation literal $Y = y$ is possible when a is the last action executed. The formulas $W_a(Y = y)$ for the different y values in D_Y must be logically *exhaustive*, as every state-action pair must give rise to some observation $Y = y$. If in addition, the formulas $W_a(Y = y)$ for the different y values are logically *exclusive*, then every state-action pair gives rise to a single observation $Y = y$ and the sensing over Y is deterministic. If a state variable X is observable, then $W_a(X = x)$ is just the formula $X = x$.

A contingent problem P is a tuple $P = \langle V, I, A, G, V', W \rangle$ that defines a contingent model which is made of the conformant model $\langle S, S_0, S_G, A, F \rangle$ determined by the first four components in P, and the sensor model $O(a, s)$ determined by the last two components, where $o \in O(a, s)$ iff o is a valuation over the observable variables $Y \in V'$ such that $Y = y$ is true in o only if the formula $W_a(Y = y)$ in W is true in s for $y \in D_Y$.

This is a standard language for representing contingent problems in compact form featuring both incomplete information, and non-deterministic actions and sensors. Its two distinctive features in relation to similar languages are the use of multi-valued variables, and the distinction between state and observable variables.

As an illustration, if X encodes the position of an agent, and Y encodes the position of an object that can be seen by the agent when $X = Y$, we can have an observable variable $Z \in \{Yes, No\}$ encoding whether the object can be seen by the agent or not, defined by the formulas $W_a(Z = Yes) = \bigvee_{l \in D}(X = l \wedge Y = l)$, and $W_a(Z = No) = \neg \bigvee_{l \in D}(X = l \wedge Y = l)$, where D is the set of possible locations and a is any action. This will be a deterministic sensor. A non-deterministic sensor could be used if, for example, the agent cannot detect the presence of the object at certain locations $l \in D'$. For this, it suffices to push the disjunct $\bigvee_{l \in D'}(X = l)$ into the formulas characterizing $W_a(Z = Yes)$ and $W_a(Z = No)$, so that the two observations $Z = Yes$ and $Z = No$ would be possible when the agent is in a position $l \in D'$.

Since a conformant problem $\langle V, I, A, G \rangle$ can be expressed as a contingent problem $\langle V, I, A, G, V', W \rangle$ with just one (dummy) observable variable Z, with $Z \notin V$ and domain

3. These conditional effects must be consistent in the sense that is explained below.

$D_Z = \{\top\}$, and observation model $W_a(Z = \top) = true$ for every action a, we will focus from now on on the more general contingent problem.

Likewise, for convenience, if a variable Y is boolean, we often represent the literals $Y = true$ and $Y = false$ as Y and $\neg Y$. Similarly, if variable Y is observable, unless stated otherwise, we assume that the observation model for Y is deterministic so that the formula $W_a(Y = false)$ becomes the complement of the formula $W_a(Y = true)$.

4. Belief Tracking Problem and Flat Belief Tracking Algorithm

An *execution* over the problem $P = \langle V, I, A, G, V', W\rangle$ is a sequence $\langle a_0, o_0, a_1, o_1, \ldots\rangle$ of actions a_i and observations o_i such that each a_i is in A and each observation o_i is a full valuation over observation variables in V'. An execution $\langle a_0, o_0, \ldots, a_n, o_n\rangle$ is *possible* over a problem P with a non-empty belief state b_0, if it generates a sequence of beliefs $b_0, \ldots, b_n$ such that the preconditions of the action a_i are true in the belief b_i, and the belief states b_i are not empty. The problem of *belief tracking* in contingent planning is the problem of determining if an execution is possible and if the final belief state achieves the goal:

Definition 1. *Belief tracking in planning (BTP) is the problem of determining whether an execution $\langle a_0, o_0, a_1, o_1, \ldots\rangle$ over a planning problem $P = \langle V, I, A, G, V', W\rangle$ is possible, and if so, whether the resulting belief state makes the goal G true.*

A complete planner needs to solve this problem for determining which actions are applicable after a given execution, what observations may result, and whether the goal has been achieved. The machinery that we will develop is aimed at the slightly more general belief tracking problem over *generalized executions*: these are executions $\langle a_0, o_0, a_1, o_1, \ldots\rangle$ where the observations o_i are *partial* rather than *full* valuations over the observable variables. Moreover, it suffices to consider generalized executions where the observations are valuations over a single observable variable. Such observations o_i can be represented by *observation literals* ℓ_i:

Definition 2. *Generalized belief tracking in planning (GBTP) is the problem of determining whether a generalized execution $\langle a_0, \ell_0, a_1, \ell_1, \ldots\rangle$ over a planning problem $P = \langle V, I, A, G, V', W\rangle$ is possible, and if so, whether it achieves a given goal, precondition, or observation literal.*

Given a procedure for deciding GBTP, it is simple to decide BTP over an execution τ by calling the procedure for deciding GBTP over the generalized execution τ' that replaces each observation o_i by a sequence of the observation literals that are true in o_i separated by NO-OP actions (actions with no effects).

Proposition 3. *BTP is polynomial-time reducible to GBTP.*

While our interest is in belief tracking for planning, we will find it convenient to focus on the generalized problem, as none of the belief update equations or algorithms is sensitive to this distinction. For simplicity, however, we will just talk about belief tracking, and make explicit the distinctions between BTP and GBTP, and between normal and generalized executions, when needed.

The plain solution to the belief tracking problem is given by the updates expressed in Eqs. 1 and 2, where belief states are explicitly represented as sets of states, states are full valuations over the state variables, and the actions, transition function, and observations are obtained from the syntactic representation of the problem:

Definition 4. *The* flat belief tracking *algorithm over an execution* $\langle a_0, o_0, a_1, o_1, \ldots \rangle$ *and problem* P*, starts with the belief* b_0 *that contains the states that satisfy the initial situation, setting the next belief state* b_{i+1} *to* b_a^o *using* (1) *and* (2) *with* $b = b_i$*,* $a = a_i$*, and* $o = o_i$*.*

The complexity of flat belief tracking is exponential in the number of state variables. Yet, often some state variables do not add up to the complexity of tracking beliefs. Syntactically, this happens when a state variable X is initially known, all variables Y that are causally relevant to X (see below) are initially known as well, and neither X nor any variable Y causally relevant to X appears in the head of a non-deterministic effect. We say that these variables are *determined* as their value in every reachable belief is known, and can be fully predicted from the preceding actions and their preceding values. For example, the variable that encodes the position of the agent in the Wumpus game is determined, as its initial value is known and the effect of the actions on the variable is deterministic and depends only on its previous value.

Formally, we define the set of variables that are *determined* in a problem to be the largest set of state variables X in the problem that are initially known such that every state variable X' that is causally relevant to X belongs to the set. This set of variables is easily identifiable in low polynomial time. The complexity of flat belief tracking can be then expressed as follows:

Theorem 5. *Flat belief tracking is exponential in* $|V_U|$*, where* $V_U = V \setminus V_K$ *and* V_K *is the set of state variables that are determined in the problem.*

Given this result, the first question that arises is how bad is the naive approach of flat belief tracking. Interestingly, the following result for the decision problem shows that flat belief tracking is not bad in the worst case:

Theorem 6. *BTP and GBTP are Turing complete for the class* P^{NP}*.*

That is, BTP and GBTP can be decided in polynomial time using an oracle for NP (SAT, for example), and every decision problem that can be decided in polynomial time with such an oracle, can be decided in polynomial time with an oracle for BTP or GBTP. The complexity class P^{NP} includes the classes NP and coNP, and it is contained in PSPACE (Sipser, 2006).

5. Structure and Width

It is possible to improve on the complexity of flat belief tracking over a specific problem by exploiting the structure of the problem. Before introducing the graph that captures this structure, it will be convenient to make explicit some assumptions that do not restrict the generality of the approach but make the definitions simpler. *First*, we assume that the formula I encoding the initial situation contains just positive or negative literals; i.e., *unit clauses* only. This is not a restrictive assumption since any set of clauses can be encoded with the help of dummy observations. *Second*, we assume that non-deterministic effects involve just one variable in their heads. Again, this can always be achieved by adding extra variables and effects. For example, the non-deterministic effect $X \rightarrow Y \wedge Z \mid \neg Y \wedge \neg Z$ of an action can be replaced by the deterministic effects $X \wedge W \rightarrow Y \wedge Z$ and $X \wedge \neg W \rightarrow \neg Y \wedge \neg Z$, along with the non-deterministic effect $true \rightarrow W \mid \neg W$, where W is a new 'random' boolean variable that is initially unknown and changes randomly. *Third*, we assume that the problem is *consistent*, meaning that the initial situation I is logically consistent so that the initial belief state b_0 is not empty, and that the effects of any action a are consistent so that the heads of the deterministic conditional effects that are applicable in a reachable state s, along with any choice of heads of the non-deterministic conditional effects that are applicable in s, are jointly consistent.[4] *Last*, we assume that every observable variable is *relevant* to a variable appearing in some precondition or goal, with the notion of relevance to be spelled below. Observable variables that don't comply with this condition can be eliminated from the problem with no relevant information loss.

5.1 Relevance and Width

For a variable X, whether a state variable, an observable variable, or both, the *immediate causes* of X are defined as follows:

Definition 7. *A variable X is an immediate cause of a variable Y in a problem P, written $X \in Ca(Y)$, iff $X \neq Y$, and either X occurs in the body C of a conditional effect $C \rightarrow E_1 | \cdots | E_n$ and Y occurs in a head E_i, $1 \leq i \leq n$, or Y is an observable variable and X occurs in a formula $W_a(Y = y)$ for some $y \in D_Y$ and some action a.*

Basically, X is an immediate cause of Y when uncertainty about X may affect the uncertainty about Y directly, not through other variables. X is not necessarily an immediate cause of Y when X appears as the precondition of an action that affects Y, as preconditions must be known with certainty, and hence, do not propagate uncertainty. The notion of *causal relevance* is given by the transitive closure of the immediate cause relation:

Definition 8. *X is* causally relevant *to Y in P if $X = Y$, $X \in Ca(Y)$, or X is causally relevant to a variable Z that is causally relevant to Y.*

In order to test whether a given literal $Z = z$ is known after a certain execution $\langle a_0, a_1, \ldots, a_i \rangle$ of actions in the *conformant setting*, it is possible to show that one can just progress the state over the variables X that are causally relevant to Z:

Proposition 9. *Belief tracking in the deterministic or* non-deterministic conformant *setting is exponential in the maximum number of non-determined variables that are all causally relevant to a variable appearing in an action precondition or goal.*

This bound is closely related to the bound obtained by Palacios and Geffner in the deterministic setting. Indeed, if we refer to the number of non-determined state variables

4. From a semantic point of view, this means that a state s' is a possible successor of state s after an action a applicable in s, i.e. $s' \in F(a, s)$, iff for every literal $X = x$ true in s', $X = x$ is in the head of a deterministic or non-deterministic conditional effect of the action a whose body is true in s, or if $X = x$ is true in s, and there is no effect of the action a with $X = x'$ in the head, with $x' \neq x$, whose body is true in s.

that are causally relevant to X, as the *conformant width* of X, and set the width of P as the maximum conformant width over the variables X that appear in action preconditions or goals, Proposition 9 simply says that belief tracking for a *non-deterministic* conformant problem is exponential in the problem width. This width notion, however, is not exactly equivalent to the notion of Palacios and Geffner when used in the deterministic setting as it is defined over *variables* rather than *literals*. We will say more about this distinction below. In general, however, the two accounts yield similar widths over most deterministic benchmarks.

In the contingent setting, there are variables whose uncertainty may affect a variable Z but which are *not* causally relevant to Z. The situation is similar to the one arising in Bayesian networks (Pearl, 1988), where relevance flows both *causally*, in the direction of the arrows, and *evidentially*, from the *observations* against the direction of the arrows.

Definition 10. *X is* evidentially relevant *to Y in P if X is an observable variable and Y is causally relevant to X.*

The notion of relevance captures the transitive closure of the (directional) causal and evidential relations:

Definition 11. *X is* relevant *to Y if X is causally or evidentially relevant to Y, or X is relevant to a variable Z that is relevant to Y.*

Thus, a variable $X = W_1$ is relevant to a variable $Y = W_n$ iff there is a chain of variables W_i, $1 \leq i \leq n-1$, such each variable W_i is causally or evidentially relevant to the next variable W_{i+1} in the chain. For example, if X is causally relevant to Y and Z, and Y is an observable variable, then Y will be relevant to Z as Y is evidentially relevant to X and X is causally relevant to Z.

Like in Bayesian networks, the relevance relations can be understood graph-theoretically. Thus, if the directed edge $Z \rightarrow Y$ stands for Z being an immediate cause of Y, then X is causally relevant to X' when there is a directed path from X to X', and X is evidentially relevant to X' when X is an observable variable, and there is a directed path from X' to X. In terms of Bayesian networks, the relevance relation takes the transitive closure of the causal and evidential relationships, and encodes 'potential dependency' given what may be observed, using the information that certain variables will not be observed (are not observable). Unlike Bayesian networks, this means however that the relevance relation is not symmetric. Namely, a cause X of Y is relevant to Y, but Y is not automatically relevant to X if it is not causally relevant to an observable variable Z, which may be Y itself. The *context* of a variable is the set of variables in the problem that are relevant to X:

Definition 12. *The context of variable X, $Ctx(X)$, denotes the set of* state variables *in the problem that are relevant to X.*

The width of a variable is defined as the number of state variables in its context that are not determined:

Definition 13. *The* width *of a variable X, $w(X)$, is $|Ctx(X) \cap V_U|$, where $V_U = V \setminus V_K$ and V_K is the set of state variables that are determined.*

The width of a problem is then:

Definition 14. *The width $w(P)$ of a conformant or contingent problem P, whether deterministic or not, is $\max_X w(X)$ where X ranges over the variables that appear in a goal or action precondition in P.*

The relation between width and complexity can be expressed as:

Theorem 15. *Belief tracking in P is exponential in $w(P)$.*

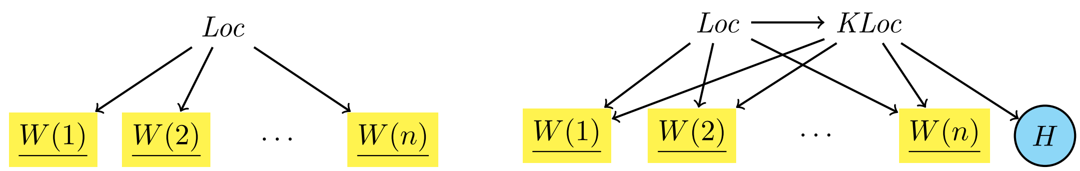
The proof for this theorem follows from the results below where an algorithm that achieves this complexity bound is presented. The significance of the theorem is that *belief tracking over planning domains with width bounded by a constant becomes polynomial in the number of problem variables*. We will see examples of this below. This complexity bound is similar to the ones obtained for *deterministic* conformant and contingent problems (Palacios & Geffner, 2009; Albore et al., 2009). The main difference is that the new account applies to *non-deterministic* problems as well. The new account is simpler and more general, but as we will see, it is also slightly less tight on some deterministic domains.

6. Examples

We illustrate the definitions above with some benchmark domains, starting with DET-Ring (Cimatti, Roveri, & Bertoli, 2004). In this domain, there is a ring of n rooms and an agent that can move forward or backward along the ring. Each room has a window which can be opened, closed, or locked when closed. Initially, the status of the windows is not known and the agent does not know his initial location. In this domain the agent has no means for obtaining information about the status of the windows or its position, and its goal is to have all windows locked. A plan for this *deterministic conformant* problem is to repeat n times the actions $(close, lock, fwd)$, skipping the last fwd action. Alternatively, the action fwd can be replaced by the action bwd throughout the plan. The state variables for the problem encode the agent location $Loc \in \{1, \ldots, n\}$, and the status of each window, $W(i) \in \{open, closed, locked\}$, $i = 1, \ldots, n$. The location variable Loc is (causally) relevant to each window variable $W(i)$, but no window variable $W(i)$ is relevant to Loc or $W(k)$ for $k \neq i$, as $W(i)$ is not causally relevant to an observable variable. None of the variables is determined and the largest contexts are for the window variables $W(i)$ that include two variables, $W(i)$ itself and Loc. As a result the width of the domain is 2, which is independent of the number of state variables $W(i)$ that grows with the number of rooms n. The causal graph of the problem, where a directed edge $X \rightarrow Y$ means that X is an immediate cause of Y is shown in Figure 1a.

NON-DET-Ring is a variation of the domain where the actions fwd and bwd of the agent have a non-deterministic effect on the status of all windows that are not locked, capturing the possibility of external events that can open or close unlocked windows. This non-determinism has no effect on the causal graph over the variables. As a result, the change has no effect on the contexts or domain width that remains bounded and equal to 2 for any number of rooms n.

The last version of the domain considered by Cimatti et al. is NON-DET-Ring-Key, where a key is required to lock the windows. The initial position of the key is not known,



(a) DET-Ring (b) CONT-NON-DET-Ring-Key

Figure 1: Causal graphs for the problems DET-Ring (left) and CONT-NON-DET-Ring-Key (right). In the latter, the variable H is observable and tells us whether the key is being held or not. An arc $X \rightarrow Y$ denotes that X is an immediate cause of Y. In these graphs, variables in preconditions or goals are underlined and yellow colored, while observable variables are enclosed in a blue circle.

yet if the agent tries to collect the key from a room and the key is there, the agent will have the key. A conformant plan for this problem is to repeat the actions *pick* and fwd, n times, skipping the last fwd action, following then the plan for DET-Ring. In NON-DET-Ring-Key, there is an additional state variable, $KLoc \in \{1, \ldots, n, hand\}$, that represents the key location. The agent location Loc is relevant to $KLoc$ which is relevant to each window variable $W(i)$. As a result, both the size of the contexts $Ctx(W(i))$ and the problem width increase by 1. The width however remains bounded with a value of 3 independently of the number of rooms n.[5]

In the presence of partial observability, the analysis is similar but it is necessary to consider the relevance relationships that arise due to the presence of observable variables. For example, one can express that the agent can always observe whether it is holding the key or not, by having a boolean observable variable H with (deterministic) observation model $W_a(H = true)$ given by $KLoc = hand$, for all actions a. The only new relevance relation among state variables that arises from adding this observable variable is between Loc and $KLoc$, as both are causally relevant to H. Before, Loc was relevant to $KLoc$ but not the other way around. Yet this does not affect the domain width that remains 3 for any n. The causal graph of the resulting domain is shown in Figure 1b.

7. Factored Belief Tracking

Belief tracking over a problem P is exponential in the width $w(P)$ of P. The algorithm that achieves this bound exploits the relevance relations encoded in the variable contexts for decomposing beliefs. In particular, if no variable is relevant to any other variable, the problem width is 1, and beliefs over each variable can be maintained separately. The belief decomposition is obtained from projecting the problem P into smaller problems P_S where S is a set of state variables in P. Semantically, the projected problems P_S capture the dynamics of the problem P when expressed over a subset S of state variables. Syntactically, the projected problems P_S are defined by means of the logical notion of projection. The

5. The problem can also be encoded by making 'holding key' a *precondition* rather than a *condition* for locking the windows. In such an encoding, the variable $KLoc$ is no longer relevant to the window variables $W(i)$ according to the definitions, as then $KLoc = hand$ must be known with certainty, and hence uncertainty about the windows variables $W(i)$ is not affected by uncertainty about $KLoc$. The result is that in such an encoding, the domain width reduces to 2.

logical projection of a formula F over a subset S of its variables refers to the formula F' defined over the variables in S, such that the valuations that satisfy F' are exactly those that can be extended into valuations that satisfy F (Darwiche & Marquis, 2002). Likewise, the projection of a conditional effect $C \rightarrow E^1|\cdots|E^n$ is the conditional effect $C_S \rightarrow E^1_S|\cdots|E^n_S$ where the body C and the effects E^i are replaced by their logical projections C_S and E^i_S respectively.

Definition 16. *The projection of problem $P = \langle V, I, A, G, V', W\rangle$ over a set of variables $S \subseteq V$ is the problem $P_S = \langle V_S, I_S, A_S, G_S, V'_S, W_S\rangle$ where V_S is S, I_S and G_S are the initial and goal formulas I and G logically projected on the variables in S, A_S is A but with the preconditions and conditional effects projected over S, V'_S is V', and W_S is the set of formulas $W_a(Y = y)$ in W logically projected on the variables in S.*

The notion of a projected planning problem has been used before in the setting of classical planning for introducing a class of admissible heuristics known as pattern databases (Edelkamp, 2001). Here we use it in the richer contingent setting for decomposing the belief tracking problem over P into the belief tracking problem over smaller problems P_S obtained from P by projecting away some of the state variables in P.

Before defining the target subproblems P_S of the decomposition, notice that variables Y that are both state and observable variables in P but are not in S, will belong to V'_S but not to V_S, meaning that they will be just observable variables in the projected problem P_S. Moreover, the formulas for such variables Y in W_S will become $W_a(Y = y) = true$ for all $y \in D_Y$, meaning that in the problem P_S, the observations $Y = y$ will be possible for any y, regardless of the state and last action done. Such observations will thus be completely irrelevant in P_S and will have no effect. In any case, P and P_S share the same set of actions and the same set of observations even if some of the actions and observations in P_S may be defined over a smaller set of state variables.

The target subproblems P_S are defined in terms of the set of state variables that are relevant to precondition and goal variables. Recall that we assume that each observable variable in the problem is relevant to some action precondition or goal, as else the variable could be safely removed.

Definition 17. *The projection of a problem P over a variable X, denoted as P_X, is the projection P_S of P over the set of variables $S = Ctx(X)$, where $Ctx(X)$ is the context for X in P; i.e., the set of state variables in P that are relevant to X.*

Two basic properties of the projected problems P_X are:

Proposition 18. *If variable X appears in a goal or precondition, then the number of state variables in P_X that are not determined is bounded by $w(P)$.*

Proposition 19. *If an execution $\langle a_0, o_0, a_1, o_1, \ldots\rangle$ is possible in P, then it is also possible in P_X for any state variable X in P.*

If b is the belief that results after an execution in P, we will call b_X the belief that results after the same execution in the projected problem P_X. The *completeness* of the decomposition of the global belief b over P is expressed in terms of the local beliefs b_X over the subproblems P_X. We treat the beliefs b and b_X as relations in a database where the state variables in these beliefs are the 'columns' and the possible combination of values (states and local states) are the 'rows'. The *projection* $\Pi_Y\, b$, for a set of variables Y thus represents the combination of values of the variables in Y that are possible in b, while the join $b_X \bowtie b_Y$ represents the combination of values x and y over the sets of variables in the two beliefs b_X and b_Y such that x and y coincide over the variables that are in both X and Y. For example, if b contains the valuations (states) $X = 1, Y = 1$ and $X = 2, Y = 2$, the projection $\Pi_{\{X\}} b$ will contain the valuations $X = 1$ and $X = 2$. Likewise, if b' contains $Y = 1, Z = 1$ and $Y = 1, Z = 2$, the join $b \bowtie b'$ will contain $X = 1, Y = 1, Z = 1$ and $X = 1, Y = 1, Z = 2$.

Theorem 20. *For a state variable X, let b and b_X be the beliefs that result from an execution that is possible over both P and P_X. Then,*

$$\Pi_X\, b_X = \Pi_X\, b\,. \tag{3}$$

Equation 3 states that a literal $X = x$ is possible in the true global belief b iff it is possible in the belief b_X that results from the same execution in the projected problem P_X. This is exactly the type of completeness that is needed in planning for any variable X involved in an action precondition or goal. The stronger form of completeness over *all formulas*, that can be expressed as

$$\bowtie_X\, b_X = b\,, \tag{4}$$

where '$\bowtie$' stands for the join operation and X ranges over all precondition and goal variables in the problem, is not needed, and it is actually *not* necessarily true, even when all state variables appear in some context $Ctx(X)$. For example, if the value of the boolean variable Z is initially unknown, and variables X and Y are initially false, an action a with conditional effects $Z \rightarrow X \wedge Y$ and $\neg Z \rightarrow Z$ results in a belief b with two states, corresponding to the terms $Z \wedge X \wedge Y$ and $Z \wedge \neg X \wedge \neg Y$. If X and Y are precondition or goal variables such that they are not relevant to each other, the projected problem P_X will contain the variables X and Z, and the projected problem P_Y will contain the variables Y and Z. The belief b_X resulting from the execution of the action a in P_X will include then the local states corresponding to the terms $Z \wedge X$ and $Z \wedge \neg X$, while the belief b_Y over P_Y will include the local states corresponding to the terms $Z \wedge Y$ and $Z \wedge \neg Y$. Clearly, the projection of b and b_X (b_Y) over the variable X (Y) coincide as dictated by (3), but the join of the two local beliefs b_X and b_Y does not yield the global belief b as would correspond to (4); indeed, a formula like $X \wedge \neg Y$ is false in the latter but not in the former. From (3), we can prove inductively on the size of any execution that:

Theorem 21. *1) An execution is possible in P iff it is possible over each of the subproblems P_X for X being a precondition or goal variable in P. 2) For an execution τ and precondition or goal variable X, $X = x$ (resp. $X \neq x$) is true in b iff $X = x$ (resp. $X \neq x$) is true in b_X, where b and b_X are the beliefs that result of executing τ in P and P_X respectively.*

Since plain belief tracking over each projected problem P_X is exponential in the size of P_X, which is bounded by $w(P)$ once the determined variables are excluded, it follows that:

Theorem 22. Flat belief tracking *over each of the projected problems P_X for X being a precondition or goal variable in P, provides a sound and complete* factored algorithm *for belief tracking over P that is time and space exponential in the width of P.*

We call this algorithm, *factored belief tracking*. In order to check whether a precondition or goal literal $X = x$ is true after an execution, factored belief tracking checks whether $X = x$ is true in the belief b_X that results from the execution over the subproblem P_X. An execution is not possible if an action precondition $X = x$ is not true in b_X or if it results in an empty belief over some subproblem. Theorem 22 thus says that factored belief tracking is a sound and complete algorithm for BTP with time and space complexity exponential in the problem width. Indeed, since every observable variable Y is relevant to some precondition or goal variable X by assumption, then every direct cause Z of Y is relevant to X because Y is evidentially relevant to X. Thus, any formula $W_a(Y = y)$ can be evaluated in b_X to determine whether the observation $Y = y$ is necessary, possible or impossible after applying the action a. Thus, factored belief tracking also solves the generalized BTP problem.

As an illustration of Theorem 22, let us go back to the DET-Ring problems P whose structure was analyzed before. The theorem implies that in order to check whether a given possible execution achieves the goal in P, it is sufficient to check whether each goal literal $W(i) = \mathit{locked}$, for $1 \leq i \leq n$, is achieved by the execution over the subproblem $P_{W(i)}$. Thus, factored belief tracking over P can be done in $O(n^2)$ time since there are n subproblems $P_{W(i)}$, each one involving 2 variables: $W(i)$ with a constant-size domain and *Loc* with a domain of size n.

The exact same situation arises in the non-deterministic conformant problem NON-DET Ring whose causal graph is the same as the one for DET-Ring. On the other hand, for NON-DET-Ring-Key, all the subproblems must keep track of the *KLoc* variable encoding the key location, and thus a belief update operation requires $O(n^3)$ time, which is still much better than flat belief tracking over P which requires time exponential in n. The same complexity results applies when the problem is no longer conformant and the agent can observe whether it is holding the key or not.

Some experimental figures for these domains are shown in Table 1, where factored belief tracking was used in combination with simple heuristics. The experiments were run on a Xeon 'Woodcrest' 5140 CPU running at 2.33 GHz and with 8 GB of RAM. The planner KACMBP by Cimatti et al. uses an OBDD-based belief representation and cardinality heuristics, and can solve problems with up to $n = 20$ rooms, producing plans with 206 steps in slightly more than 1,000 seconds in NON-DET-Ring-Key. Conformant planners such as T_0 (Palacios & Geffner, 2009) cannot be used as the problem is non-deterministic. Tables 1a and 1b show the scalability of the factored belief tracking algorithm in the context of a greedy best-first search with a heuristic $h(b)$, similar to the one used by Albore, Ramirez, and Geffner (2011), where $h(b) = \sum_{i=1}^{n} h(b_i)$, with b_i being the belief factor in the projected problem for the goal variable $W(i)$ representing the status of the ith window, and $h(b_i)$ representing the fraction of states in b_i where the goal $W(i) = \mathit{locked}$ is false. As displayed in the tables, the resulting planner scales up polynomially, and for NON-DET-Ring-Key with 100 rooms, produces a plan with $1,111$ actions in 783.1 seconds. For the contingent version of the problem in which the agent detects when the key is in the room, CONT-DET-Ring-Key, a policy greedy in the cardinality heuristic $h(b) = \max_{i=1}^{n} |b_i|$ is used instead, with

n	steps	exp.	time
10	68	355	< 0.1
20	138	705	0.1
30	208	1,055	0.9
40	277	1,400	3.1
50	345	1,740	8.3
60	415	2,090	18.6
70	476	2,395	34.5
80	545	2,740	62.8
90	610	3,065	106.4
100	679	3,410	171.0

(a) DET-Ring-Key

n	steps	exp.	time
10	118	770	< 0.1
20	198	1,220	0.8
30	278	1,670	4.2
40	488	3,210	15.2
50	438	2,570	34.4
60	468	2,660	52.2
70	543	3,080	100.6
80	616	3,480	172.9
90	682	3,880	285.6
100	1,111	7,220	783.1

(b) NON-DET-Ring-Key

n	avg. steps	avg. time
10	326.8 ± 4.3	0.0
20	$1,036.0 \pm 13.5$	0.1
30	$2,068.0 \pm 26.5$	0.5
40	$3,462.9 \pm 47.2$	1.8
50	$5,130.7 \pm 71.0$	4.4
60	$7,070.9 \pm 100.9$	9.3
70	$9,334.1 \pm 127.6$	17.5
80	$11,724.0 \pm 162.2$	30.6
90	$14,617.4 \pm 204.6$	50.0
100	$17,891.2 \pm 252.3$	79.0

(c) CONT-DET-Ring-Key

Table 1: Results for conformant and contingent Ring problems obtained by combining factored belief tracking with simple heuristics. Each data point in panel (c) for the contingent problem is the average (and sample standard deviation) over 1,000 random instances. Times are in seconds. The column "exp." contains number of expansions.

ties broken randomly, where b_i is the belief factor for the goal variable $W(i)$. As it can be seen in Table 1c, the resulting planner runs in polynomial time and can solve problems with up to 100 rooms. Thus, while the heuristic and policy are weak, and long executions result, belief tracking in this problem is efficient and scales up well.

8. Causal Belief Tracking

Factored belief tracking is exponential in the problem width. In many problems, however, the width may be too high for the method to be usable in practice. As an illustration, consider a problem P with state variables $X_1, \ldots, X_{n+1}$, and observable variables $O_1, \ldots, O_n$ such that O_i is true iff $X_i = X_{i+1}$. The sensors are thus $W_a(O_i = true) = (X_i = X_{i+1})$ and $W_a(O_i = false) = (X_i \neq X_{i+1})$ for all actions a and $1 \leq i \leq n$. Let us also assume that the actions in the problem may affect each of the X_i variables but do not introduce causal relations among them, and that all the state variables appear in preconditions or goals. The causal graph of the problem is shown in Figure 2. Its width is $n+1$ as all the state variables interact. Indeed, each variable X_i is relevant to each variable X_k, with the relevance flowing from X_i to X_{i+1}, and vice versa, as both variables are causally relevant to the observable variable O_i which is evidentially relevant to both. The result is that the problem P and the projected problems P_{X_i} all coincide and denote the same problem, as the contexts for each of the state variables include all state variables.

We now focus on a different decomposition for belief tracking that maps a problem P into smaller subproblems P^c_X whose size is bounded by the number of state variables that are all *causally relevant* to a given precondition, goal, or observation variable. This new width measure will be called the *causal width* of the problem. The problem shown in Figure 2 has width $n+1$ but causal width 2. We will then explore belief tracking algorithms that are exponential in the problem causal width and analyze the conditions under which they are

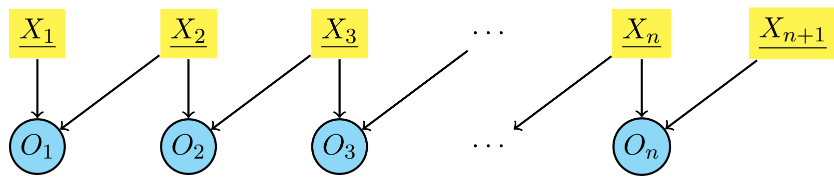


Figure 2: Causal graph for the 2-layer network example with state variables $X_1, \ldots, X_{n+1}$ and observable variables $O_1, \ldots, O_{n+1}$. The immediate causes of each observable O_i are the variables X_i and X_{i+1}. Precondition or goal variables appear as underlined and in a yellow box, while observable variables appear within a blue circle. Since all X_i variables are relevant to each other, the *width* of the problem is $n+1$. On the other hand, since at most two variables are *causally relevant* to a precondition, goal, or observable variable, the *causal width* of the problem is 2.

complete. For this, we first generalize and make explicit the *decomposition* underlying the factored belief tracking algorithm:

Definition 23. *A* decomposition *of a problem P is a pair $D = \langle T, B\rangle$, where T is a set of variables X appearing in P, called the* target variables *of the decomposition, and B is the collection of* beams $B(X)$ *associated with each such target variable which are made up of* state variables *from P.*

A decomposition $D = \langle T, B\rangle$ maps P into a set of *subproblems* P^D_X, one for each variable X in T, that corresponds to the *projections* of P over the state variables in the beam $B(X)$. The decomposition that underlies factored belief tracking is:

Definition 24. *The* factored decomposition $F = \langle T_F, B_F\rangle$ *of P is the decomposition with target variables T_F given by the state variables X appearing in action preconditions or goals, and beams $B_F(X)$ given by the state variables Y that are* relevant *to X.*

Factored belief tracking is flat belief tracking applied to the subproblems determined by the *factored decomposition*. The algorithms that we introduce next are based on a different decomposition:

Definition 25. *The* causal decomposition $C = \langle T_C, B_C\rangle$ *of P is the decomposition with target variables T_C given by the observable variables and the state variables appearing in action precondition or goals, with beams $B_C(X)$ given by the state variables Y that are* causally relevant *to X.*

The causal decomposition determines a larger number of subproblems, as subproblems are also generated for the observable variables, but these subproblems have smaller beams $B_C(X)$, as they only contain the state variables that are *causally relevant* to X as opposed to the variables that are *relevant* to X. The *causal width* of a problem is given by the size of the largest beam in the causal decomposition, discounting the variables that are determined in the problem:

Definition 26. *The* causal width *of a variable X in a problem P, $w_c(X)$, is the number of state variables that are* causally *relevant to X and are not determined. The* causal width *of P is $\max_X w_c(X)$, where X ranges over the target variables in the causal decomposition of P.*

The first and simplest belief tracking algorithm defined over the causal decomposition is what we call *Decoupled Causal Belief Tracking*, which runs in time and space that are exponential in the problem causal width:

Definition 27. *Decoupled causal belief tracking (Decoupled CBT) is* flat belief tracking *applied independently to each of the problems P_X^C determined by the causal decomposition $C = \langle T_C, B_C \rangle$ of P. The subproblem P_X^C is the problem P projected on the variables in $B_C(X)$ for $X \in T_C$; i.e., $P_X^C = P_{B_C(X)}$.*

Since causal width is never greater than width and is often much smaller, Decoupled CBT runs much faster than factored belief tracking in general. This, however, comes at a price that we express using the expression $\Pi_S b$ for denoting the projection of (the states in the) belief b over the variables in S.

Theorem 28. *Decoupled CBT runs in time and space that are exponential in $w_c(P)$, and it is sound but not complete. That is, for any target variable X in the causal decomposition, if b and b_X are the beliefs resulting from an execution on P and P_X^C respectively, then $b_X \supseteq \Pi_{B_C(X)} b$ is necessarily true, but $b_X \subseteq \Pi_{B_C(X)} b$ is not.*

One reason for the incompleteness is that the beliefs b_X associated with different target variables X are assumed to be independent in Decoupled CBT while this may not be true. Indeed, the causal decomposition of a problem may give rise to a beam $B_C(Y)$ involving variable X, and a second beam $B_C(Z)$ involving the same variable X and another variable X'. If variable Y is then observed, $X = x$ may become false, which from a further observation on Z may lead to $X' = x'$ becoming false as well. Yet, in Decoupled CBT, this inference cannot be captured as there is no information flow across beams. In the factored decomposition a situation like this cannot happen as variable X' will be relevant to variable X and hence beams that contain X will necessarily contain X' (X' is relevant to X because it's causally relevant to Z which is evidentially relevant to X).

In the causal decomposition, beams are kept small by not closing them with the relevance relation, but as a result, the beliefs over such beams are no longer independent. However, regarding the beliefs as tables or relations, a *consistency* relation among the local beliefs in the causal decomposition can be enforced by means of the join operation. The resulting algorithm is *Coupled Causal Belief Tracking*, abbreviated simply as *Causal Belief Tracking*:

Definition 29. *Causal Belief Tracking (CBT) is the belief tracking algorithm that operates on the causal decomposition $C = \langle T_C, B_C \rangle$ by setting the beliefs b_X^0 at time 0 for each beam $B_C(X)$ to the projection on $B_C(X)$ of the initial belief, $X \in T_C$, and the successive beliefs b_X^{i+1} as:*

$$b_X^{i+1} = \Pi_{B_C(X)} \bowtie \{(b_Y^i)_a^o \,:\, Y \in T_C \text{ and } Y \text{ is relevant to } X\} \tag{5}$$

where $a = a_i$ and $o = o_i$ are the action and observation at time i in the execution, and $(b_Y^i)_a^o$ is b_a^o from Eqs. 1-2 with $b = b_Y^i$.

In CBT, the beliefs are not tracked independently over each of the subproblems P_X^C of the causal decomposition; rather, the beliefs are first progressed and filtered *independently*, but are then merged and projected back onto the beams, making them consistent with each other. The progression and filtering of the local beliefs in the causal decomposition is performed in time and space exponential in the problem *causal width*, but the full consistency operation captured by the join-project operation in (5) requires time that in the worst case is exponential in the problem *width*:

Theorem 30. *CBT is* space *exponential in the* causal width *of the problem, and* time *exponential in its* width*.*

CBT is sound but incomplete. However, the range of problem for which CBT is complete, unlike Decoupled CBT, is large and meaningful enough, and it includes for example three of the domains to be considered in the experiments below: Battleship, Minesweeper and Wumpus. We express the completeness conditions for CBT by introducing the notion of *memory variables*:

Definition 31. *A* state variable X *is a* memory variable *in problem P when the value X^k of the variable X at time point k in an execution is* determined uniquely *from an observation of the value X^i of X at any time point i, $i \geq k$, the actions in the execution, and the initial belief state of the problem.*

For example, *static variables* are memory variables as they do not change and thus knowing their value at any time point determines their value at any other point. *Determined* variables (Section 4) are also memory variables since the value X^k of such variables is determined by the initial belief and the actions done up to time k. Likewise, variables in *permutation domains* where actions permute the values of the variables (Amir & Russell, 2003), are also memory variables. These are three sufficient conditions for a state variable to be a memory variable that are all easy to check. A problem is said then to be *causally decomposable* when the following condition holds:

Definition 32. *A problem P is* causally decomposable *when for every pair of beams $B_C(X)$ and $B_C(X')$ in the causal decomposition of P with a non-empty intersection, where X' is an observation variable, either 1) the variables in the intersection are all memory variables, or 2) there is a variable W in the causal decomposition that is relevant to X or X' and whose causal beam $B_C(W)$ contains both $B_C(X)$ and $B_C(X')$.*

If the problem is causally decomposable, the filtering implemented by the updates in CBT using Equation 5 suffices for completeness:

Theorem 33. *Causal belief tracking is always sound and it is complete for causally decomposable problems.*

The importance of this result is that there are many meaningful domains whose problem instances are causally decomposable; in particular, domains where all variables that appear in two different beams are static (this include Minesweeper), domains where all variables that appear in two different beams are either static or determined (this includes Wumpus, where the non-static variable for the agent location is determined), domains where the hidden non-static state variables only appear in one beam (this includes Battleship where the hidden non-static variables do not appear in intersection of beams), and other cases as well. In Sect. 11.4, we present a variation of Wumpus in which the monster moves non-deterministically in the grid and that it is also an instance of a causally-decomposable problem.

9. Approximation: Beam Tracking

The causal belief tracking algorithm shows that it is possible to track beliefs for planning in a sound and complete manner for a large and meaningful class of problems, while considering the beliefs over subproblems that are smaller than those in the factored decomposition. The algorithm, however, while space exponential in the causal width of the problem, it is time exponential in the problem width. This is because of the global consistency operation enforced by (5). *Beam tracking* is the final belief tracking algorithm that we consider: it replaces this global consistency operation by a local consistency operation that can be performed in polynomial time. Beam tracking is thus an approximation of causal belief tracking which is aimed at being efficient and effective rather than complete.

Definition 34. Beam tracking *is the belief tracking algorithm that operates on the causal decomposition* $C = \langle T_C, B_C \rangle$*, setting the beliefs* b_X^0 *at time 0 to the projection of the initial belief over the beam for* $X \in T_C$*, and setting the successive beliefs* b_X^{i+1} *in two steps. First, they are set to the progressed and filtered belief* b_a^o *for* $b = b_X^i$*,* $a = a_i$ *and* $o = o_i$*, where* a_i *and* o_i *are the action and observation at time i in the execution. Then, a* local *form of consistency is enforced upon these beliefs by means of the following updates until a fixed point is reached:*

$$b_X^{i+1} = \Pi_{B_C(X)}(b_X^{i+1} \bowtie b_Y^{i+1}) \tag{6}$$

where Y *refers to any other target variable in the causal decomposition such that* $B_C(Y) \cap B_C(X)$ *is non-empty.*

The filtering represented by the iterative update in Eq. 6 defines a form of *relational arc consistency* (Dechter & Beek, 1997) where equality constraints among beams sharing common variables is enforced in polynomial time and space in the size of the beams. Beam tracking remains *sound* but is *not complete*. In causally decomposable problems, however, the incompleteness is the sole result of replacing global by local consistency.

10. Extensions, Modeling, and Width

Before testing the beam tracking algorithm empirically, we present two simple extensions to the language of contingent planning that are useful for modeling, and briefly discuss modeling choices that affect the causal width of a problem. The first extension allows the use of *defined variables* in preconditions and goals; the second extension allows the use of *state constraints* for restricting the possible value combination of subsets of variables.

10.1 Defined Variables

A variable Z with domain D_Z can be defined as a function of a subset of state variables in the problem, or as a function of the *belief* over such variables. For example, a boolean variable Z can be defined as true when two variables X and Y are equal, or when a third variable W is known to be true. Defined variables Z that are a function of a set S_Z of state variables or a function of the belief over such variables, can then be handled in action preconditions and goals by introducing a beam in the decomposition that includes the variables in S_Z along with the variables that are relevant or causally relevant to them, according to whether the decomposition is factored or causal. The width and causal width of the problem follow then, as before, as the size of the largest beam in the factored and causal decompositions with the determined variables excluded.

10.2 State Constraints

State constraints are used to restrict the value combinations of given subsets of state variables. The game of Battleship, for example, can be modeled with state variables associated with each of the cells in a grid for representing whether the cell is part of a ship, the size of the ship to which the cell belongs (if any), the relative position of the cell within the ship to which the cell belongs (if any), and whether such a ship is placed vertically or horizontally. These state variables, however, are not independent, and indeed, if a ship of size 10 is horizontally placed at cell $(0, 0)$, the cells $(0, i)$, for $i \in \{0, 1, \ldots, 9\}$ must belong to (the same) ship.

Formally, a state constraint represented by a formula C over the state variables can be encoded by means of a dummy observable variable Y that is *always observed to be true*, and that can be observed to be true only in states where C holds; i.e., with model $W_a(Y = true) = C$ for every action a. For the implementation, however, it pays off to treat such constraints C as relations (the set of valuations that satisfy C), and to include them in all the 'joins' over the beliefs that include the variables in C. In causal belief tracking this has no effect on the completeness or complexity of the algorithm, but in beam tracking, changing the update in (6) to

$$b_X^{i+1} = \Pi_{B_C(X)}(b_X^{i+1} \bowtie b_Y^{i+1} \bowtie C_1 \bowtie \cdots \bowtie C_n) \tag{7}$$

where $C_1, \ldots, C_n$ are the state constraints whose variables are included in $B_C(X) \cup B_C(Y)$, makes local consistency stronger with no effect on the complexity of the algorithm. Moreover, when there is one such pair of beams for each state constraint, the state constraints can increase the *causal width* of the problem by a constant factor of 2 at most, yet the *effective causal width* of the problem does not change, as the beams associated with the dummy observables introduced for such constraints are redundant and can then be ignored. In this later case, when using beam tracking, the constraints C_i do not need to be stored in extensional form as relations but can be handled intentionally as boolean functions that test whether an assignment in the join of two beams satisfies the constraint.

10.3 Modeling and Width

The complexity of the belief tracking algorithms is a function of the width or causal width of the problem, which in turns depends on the way the problem is encoded. Often small changes in the encoding can have a drastic effect on the resulting widths. For example, in the Wumpus problem (Russell & Norvig, 2009), it is natural to define the conditions under which the stench signal can be received by setting its observation model to:

$$W_a(stench = true) = \bigvee_c \big[(pos = c) \land \bigvee_{c'} wump_{c'}\big]$$

where *pos* encodes the agent position, c ranges over the possible cells, c' ranges over the cells that are adjacent to c, and $wump_{c'}$ denotes the presence of a wumpus at c'. This encoding, however, results in a beam for the observable variable *stench* that includes all the $wump_c$

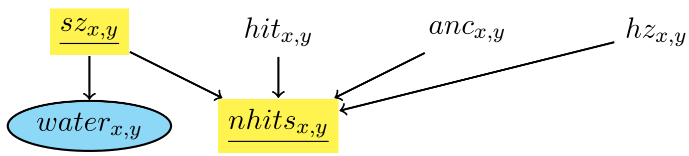


Figure 3: Causal graph fragment for Battleship. Circled variables are observable while the others are state variables. The problem has one type of variable for each cell (x, y) on the grid. Causal width for the problem is 5.

variables, and hence whose size grows with the grid size. A better alternative that results in beams of *bounded causal width* is to exploit the fact that the position of the agent *pos* is determined. Taking advantage of this, the observable variable *stench* can be replaced by observable variables $stench_c$, one for each cell in the grid, with sensors characterized by the model:

$$W_a(stench_c = true) = (pos = c) \wedge \bigvee_{c'} wump_{c'} \,.$$

The beams for the $stench_c$ variables contain at most four $wump_{c'}$ variables, one for each cell c' adjacent to c. In this way, the causal width of the Wumpus problem becomes bounded and independent of the grid size, and of the number of wumpus and pits (see below). The idea can be generalized and automated. Any observation model of the form $W_a(Z = z) = \bigvee_x \varphi(x) \wedge \psi(x)$, where $\varphi(x)$ is a formula constructed from determined variables, can be replaced by observation models $W_a(Z_x = z) = \varphi(x) \wedge \psi(x)$ by expanding the number of observable variables. Likewise, multiple observation models $W_{a_i}(Z = z) = \varphi_i$ for one observable variable Z and different actions $\{a_i\}_{i \in R}$ can be conveniently replaced by observation models $W_{a_i}(Z_i = z) = \varphi_i$, $i \in R$ for different observable variables Z_i, when the different φ_i formulas involve different variables. These alternatives in the domain encoding can be the difference between bounded and unbounded causal width, and hence, on whether the complexity of beam tracking will grow polynomially or exponentially.

11. Experiments

We have tested beam tracking over large instances of Battleship, Minesweeper, and Wumpus, in combination with simple heuristics for action selection that make use of the computed beliefs. The width of these problems is not bounded, and hence, neither factored or causal belief tracking can be used except over small instances. On the other hand, all these domains have small and bounded causal widths in the encodings provided, and hence beam tracking runs efficiently in both time and space. Exact belief tracking in some of these domains is difficult (Kaye, 2000; Scott, Stege, & Rooij, 2011), and the sizes of the instances considered are much larger than those used in contingent planning. Moreover, some of these domains do not have full contingent solutions. We thus compare our on-line planner that relies on handcrafted heuristics with two reported solvers that rely on belief tracking algorithms tailored to the domains. We also consider a non-deterministic version of the Wumpus domain. The results have been obtained on a Xeon 'Woodcrest' 5140 CPU running at 2.33 GHz with 8GB of RAM.

11.1 Battleship

Battleship is a popular two-player guessing game. The standard version consists of four ships of length 2, 3, 4 and 5 units that are secretly placed on a 10-by-10 grid, with no ship adjacent or diagonally adjacent to another. The task is to sink the ships by firing torpedos at specific cells. For each fired torpedo, we are told whether the torpedo hits water or ship. A ship is sunk when all its cells are hit. The problem is encoded with 6 state variables per cell (x, y):[6] $hit_{x,y}$ tells if a torpedo has been fired at the cell, $sz_{x,y}$ tells the size of the ship occupying the cell (0 if no such a ship), $hz_{x,y}$ tells if the ship is placed horizontally or vertically (true if no such ship), $nhits_{x,y}$ tells the number of hits on the ship (0 if no such ship), and $anc_{x,y}$ tells the relative position of the ship on the cell (0 if no such ship). There is a single observable boolean variable *water* with a deterministic sensor model given by $W_{fire(x,y)}(water_{x,y} = true) = (sz_{x,y} = 0)$. The action model is more complex because firing a torpedo at (x, y) may cause a change in the variables associated to other cells (x', y'). Indeed, if d denotes the maximum size of a ship (5 in the standard game), then $fire(x, y)$ includes conditional effects for variables referring to cells (x', y') that are at a vertical or horizontal distance of at most d units. The goal of the problem is to achieve the equality $nhits_{x,y} = sz_{x,y}$ over the cells that may contain a ship. State constraints are used for constraining sets of state variables as described above. In this encoding, the causal beams never contain more than 5 variables, even though the problem width is not bounded and grows with the grid size. Figure 3 shows a fragment of the causal graph for Battleship.

Table 2 shows results for two policies: a *random* policy that fires at a non-fired cell at random, and a *greedy* policy that fires at the non-fired cell most likely to contain a ship. Approximations of these probabilities are obtained from the *beliefs* maintained by beam tracking.[7] The difference in performance between the two policies shows that the beliefs are very informative. Moreover, for the 10×10 game, the agent fires 40.0 ± 6.9 torpedos in average, matching quite closely the average results of Silver and Veness (2010) that are obtained with a combination of UCT (Kocsis & Szepesvári, 2006) for action selection, and a particle filter (Doucet et al., 2000) hand-tuned to the domain for belief tracking. Their approach, however, involves 65,000 simulation per action that result in the order of 2 seconds per game over 10×10 instances, while our greedy approach takes 0.0096 seconds per game.

11.2 Minesweeper

The objective in Minesweeper is to clear a rectangular minefield without detonating a mine. Each play either opens or flags a cell. In the first case, if the cell contains a mine, the game is terminated; otherwise an integer counting the number of mines surrounding the cell is revealed. An initial configuration for minesweeper consists of a $m \times n$ minefield with k randomly-placed mines. There are three standard difficulty levels for the game that are made up of 8×8, 16×16 and 16×30 boards with 10, 40 and 99 mines respectively.

6. This is a rich encoding that allows to accommodate the observation that a ship has been fully sunk. In the experiments, however, this observation is not used in order to compare with the results reported by Silver and Veness (2010).
7. Probabilities for events defined by the variables in a beam are obtained by the ratio of number of states in the beam that satisfy the event to the total number of states in the beam.

				avg. time per	
dim	policy	#ships	#torpedos	decision	game
10×10	greedy	4	40.0 ± 6.9	2.4E-4	9.6E-3
20×20	greedy	8	163.1 ± 32.1	6.6E-4	1.0E-1
30×30	greedy	12	389.4 ± 73.4	1.2E-3	4.9E-1
40×40	greedy	16	723.8 ± 129.2	2.1E-3	1.5
10×10	random	4	94.2 ± 5.9	5.7E-5	5.3E-3
20×20	random	8	387.1 ± 13.6	7.4E-5	2.8E-2
30×30	random	12	879.5 ± 22.3	8.5E-5	7.4E-2
40×40	random	16	$1{,}572.8 \pm 31.3$	9.5E-5	1.4E-1

Table 2: Results for Battleship. The table contains results for the greedy and random policies described in the text. For the 10×10 board, there are 4 ships of sizes 2, 3, 4 and 5. As the size of the board is increased with n, the number of ships of each size gets multiplied by n. Average and sample standard deviation for the number of torpedos required to sunk all ships, calculated over 10,000 random instances for each board, are shown. Average times are in seconds.

The problem is encoded with $3mn$ boolean state variables $mine_{x,y}$, $opened_{x,y}$ and $flagged_{x,y}$ that denote the presence/absence of a mine at cell (x, y) and whether the cell has been opened or flagged, and mn observable variables $obs_{x,y}$ with domain $D = \{0, \ldots, 9\}$. There are two type of actions $open(x, y)$ and $flag(x, y)$ where the first has no precondition and effect $\neg flagged_{x,y} \rightarrow opened_{x,y}$, while the second has precondition $\neg mine_{x,y}$ and effect $flagged_{x,y}$. The sensor model is given by formulas that specify the integer that the agent receives when opening a cell in terms of the status of the $mine_{x',y'}$ variables over the surrounding cells. These formulas are:

$$
\begin{aligned}
W_{open(x,y)}(obs_{x,y} = 9) &= mine_{x,y}\,, && \\
W_{open(x,y)}(obs_{x,y} = k) &= \neg mine_{x,y} \wedge \textstyle\bigvee_{t \in N(x,y,k)} t\,, && \text{for } 0 \le k < 9\,, \\
W_{open(x,y)}(obs_{x',y'} = k) &= true\,, && \text{for } (x', y') \neq (x, y) \text{ and } 0 \le k \le 9\,, \\
W_{flag(x,y)}(obs_{x',y'} = k) &= true\,, && \text{for each } (x', y') \text{ and } 0 \le k \le 9\,,
\end{aligned}
$$

where $N(x, y, k)$ are the terms over the 8 cell variables $mine_{x',y'}$ surrounding the cell (x, y) that make exactly k literals true. In the initial situation, the variables $opened_{x,y}$ and $flagged_{x,y}$ are false and $mine_{x,y}$ is unknown. The goal of the problem is to get the disjunction $flagged_{x,y} \vee opened_{x,y}$ for each cell (x, y) without triggering an explosion.

The beams that result from the factored decomposition contain all the $3mn$ state variables, making all beams identical and resulting in an *unbounded* width of $3mn$. The causal width, on the other hand, is 9 as the causal beams for $opened_{x,y}$ and $flagged_{x,y}$ are identical and contain just 3 variables, while the beams for $obs_{x,y}$ contain the 9 $mine_{x',y'}$ variables for the cells (x', y') that surround the cell (x, y) along with the variable $mine_{x,y}$. Figure 4 contains a fragment of the causal graph for Minesweeper.

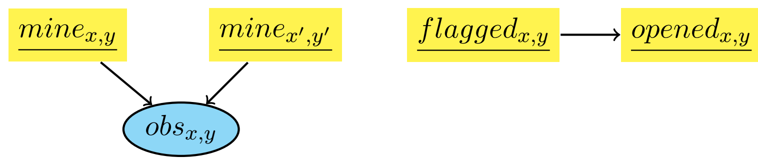


Figure 4: Sketch of the causal graph for Minesweeper. There are observable variables $obs_{x,y}$ and state variables $mine_{x,y}$, $flagged_{x,y}$ and $opened_{x,y}$ for each cell (x, y). The cell (x', y') represents one of the adjacent cells to (x, y). Since there are 8 such cells, the causal width of the problem is 9.

					avg. time per	
dim	#mines	density	%win	#guess	decision	game
8×8	10	15.6%	83.4	606	8.3E-3	0.21
16×16	40	15.6%	79.8	670	1.2E-2	1.42
16×30	99	20.6%	35.9	2,476	1.1E-2	2.86
32×64	320	15.6%	80.3	672	1.3E-2	2.89

Table 3: Results for Minesweeper. The table contains results for the three standard levels of the game plus a larger instance. Average results over 1,000 runs are shown. Average times are in seconds.

Table 3 shows results for the three standard levels of the game and for a much larger instance. As in Battleship, the greedy policy used for action selection makes use of the beliefs computed by beam tracking, flagging or opening a cell when certain about its content, else selecting the cell with the lowest probability of containing a mine and opening it, with the probabilities approximated from the beliefs over the beams as indicated before. Despite the complexity of the game, NP-complete for checking consistency (Kaye, 2000) and coNP-complete for inference (Scott et al., 2011), beam tracking scales well and solves difficult games quickly. Moreover, the results shown in the table are competitive with those recently reported by Lin, Buffet, Lee, and Teytaud (2012), which are obtained with a combination of UCT for action selection, and a domain-specific CSP solver for tracking beliefs. The success ratios that they report are: $80.2 \pm 0.48\%$ for the 8×8 instances with 10 mines, $74.4 \pm 0.5\%$ for the 16×16 instances with 40 mines, and $38.7 \pm 1.8\%$ for the 16×30 instances with 99 mines. The authors do not report times.

11.3 Wumpus

The Wumpus game (Russell & Norvig, 2009) consists of a maze in which there is an agent that moves around looking for the gold while avoiding hidden pits and wumpus monsters. Initially, the agent does not know the positions of the gold, pits or wumpuses, but it senses glitter when at the same cell as the gold, and senses a stench or a breeze when at an adjacent cell to a wumpus or a pit respectively. An $m \times n$ instance is described with known state variables for the position and orientation of the agent, and hidden boolean variables for each cell that tell whether there is a pit, a wumpus, or nothing at the cell. One more

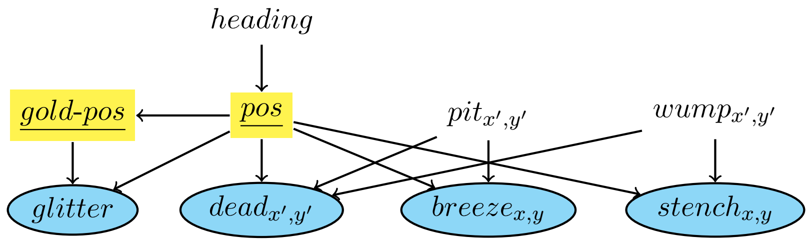


Figure 5: Fragment of the causal graph for Wumpus. There are observable variables $breeze_{x,y}$, $stench_{x,y}$ and $dead_{x,y}$, and state variables *heading*, *pos*, $pit_{x,y}$ and $wump_{x,y}$, for (x, y) ranging over the grid cells. Cells (x', y') stand for cells adjacent to (x, y). The causal width of the problem is 4 as there are 4 such cells, while the state variables *heading* and *pos* are determined.

hidden state variable stores the position of the gold. The observable variables are boolean: *glitter*, $breeze_{x,y}$, $stench_{x,y}$ and $dead_{x,y}$, with (x, y) ranging over the different cells. The actions are move forward, rotate right or left, and grab the gold. The causal width for the encoding is 4 while the problem width grows with m and n. Figure 5 shows a fragment of the causal graph for Wumpus. The size of the causal beams for the breeze and stench variables is bounded by 4 because each cell has at most 4 neighbors and the heading and position variables for the agent are determined.

Table 4 shows results for different grid sizes and number of pits and wumpus, for an agent that selects actions with a greedy policy based on a heuristic that returns the length of a minimum-length safe path to the nearest cell that may contain the gold. The beliefs computed by beam tracking are used to determine which cells are safe (known to contain no wumpus or pit) and may contain the gold. We are not aware of any other tested and scalable solver for Wumpus for making a comparison, with the exception of our own recent LW1 planner that has built on this work (Bonet & Geffner, 2014). The figures in the table show clearly that beam tracking computes beliefs effectively and efficiently in this domain. For instance, the 30×30 instances with 32 pits and 32 wumpus are solved successfully 89% of the time, in less than 4.4 seconds on average. Moreover, all the unsolved instances were actually shown to be unsolvable in the sense that the agent could not reach an unvisited cell in a safe manner. This was proved for each unsolved instance by calling a SAT solver on a propositional theory that encodes the game and the literals learned by the agent after the execution.

11.4 Non-Deterministic Moving Wumpus

In order to evaluate beam tracking in a more complex non-deterministic domain (the NON-DET-Ring-Key domain in Section 7 has small width), we designed a non-deterministic variant of the Wumpus domain. In Moving Wumpus there is just one wumpus in the grid but this wumpus moves around non-deterministically everytime that the agent moves. The grid still contains the hidden pits and the hidden gold, but in order to make the game safer for the agent, the wumpus sensor is enhanced to detect the position of the wumpus when at a (euclidean) distance less than 3 from the agent (else there is no safe strategy for escaping death in general).

					avg. time per	
dim	#pits/#wumpus	%density	#decisions	%win	decision	game
5×5	1 / 1	8.0	22,863	93.6	3.8E-4	8.7E-3
10×10	2 / 2	4.0	75,507	98.3	9.6E-4	7.2E-2
15×15	4 / 4	3.5	165,263	97.9	1.6E-3	2.6E-1
20×20	8 / 8	4.0	295,305	97.8	2.4E-3	7.2E-1
25×25	16 / 16	5.1	559,595	94.0	3.8E-3	2.1
30×30	32 / 32	7.1	937,674	89.0	4.7E-3	4.4
35×35	64 / 64	10.4	2,206,905	54.3	3.7E-3	8.2
40×40	128 / 128	16.0	4,471,168	7.3	2.8E-3	12.7
45×45	256 / 256	25.2	6,026,625	0.8	8.6E-3	51.8
50×50	512 / 512	40.9	7,492,503	0.1	1.3E-2	100.4

Table 4: Results for Wumpus. For each size, we performed 1,000 runs. The table shows the total number and density of pits and wumpus in the grid, the total number of decisions across all the runs, the percentage of runs in which the agent found the gold, and the average time in seconds per decision and game.

Moving Wumpus is causally decomposable and thus the incompleteness of beam tracking in this domain is only due to the replacement of the full consistency among beams done by CBT by the weaker but efficient (relational) arc consistency done by beam tracking. To see this, observe that the only variable that is not a memory variable is the position of the wumpus *WLoc*. However, there are only two beams in the causal decomposition that contain this variable: the beam for *WLoc* and the beam for the observable variable that tells the position of the wumpus, and the former beam is contained in the latter beam.[8]

Experimental results for beam tracking over this domain are presented in Table 5 for a policy obtained using the AOT lookahead algorithm based on AO* (Bonet & Geffner, 2012a) that builds a lookahead tree of depth 10 using 50 expansions, and a heuristic function that measures the distance between the agent position and the closest unvisited cell.

The algorithm was evaluated on different instances with grids $n \times n$ for $n = 4, 6, 8, \ldots, 20$, each with a number of pits equal to $(n-4)/2$. For each grid size, we performed 1,000 evaluations for different initial configurations where the wumpus, pits and gold are randomly placed. An instance of this game may turn unsolvable because the gold is isolated from the agent by pits, because the agent finds itself in a position where there is no safe movement, or because the agent exceeded the maximum number of actions (set to 3 times the number of cells in the grid).

8. Indeed, a more general version of this problem involves m wumpuses that move non-deterministically in the grid. This version is also causally decomposable as the beams for the positions of the wumpuses (one for each wumpus) are all contained in the beam for the observable variable. In such general case, the problem would have causal width equal to m.

					avg. time per	
dim	#pits	%density	#decisions	%win	decision	game
4×4	0	0.0	13,770	97.6	3.5E-2	4.9E-1
6×6	1	2.7	30,666	95.0	1.6E-1	5.1
8×8	2	3.1	54,528	94.8	4.1E-1	22.7
10×10	3	3.0	85,635	93.0	8.5E-1	73.5
12×12	4	2.7	123,921	93.6	1.3	173.4
14×14	5	2.5	159,977	93.4	2.2	352.4
16×16	6	2.3	231,307	91.7	3.1	722.0
18×18	7	2.1	309,919	90.0	4.1	1,282.3
20×20	8	2.0	362,816	90.8	5.3	1,942.8

Table 5: Results for the Non-Deterministic Moving Wumpus domain. For each grid size, averages over 1,000 runs shown. The table shows the total number and density of pits in the grid, the total number of decisions across all the runs, the percentage of runs in which the agent found the gold, and the average time in seconds per decision and game.

12. Related Work

The formulation in the paper is closely related to recent translation-based approaches to conformant and contingent planning that compile beliefs away (Palacios & Geffner, 2009; Albore et al., 2009). These translations, however, assume that the problems are *deterministic*. Our account yields similar widths on most deterministic benchmarks, but is simpler, because it is defined over multi-valued variables, and is more general, because it handles non-deterministic actions. Yet our account is also less tight on some deterministic problems. As an illustration, if $I = \{x_1 \vee \cdots \vee x_n\}$ and the actions are a_i, each with conditional effect $x_i \rightarrow G$, $i = 1, \ldots, n$, the conformant problem with goal G has width 1 in Palacios and Geffner's account, but width n in ours. The relevance account based on literals is indeed finer than the one based on variables but it is also more difficult to generalize to non-deterministic settings. This difference does not seem to have practical effects over most benchmarks where disjunctions in the initial situation are exclusive and implicitly encode the possible values of a set of multi-valued variables. Another important difference with these approaches is that complete translations are *always* exponential in the problem width, while our complexity bound is worst case; i.e., if the variables in contexts are highly correlated, the actual complexity of factored belief tracking can be much lower.

The notion of width appears also in Bayesian networks where inference is exponential in the width of the network (Pearl, 1988). Three differences that can be pointed out in relation to our notion of width are that 1) we exploit the knowledge that certain variables are not observable, 2) we can determine and use the knowledge that certain variables are determined, and 3) we make use of the distinction between action conditions and preconditions in planning. As an example, a problem where an agent has to go through n doors whose status, open or closed, can only be observed when the agent is near the door, will have width no smaller than n when modeled as a dynamic Bayesian network, as all the door variables affect the agent location variable. In our setting, however, the problem has width

1 because the status of a door need to be known to the agent before it can open, close or walk through the door.

The causal decomposition and the resulting causal belief tracking algorithms are similarly related to the ideas of variable splitting or renaming in graphical models, where a variable X appearing in different factors f_i is replaced by different variables X_i, one per factor f_i (Choi & Darwiche, 2006; Ramirez & Geffner, 2007), so that the problem width can be reduced. Then, equality constraints relating the X_i variables must be enforced. Approximate belief tracking algorithms for dynamic bayesian networks and POMDPs have also appealed to the idea of decomposing global beliefs over all the variables into local beliefs over subsets of variables (Boyen & Koller, 1998; Shani, Poupart, Brafman, & Shimony, 2008). A key difference with the causal belief tracking algorithm is that we provide the conditions under which this type of decomposition remains sound and complete. On the other hand, we only deal with uncertainty represented by sets of states, not probability distributions.

A number of logical schemes for representing and tracking beliefs have been used and developed in contingent planning, appealing to OBDDs, CNF, and DNF representations (Bertoli et al., 2001; Bryce et al., 2006; To et al., 2011), relevance considerations (Tran, Nguyen, Son, & Pontelli, 2013), and lazy SAT and regression techniques (Hoffmann & Brafman, 2005; Rintanen, 2008; Shani & Brafman, 2011). None of these approaches, however, has been tried on the domains considered in this paper or over instances of similar size. Indeed, while the causal width of these domains bounds the complexity of beam tracking, no similar bound is known for these schemes that unlike beam tracking are complete. Moreover, while in principle some of these schemes handle non-determinism naturally, other methods like those based on SAT do not. The K-replanner (Bonet & Geffner, 2011) is also based on a very efficient and effective belief tracking method that is polynomial but not fully general and cannot deal with non-deterministic actions. The follow up LW1 planner (Bonet & Geffner, 2014) shares the features of the K-replanner and is complete for width-1 problems.

From an experimental perspective, several comments and questions are in order on the relation between beam tracking and the algorithms used for belief tracking in contingent planners over the existing benchmarks. First of all, practically all of the benchmarks used so far in contingent planning are *easy* from a belief tracking point of view. Indeed, the quadratic and linear time representation of beliefs in CLG and LW1 respectively, have been shown to be adequate for all such problems, including the Wumpus problems above. The exception to this is Minesweeper, where belief tracking is provably NP-hard and where the linear approximation in LW1 turns out to be much weaker than beam tracking, failing to solve "without guessing" most of the instances that beam tracking can solve in this way (Bonet & Geffner, 2014). This means that, whether the width of these problems is low or high, their *effective width* is 1, and in such cases, beam tracking cannot help computationally, and actually, may degrade performance (except in Minesweeper), as beam tracking is exponential in the problem *causal width*, which while lower than width in general is usually higher than 1. The effective width of a problem P is the minimum non-negative integer value i such that the contingent translation $X_i(P)$ (Palacios & Geffner, 2009; Albore et al., 2009) has a solution. The effective width of a problem is never greater than its width but can be much smaller than both its width and than its causal width. For example, a

						avg. time per	
dim	#mines	%density	%succ	%failure	%aborted	decision	game
8×8	10	15.6	93.0	7.0	0.0	0.8	56.3
16×16	40	15.6	94.0	6.0	0.0	4.9	1,268.4
30×16	99	20.6	65.0	6.0	29.0	12.4	5,998.6

Table 6: Comparison with the SDR on-line planner over Minesweeper instances. SDR is fed with random hidden states and solutions (action sequences) computed by beam tracking with no guessing. The planner task is then to check the applicability of actions in the given solution and whether the goal holds. For each instance size, SDR is tested over 100 different random problems. The column failure indicates the number of times that SDR was not able to verify a correct solution, while the column aborted indicates the number of times that SDR terminated early due to a bug. Times are in seconds. Beam tracking takes a few seconds for solving these instances (see Table 3).

problem with actions a_i with conditional effects that map valuations v_i of a set of variables $X_1, \ldots, X_n$ into the goal literal $Y = y$, will have a width and a causal width not smaller than n as all the variables X_i are causally relevant to Y. Yet the effective width of such a problem may be 1 if the values of each the X_i variables can be observed directly or inferred from the observations, or also, if the goal can be achieved without using any of these actions at all. In this sense, while the notion of effective width provides a *lower bound* on the number of state variables whose uncertainty must be tracked jointly in order to make the problem solvable, the notions of width, characterized syntactically, provides an *upper bound* on the number of state variables whose uncertainty must be tracked jointly so that no solution would be missed. The gap between these two bounds can be large indeed, and obtaining syntactic characterizations of the former is an open problem.

A related question is how the various belief tracking algorithms used in contingent planning such as regression, OBDDs, CNF, and DNF, scale up over these domains. While a general comparison of these *complete* but *exponential* algorithms with *incomplete* and *polynomial* algorithms like beam tracking (over domains with bounded causal width) would not be fair, it would still be interesting to find out in which of the "easy" cases these algorithms scale up polynomially and in which exponentially. Performing these tests, however, is not simple, as it requires getting into the code of the planners so that they would all follow a fixed common policy in each instance, thus leaving the planning component aside. Moreover, even fixing a policy for each instance, is not enough, as some of the planners are off-line and hence track beliefs over many possible executions and not just one, as in the case of on-line planners.

Just for the purpose of an illustration we performed this test in one of the difficult domains, Minesweeper, by supplying the on-line planner SDR (Shani & Brafman, 2011) the execution computed from beam tracking along with the hidden initial state for such an execution. In this setting, the on-line planner SDR is not doing planning, but rather it is tracking the beliefs over the problem to verify goal achievement and the preconditions of the given applicable action at each time point. These are all Minesweeper instances solved by beam tracking without guessing; i.e., by pure inference after a first fixed choice. Table 6 shows the results for SDR that tracks beliefs using a form of regression (Rintanen, 2008; Shani & Brafman, 2011). Two observations can be made by comparing the results in this table with those in Table 3 for beam tracking. First, while SDR takes 56.3, 1,268.4 and 5,998.6 seconds on average for *verifying* solutions for 8×8, 16×16 and 30×16 instances respectively, beam tracking takes 0.21, 1.42 and 2.86 seconds for *finding* these solutions by following a greedy policy. Since finding solutions is more expensive than verifying them – one must at least identify all applicable actions at each time point – the difference in performance turns out to be of several orders of magnitude, growing with the grid size. In addition, the regression mechanism in SDR fails to verify correct solutions in several cases and aborts with failure in a large number of cases for the large instances. In any case, the performance gap is not surprising: belief tracking in Minesweeper is NP-hard, thus complete algorithms like regression will run in exponential time in the worst case, while beam tracking remains polynomial as the causal width of the domain is bounded. In other challenging problems, the gap in performance between beam tracking and complete belief tracking algorithms will be similar. Beam tracking will be useful then if the causal width of the problem is bounded and not too large, trading off in a principled way completeness by tractability.

13. Summary

Effective belief tracking is crucial for planning with incomplete information and sensing. While the problem is intractable in general, it has been shown elsewhere that belief tracking over deterministic problems is exponential in a width parameter that is often bounded and small. In this work, we have introduced a related formulation that applies to non-deterministic problems as well. The *factored belief tracking* algorithm results from a set of projected problems whose size is bounded by the problem width. The beliefs over goals and preconditions are then obtained directly from the beliefs over these projected problems that can be maintained independently. We have then developed a *different decomposition scheme* and belief tracking algorithm that maintains beliefs over smaller projections, and have provided the conditions under which the algorithm is complete. *Causal belief tracking* is space exponential in the problem *causal width* but remains time exponential in the problem *width*, as the global consistency of the beliefs over the smaller projections need to be enforced. Finally, *beam tracking* is a sound but incomplete approximation of causal belief tracking where global consistency is replaced by a local but powerful form of consistency. Beam tracking runs in *time and space that are exponential in the problem causal width* that is often much smaller than the problem width. We have tested beam tracking over large instances of Battleship, Minesweeper, and Wumpus, in combination with simple heuristics for action selection, where performance compares well with state-of-the-art solvers while using orders-of-magnitude less time. In the future, we would like to explore extensions of the proposed framework for belief tracking in POMDPs, where belief states are not sets of states but probability distributions, and particle-based algorithms provide a common approximation (Doucet et al., 2000).

Acknowledgments

We thank Gabriel Detoni for his Java Tewnta framework (`http://code.google.com/p/tewnta`) for implementing client/server games with a graphical interface on which we developed graphical interfaces for Battleship, Minesweeper and Wumpus. Thanks also to James Biagioni for his `wumpuslite` JAVA simulator (`http://www.cs.uic.edu/~jbiagion/wumpuslite.html`) that we adapted to run the experiments for Wumpus, and to Guy Shani for help running SDR. Hector Geffner is partially supported by EU FP7 Grant# 270019 (Spacebook) and MICINN CSD2010-00034 (Simulpast).

Appendix A. Proofs

Formal results that are needed but which are not stated as propositions or theorems in the main text of the article appear here in the form of lemmas.

A.1 Complexity of Flat Belief Tracking

Let us first formally define the decision problems BTP and GBTP. BTP is the language

$$\text{BTP} = \{\langle P, \tau\rangle : P \text{ is a contingent problem, } \tau \text{ is a possible execution, and } b_\tau \models G\}$$

where $P = \langle V, I, A, G, V', W\rangle$, $\tau = \langle a_0, o_0, \ldots, a_n, o_n\rangle$ is an execution, and b_τ is the belief that results of the execution of τ on the initial belief state. GBTP is like BTP except that it consists of triplets $\langle P, \tau, \ell\rangle$ such that P is a contingent problem, τ is a possible generalized execution, ℓ is a goal, precondition or observation literal, and $b_\tau \models \ell$.

Observe that BTP and GBTP respectively include the tuples $\langle P, \tau\rangle$ and $\langle P, \tau, \ell\rangle$ such that the problem has an empty initial belief state, due to two complementary literals appearing as unit clauses in I, since in such case every execution is trivially possible and b_τ trivially entails any literal ℓ.

Proposition 3. *BTP is polynomial-time reducible to GBTP.*

Proof. The idea is to map a normal execution τ into a generalized execution τ_m that results of replacing each pair $\langle a, o\rangle$ in τ by the sequence $\langle a, \ell_1, \text{NOOP}_a, \ell_2, \ldots, \text{NOOP}_a, \ell_{|V'|}\rangle$ where $\ell_1, \ldots, \ell_{|V'|}$ are the observation literals made true by o, one for each observable variable in V', and NOOP_a is the action that requires nothing and does nothing and whose sensor model is $W_{\text{NOOP}_a}(\ell) = W_a(\ell)$ for each observation literal ℓ.

Formally, given an instance $\langle P, \tau\rangle$ for BTP, the reduction must generate in polynomial time an instance $\langle P', \tau', \ell\rangle$ for GBTP such that $\langle P, \tau\rangle \in \text{BTP}$ iff $\langle P', \tau', \ell\rangle \in \text{GBTP}$.

The problem P' is the problem P extended with the actions NOOP_a, a new boolean variable X_{goal} that denotes the achievement of the goal G in P, a new action a_{goal} with precondition G and effect X_{goal}, and a new dummy observable variable Y with domain $\{\top\}$ and models $W_a(Y = \top) = true$ for all actions a. On the other hand, the generalized execution τ' is $\langle \tau_m, a_{goal}, Y = \top\rangle$ and $\ell = X_{goal}$. Clearly, the reduction works in polynomial time and $\langle P, \tau\rangle \in \text{BTP}$ iff $\langle P', \tau', \ell\rangle \in \text{GBTP}$. $\square$

Theorem 5. *Flat belief tracking is exponential in $|V_U|$, where $V_U = V \setminus V_K$ and V_K is the set of state variables in V that are determined.*

Proof. As described in Definition 4, flat belief tracking consists of an explicit representation of beliefs as set of states, but some savings in space and time can be obtained by noting that the variables in V_K are determined.

With an explicit representation of beliefs, the belief tracking problem gets trivially solved because checking whether an execution $\tau = \langle a_0, o_0, \ldots, a_n, o_n \rangle$ is possible and a literal ℓ is true after τ reduces to computing the belief b_{n+1} that results from τ and checking whether b_{n+1} is empty and whether every state in it satisfies ℓ. The time complexity of this algorithm is the time needed to compute the initial belief b_0 plus $(n+1)$ multiplied by the time needed to compute b_{i+1} from b_i plus the time needed to check the validity of ℓ. Among these times, the last is the easiest to calculate as it is linear in the size of b_{n+1}. We thus need to bound the first two times. We begin the proof by showing that flat belief tracking can be done in time exponential in $|V|$ and then reduce the exponential dependency from $|V|$ to $|V_U|$.

For computing b_0 it is enough to generate all possible states (valuations of variables) and filter out those that do not satisfy the clauses in I. The total time thus spent is $|V| \times |I| \times 2^{O(|V|)}$ since there are $2^{O(|V|)}$ valuations, $|I|$ clauses, and each clause has at most $|V|$ literals.[9]

The time to compute b_{i+1} from b_i consists of the time to check that the preconditions of a hold at b, and the times to compute b_a from b and b_a^o from b_a when $b = b_i$, $a = a_i$ and $o = o_i$. The preconditions are easily verified by iterating over all states in b. The time for this is bounded by $|V| \times 2^{O(|V|)}$ since a contains at most $|V|$ preconditions and b contains at most $2^{O(|V|)}$ states. If some precondition is not satisfied at some state in b, the execution is not possible.

The belief b_a can be computed from b by iterating over each state s in b, and each possible state s' for b_a, and then and checking whether $s' \in F(a,s)$. The two nested iterations require time $2^{O(|V|)} \times 2^{O(|V|)} = 2^{O(|V|)}$. The test $s' \in F(a,s)$ can be performed in time that is exponential in $|V|$ as follows. Let $C_i \rightarrow E_1^i | \cdots | E_{n_i}^i$, for $1 \leq i \leq m$, be the collection of conditional effects for the action a that *trigger* at the state s. If $s' \in F(a,s)$, then s' is the result of applying one head from each such conditional effect on s. Since the problem has $|V|$ variables, then among the m heads there are at most $|V|$ heads that map s into s' while the rest (if any) are subsumed by the first. All subsets of heads of size at most $|V|$ can be enumerated in $2^{O(|V|)}$ time, for each such subset checking whether s gets mapped into s' requires $O(|V|)$ time. Therefore, checking $s' \in F(a,s)$ requires $2^{O(|V|)}$ time as well as computing b_a from b.

Once b_a is obtained, b_a^o is calculated by removing (filtering) from b_a all states that do not comply with the observation o. For each state s in b_a and each observation literal ℓ compatible with o, the state s belongs to b_a^o iff $s \models W_a(\ell)$. This latter test can be performed in time linear in $|W_a(\ell)|$, the size of the formula $W_a(\ell)$. Hence, since there are $|V'|$ observation literals compatible with o, b_a^o can be computed from b_a in time $O(|b_a| \times |V'| \times |W_a|)$ where $|W_a| = \max_\ell |W_a(\ell)|$ and the max ranges over all observation literals ℓ. If b_a^o is empty and b_a is non-empty, the execution is not possible.

9. In this calculation, we implicitly assume that the variable domains are of constant size. Otherwise, if the domains have size n that is linear in the input size, the number of valuations is bounded by $2^{O(|V| \log n)}$ instead of $2^{O(|V|)}$. In either case, the number of valuations is still exponential in the number of variables as well as the resulting complexity of flat belief tracking.

Once all times are weighed in, we see that flat belief tracking can be done in time that is exponential in $|V|$.

We now reduce the exponent from $|V|$ to $|V_U|$. This is direct since any determined variable has the same valuation across all states in a reachable belief. Hence, such variables do not contribute to increase the number of states in reachable beliefs. Likewise, only subsets of heads of size $|V_U|$ need to be considered when computing the belief b_a from b. Hence, all computations can be done in time and space that is exponential only in $|V_U|$. $\square$

Theorem 6. *BTP and GBTP are Turing complete for the class* P^{NP}.

Proof. By Proposition 3, BTP is polynomial-time reducible to GBTP, and thus it is enough to show the hardness for BTP and the inclusion for GBTP.

The class P^{NP} is the set of all decisions problems that can be decided in (deterministic) polynomial time using an oracle for SAT. To show that BTP is hard for this class, it is enough to show that UNSAT can be reduced in polynomial time to BTP since then every call to the NP oracle can be replaced by a call to the BTP oracle. On the other hand, to show that GBTP belongs to P^{NP}, it is enough to show that there is an algorithm for the complement of GBTP (since P^{NP} is closed under complementation) that runs in polynomial time and that makes calls to an oracle for SAT.

Hardness. Let $\Delta = \{C_1, \ldots, C_m\}$ be a CNF theory over boolean variables $X_1, \ldots, X_n$. We need to construct in polynomial time a contingent problem $P = \langle V, I, A, G, V', W\rangle$ and an execution τ such that $\langle P, \tau\rangle \in$ BTP iff Δ is unsatisfiable. The variables in the problem P are all boolean given by $V = \{X_1, \ldots, X_n, Q\}$ and $V' = \{Z_1, \ldots, Z_m\}$. I is the empty set of clauses and $G = \{Q = true\}$. The actions are $a_1, \ldots, a_m$, all with empty preconditions and no conditional effects, but with sensor model $W_{a_i}(Z_i = true) = C_i \vee Q$ and $W_{a_i}(Z_j = true) = false$ for $j \neq i$. Finally, the execution is $\tau = \langle a_1, o_1, \ldots, a_m, o_m\rangle$ where o_i is the V'-valuation that makes Z_i true and Z_j false for $j \neq i$.

Note that the initial belief contains all the 2^{n+1} V-valuations, half of them satisfying Q and the other half $\neg Q$. After the first observation o_1 is received, only the valuations that satisfy the clause C_1 or Q are preserved. Thus, inductively, after observation o_i is received, only the valuations that satisfy the clauses in $\{C_1, C_2, \ldots, C_i\}$ or Q are preserved. Therefore, b_τ is the set of valuations that satisfy Δ or Q and hence it is non-empty (i.e., τ is a possible execution). Thus, $b_\tau \models G$ iff all valuations for $\neg Q$ are gone iff Δ is unsatisfiable.

Inclusion. The complement of GBTP consists of all tuples $\langle P, \tau, \ell\rangle$ such that b_0 is non-empty and either τ is non-executable or $b_\tau \not\models \ell$. Since I consists only of unit clauses, $b_0 \neq \emptyset$ iff I contains no pair of complementary literals. Assume that $\tau = \langle a_0, \ell_0, a_1, \ell_1, \ldots, a_n, \ell_n\rangle$, where the literals ℓ_i are observation literals, and let b_i be the belief before the action a_i is applied; i.e., $b_i = b_a^o$ for $b = b_{i-1}$, $a = a_{i-1}$ and $o = \ell_{i-1}$. Then, τ is possible iff each b_i is non-empty and each action a_i is applicable at b_i. Assume that we have established that the prefix $\tau_i = \langle a_0, \ell_0, \ldots, a_{i-1}, \ell_{i-1}\rangle$ is possible, then checking whether τ_{i+1} is possible involves two operations: 1) checking that each precondition literal of a_i holds at b_{i-1} and 2) checking whether there is at least one state in b_{a_i} that complies with ℓ_i, for $b = b_{i-1}$.

The first check can be done by calling the SAT oracle over the CNF theory Δ_{i-1}, over time-indexed propositions for state-variable literals and actions, that encodes all possible state trajectories for a fixed valuation of the actions. The time horizon for the theory is

$T = 0, \ldots, i-1$, and the theory is built in such way that it is satisfiable iff there is a state s at time $i-i$ (i.e., s in b_{i-1}) that does not satisfy at least one precondition of a_i. This theory is of polynomial size and can be built in polynomial time. Likewise, the second check can be performed by calling the SAT oracle over the CNF theory Γ_{i-1} with the property that it is satisfiable iff there is a state in b_{i-1} that complies with the observation ℓ_i.

Hence, the algorithm A that decides the complement of GBTP works by building theories Δ_t and Γ_t for $t = 0, \ldots, n$. At each stage t, A ACCEPTs the input if Δ_t is satisfiable or Γ_t is unsatisfiable. If, at the end, the algorithm has not accepted yet, then it builds another theory Δ_{n+1}, that is like Δ_i but instead of checking whether a precondition of action a_i doesn't hold, checks whether the input literal ℓ doesn't hold. If Δ_{n+1} is satisfiable, A ACCEPTs since the belief b_τ does not satisfy ℓ, else A REJECTs because $\langle P, \tau, \ell \rangle \in$ GBTP. $\square$

A.2 Factored Belief Tracking

In the following, for a state s (valuation of variables) and a subset S of variables, we write $s|_S$ to denote the valuation s restricted to the variables in S, also called the projection of s on S. In general, we use the symbols s, t and their primed versions to denote states, and the symbols u, v and their primed versions to denote projected states (restrictions or partial valuations).

Proposition 9. *Belief tracking in the deterministic or* non-deterministic conformant *setting is exponential in the maximum number of non-determined variables that are all causally relevant to a variable appearing in an action precondition or goal.*

Proof. This proposition is a special case of Theorem 15 (and also of Theorem 33). $\square$

Theorem 15. *Belief tracking in P is exponential in $w(P)$.*

Proof. In the conformant setting, there are no observable variables and hence the evidential relevance relation is empty and the relevant relation equals the causally relevant relation. Therefore, the context of a variable X equals the set of variables that are causally relevant to X, and this theorem establish Proposition 9 in the conformant setting.

In the general setting, the theorem is shown by constructing an algorithm for belief tracking whose time complexity is only exponential in $w(P)$. The definition and analysis of the algorithm is done through a series of claims that terminate at Theorem 22 below. $\square$

Proposition 18. *If variable X appears in a goal or precondition, then the number of state variables in P_X that are not determined is bounded by $w(P)$.*

Proof. The number of state variables in P_X is $|Ctx(X)|$ and the number of state variables that are not determined in P_X is $|Ctx(X) \cap V_U|$. By definition of width, this quantity is less than or equal to $w(P)$ when X is a goal or precondition variable. $\square$

We now establish two fundamental lemmas about the progression of actions and projection of observable models. In the following, we say that a subset S of variable is *causally closed* if for any variable $X \in S$ and any variable Y that is causally relevant to X, $Y \in S$. Likewise, the *causal closure* of a variable Z is the minimum (with respect to set inclusion) subset S of variables that is causally closed and includes Z.

Lemma 1 (Factored Progression)**.** *Consider a consistent problem P. Let s be a state, and a be an action that is applicable at s. Then, for any* causally-closed *subset S of variables:*

1) for every u', if $u' \in F_S(a, s|_S)$ then there is $s' \in F(a,s)$ such that $u' = s'|_S$, and

2) for every s', if $s' \in F(a,s)$ then $s'|_S \in F_S(a, s|_S)$

where F_S is the transition function on the projected problem P_S. Therefore, $\Pi_S F(a,s) = F_S(a, s|_S)$ for every state s on which a is applicable, and $\Pi_S F(a,U) = F_S(a, \Pi_S U)$ for every set U of valuations on which a is applicable.

Proof. Part 1. Let u' be an element in $F_S(a, s|_S)$ and let $H_S = \{E^i_S\}_{i=1}^m$ be the collection of heads of the conditional effects $C^i_S \rightarrow E^i_S$ for a that trigger at $s|_S$ and result in u'. For fixed $i \in \{1, \ldots, m\}$, we know that $s|_S \models C^i_S$. If $E^i_S \neq \emptyset$ then, by definition of the causally relevant relation, $Vars(C^i) \subseteq S$ and thus $C^i_S = C^i$. Therefore, $s \models C^i$ and this effect also triggers when a is applied at s. We now show that no other effect that *affects* a variable in S triggers when a is applied at s. Indeed, if a conditional effect $C \rightarrow F$ that affects a variable in S triggers, then $s \models C$ and thus $s|_S \models C_S$ and $F_S \in H_S$. Finally, the effects that trigger and affect the variables in S are the same in both problems P and P_S. Since P is consistent, the set of effects $\{E^i\}_{i=1}^m$ is contained in a set H of heads for the effects that trigger when a is applied at s. Therefore, u' is the projection over S of the state s' that results of applying the effects in H at s; i.e., $u' = s'|_S$.

Part 2. Let s' be an element in $F(a,s)$ and let $H = \{E^i\}_{i=1}^m$ be the collection of heads of the conditional effects $C^i \rightarrow E^i$ for a that trigger at s and result in s'. For fixed $i \in \{1, \ldots, m\}$, we know that $s \models C^i$ and thus $s|_S \models C^i|_S$. Therefore, the effects $C^i_S \rightarrow E^i_S$ also trigger when a is applied at $s|_S$ in P_S. We now show that no other effect that *affects* a variable in S triggers when a is applied at $s|_S$ in P_S. Indeed, let us suppose that a projected conditional effect $C_S \rightarrow F_S$ that affects a variable in S triggers at $s|_S$. Then, $Vars(F) \subseteq S$ and thus $Vars(C) \subseteq S$, since S is causally closed, and $C_S = C$. Therefore, $s \models C$ and this effect also triggers when a is applied at s. Finally, since the effects that trigger and affect the variables in S are the same in both problems P and P_S, then $s'|_S$ is the result of applying the projected effects $\{E^i_S\}_{i=1}^m$ at $s|_S$; i.e., $s'|_S \in F_S(a, s|_S)$. □

Lemma 2 (Observational Closure)**.** *For every variable X, action a, and observation literal ℓ = '$Z = z$', $W_a(\ell)_S$ is either $W_a(\ell)$ or 'true', where $S = Ctx(X)$.*

Proof. Let $\{X_1, \ldots, X_n\}$ be the variables in $W_a(\ell)$. By the definition of the relevant relation, Z is relevant to each X_i and vice versa. Hence, if Z or some X_i belongs to S, then Z and all X_i belongs to S as well. Therefore, $W_a(\ell)_S$ is either $W_a(\ell)$ or true. □

The following results are obtained by induction on the length of the executions. As noted before, there is no loss of generality if we consider generalized executions instead of executions. However, it is easier to consider even more general executions that correspond to finite sequences over the alphabet $A \cup \{\langle a, \ell \rangle : a \in A, \ell \in Lits(V')\}$ where A is the set of actions and $Lits(V')$ is the set of observation literals. This type of executions are more general because they do not require the interleaving of actions and observations; i.e., such an execution may contain multiple actions or observations in sequence. For example, an execution like $\langle a_0, a_1, \langle a_2, \ell_2\rangle, \langle a_3, \ell_3\rangle, \ldots\rangle$ indicates that the initial belief needs to be progressed with the actions a_0 and a_1, then filtered with the formula $W_{a_2}(\ell_2)$, filtered again with the formula $W_{a_3}(\ell_3)$, and so on. As in the case of normal and generalized executions, there is a direct mapping between generalized executions and this new type of executions. If τ is one such execution, then b_τ denotes the belief that results of applying τ at the initial belief, while if $b = b_\tau$, then b_a and $b^{a,\ell}$ denote the beliefs that result from the executions $\tau' = \langle \tau, a\rangle$ and $\tau' = \langle \tau, \langle a, \ell\rangle\rangle$ respectively. Therefore, when making induction on the length of executions to prove a claim, we need to show the claim for the initial belief that corresponds to the empty execution, and then for beliefs of the form b_a and $b^{a,\ell}$ for $b = b_\tau$.

The next definition and lemma make precise the notion of 'decomposable belief' that plays a fundamental role in our results. Intuitively, a belief b is decomposable when for every pair of states $s, t \in b$, there is a state $w \in b$ that agrees with s on the variables in a subset S and agrees with t on the variables in a subset T (where S and T are certain subsets of variables); in symbols, there is $w \in b$ such that $w|_S = s|_S$ and $w|_T = t|_T$.

Definition and Lemma 3 (Decomposability). *A belief state b is* decomposable *iff for every variable X, observation literal ℓ = '$Z = z$', action a, and subset $T \supseteq Vars(W_a(\ell))$ such that T is causally closed and $T \cap Ctx(X) = \emptyset$, it holds:*

$$\forall s, t \left[s, t \in b \implies \exists w \big(w \in b \wedge w|_{Ctx(X)} = s|_{Ctx(X)} \wedge w|_T = t|_T \big) \right] .$$

It turns out that every reachable belief is decomposable.

Proof. Let b be a reachable belief. Then, there is an execution τ such that $b = b_\tau$. The proof is by induction on the length of τ. If τ is empty, the claim holds since I contains only unit clauses and $T \cap Ctx(X) = \emptyset$.

Assume now that all beliefs reachable through executions of length less than or equal to n are decomposable, and consider an execution τ' of length $n + 1$ that augments an execution τ of length n. In the following, b denotes the belief b_τ, and $res(a, s)$ denotes the state that results of applying a *deterministic* action a on state s.

Case: $\tau' = \langle \tau, a'\rangle$. Let X, ℓ, a and T be as in the statement of the lemma, $S = Ctx(X)$, and let s', t' be two states in $b_{a'}$. Therefore, there are two "determinizations" a_1 and a_2 of a' such that $s' = res(a_1, s)$ and $t' = res(a_2, t)$ for some $s, t \in b$. Apply inductive hypothesis to obtain $w \in b$ such that $w|_S = s|_S$ and $w|_T = t|_T$. Since S and T are disjoint and causally closed, there is a determinization[10] a_3 of a' such that $res(a_1, s)|_S = res(a_3, w)|_S$ and $res(a_2, t)|_T = res(a_3, w)|_T$. The sought $w' \in b_{a'}$ is thus $w' = res(a_3, w)$.

Case: $\tau' = \langle \tau, \langle a', \ell'\rangle\rangle$. As before, let X, ℓ, a and T be as in the statement of the lemma, let ℓ' = '$Z' = z'$', and let s', t' be two states in $b^{a',\ell'}$. We consider the two subcases whether $Vars(W_{a'}(\ell')) \subseteq S$ or not, for $S = Ctx(X)$:

Subcase: $Vars(W_{a'}(\ell')) \subseteq S$. Since, $s', t' \in b$, apply the inductive hypothesis to get $w' \in b$ such that $w'|_S = s'|_S$ and $w'|_T = t'|_T$. Then, $w' \in b^{a',\ell'}$ because $w'|_S = s'|_S \models W_{a'}(\ell')_S = W_{a'}(\ell')$.

10. The existence of this determinization is granted by the second assumption on the planning problem and the fact that the sets S and T of variables are disjoint.

Subcase: $Vars(W_{a'}(\ell')) \nsubseteq S$. By Lemma 2, $Vars(W_{a'}(\ell')) \cap S = \emptyset$. Let T' be the minimal causally-closed subset of variables that includes T and $Vars(W_{a'}(\ell'))$. Observe that $T' \cap S = \emptyset$ since if Y belongs to the intersection, then Z' is relevant to Y, Y is relevant to X, and thus Z' is relevant to X contradicting $Vars(W_{a'}(\ell')) \cap S = \emptyset$. Apply the inductive hypothesis using T' to get $w' \in b$ such that $w'|_S = s'|_S$ and $w'|_{T'} = t'|_{T'}$. This is the w' that we are looking for because $T \subseteq T'$ and thus $w'|_T = t'|_T$, and because $w' \in b^{a',\ell'}$ since $w'|_{T'} = t'|_{T'} \models W_{a'}(\ell')_{T'} = W_{a'}(\ell')$. □

A last technical lemma, before giving the proofs of Theorems 20 to 22, that establish the existence of partial valuations in the projection of filtered beliefs is the following:

Lemma 4 (Factored Filtering)**.** *Let X be a variable, $S = Ctx(X)$, b be a reachable belief, a be an action, and ℓ = '$Z{=}z$' be an observation literal. If $b^{a,\ell}$ is non-empty and u is such that $u \in \Pi_S\, b$ and $u \models W_a(\ell)_S$, then $u \in \Pi_S\, b^{a,\ell}$.*

Proof. Assume that $b^{a,\ell}$ is non-empty and let u be an S-valuation that satisfies the antecedent in the lemma. If $W_a(\ell)_S = W_a(\ell)$, then $u \models W_a(\ell)$ and $u \in \Pi_S\, b^{a,\ell}$.

If $W_a(\ell)_S \neq W_a(\ell)$ then by Lemma 2, $Vars(W_a(\ell)) \cap S = \emptyset$. Let T be the minimal causally-closed subset of variables that includes $Vars(W_a(\ell))$. Note that if $Y \in S \cap T$ then Z is evidentially relevant to Y which is relevant to X, and thus Z is relevant to X and $Vars(W_a(\ell)) \subseteq S$. Therefore, $S \cap T = \emptyset$. Let $t \in b^{a,\ell} \subseteq b$ and apply Lemma 3 to get $w \in b$ such that $w|_S = u$ and $w|_T = t|_T$. Hence, $w|_T \models W_a(\ell)_T = W_a(\ell)$, $w \in b^{a,\ell}$ and $u \in \Pi_S\, b^{a,\ell}$. □

Theorem 20. *For a state variable X, let b and b_X be the beliefs that result from an execution that is possible over both P and P_X. Then, $\Pi_X\, b_X = \Pi_X\, b$.*

Proof. Let τ be an execution that is possible over both P and P_X. We prove the more general result that $b_X = \Pi_S\, b$ for $S = Ctx(X)$; it is more general because $X \in S$ and $\Pi_X\, b_X = \Pi_X \Pi_S\, b = \Pi_X\, b$. The proof is by induction on the length of τ. If τ is the empty execution, then result follows readily since I contains only unit clauses. Assume that the claim holds for executions of length n, and let τ' be an execution of length $n+1$ that augments an execution τ of length n and that is possible over P and P_X. Further, let b and b_X be the beliefs that result from τ in P and P_X respectively. Then, by inductive hypothesis $b_X = \Pi_S\, b$.

Case: $\tau' = \langle \tau, a \rangle$. We need to show that $b_{X,a} = \Pi_S\, b_a$. In the following, F_S denotes the transition function in P_X. The forward inclusion is given by

$$
\begin{aligned}
u' \in b_{X,a} &\overset{①}{\Longrightarrow} \exists u \big[u \in b_X \wedge u' \in F_S(a,u) \big] \\
&\overset{②}{\Longrightarrow} \exists u \exists s \big[u \in b_X \wedge u' \in F_S(a,u) \wedge s \in b \wedge s|_S = u \big] \\
&\overset{③}{\Longrightarrow} \exists u \exists s \exists s' \big[u \in b_X \wedge u' \in F_S(a,u) \wedge s \in b \wedge s|_S = u \wedge s' \in F(a,s) \wedge s'|_S = u' \big] \\
&\overset{④}{\Longrightarrow} \exists s \exists s' \big[s \in b \wedge s' \in F(a,s) \wedge s'|_S = u' \big] \\
&\overset{⑤}{\Longrightarrow} \exists s' \big[s' \in b_a \wedge s'|_S = u' \big] \overset{⑥}{\Longrightarrow} u' \in \Pi_S\, b_a
\end{aligned}
$$

where ① is by the definition of $b_{X,a}$, ② by inductive hypothesis, ③ by Lemma 1, and ⑤ and ⑥ by the definitions of b_a and $\Pi_S\, b_a$ respectively. The backward inclusion is

$$s'|_S \in \Pi_S\, b_a \overset{①}{\Longrightarrow} \exists s\big[s \in b \wedge s' \in F(a,s)\big] \overset{②}{\Longrightarrow} \exists s\big[s \in b \wedge s' \in F(a,s) \wedge s'|_S \in F_S(a, s|_S)\big]$$
$$\overset{③}{\Longrightarrow} \exists s\big[s|_S \in b_X \wedge s'|_S \in F_S(a, s|_S)\big] \overset{④}{\Longrightarrow} s'|_S \in b_{X,a}$$

where ① is by the definition of b_a, ② by Lemma 1, ③ by inductive hypothesis, and ④ by the definition of $b_{X,a}$. Therefore, $b_{X,a} = \Pi_S\, b_a$.

Case: $\tau' = \langle \tau, \langle a, \ell\rangle\rangle$. We need to show that $b_X^{a,\ell} = \Pi_S\, b^{a,\ell}$. The forward inclusion is

$$u \in b_X^{a,\ell} \overset{①}{\Longrightarrow} u \in b_X \wedge u \models W_a(\ell)_S \overset{②}{\Longrightarrow} u \in \Pi_S\, b \wedge u \models W_a(\ell)_S \overset{③}{\Longrightarrow} u \in \Pi_S\, b^{a,\ell}$$

where ① is by the definition of $b_X^{a,\ell}$, ② by inductive hypothesis, and ③ by Lemma 4. The backward inclusion is

$$s|_S \in \Pi_S\, b^{a,\ell} \overset{①}{\Longrightarrow} s \in b \wedge s \models W_a(\ell) \overset{②}{\Longrightarrow} s|_S \in b_X \wedge s|_S \models W_a(\ell)_S \overset{③}{\Longrightarrow} s|_S \in b_X^{a,\ell}\,.$$

where ① is by the definition of $b^{a,\ell}$, ② by inductive hypothesis, and ③ by the definition of $b_X^{a,\ell}$. Therefore, $b_X^{a,\ell} = \Pi_S\, b^{a,\ell}$. ☐

Theorem 21. *1) An execution is possible in P iff it is possible over each of the subproblems P_X for X being a precondition or goal variable in P. 2) For an execution τ and precondition or goal variable X, $X = x$ (resp. $X \neq x$) is true in b iff $X = x$ (resp. $X \neq x$) is true in b_X, where b and b_X are the beliefs that result of executing τ in P and P_X respectively.*

Proof. Part 1. The proof is by induction on the length of the executions. The base case for the induction is for the empty execution which is possible in P and in each P_X. Assume that the claim holds for executions τ of length n, and let b and b_X be the beliefs that result from τ in P and in each subproblem P_X respectively. Let τ' be an execution of length $n+1$ that augments τ. In the following, $\mathcal{F}$ denotes the collection of precondition and goal variables in P, and $S = Ctx(X)$ for $X \in \mathcal{F}$.

Case: $\tau' = \langle \tau, a\rangle$. First, assume that τ' is possible in P. We need to show that τ' is possible on each P_X for $X \in \mathcal{F}$. By assumption, for each literal $\ell \in Pre(a)$ and each $s \in b$, $s \models \ell$. Let ℓ be a literal in $Pre(a)_S$ and $u \in b_X$ for $X \in \mathcal{F}$. Then, $Vars(\ell) \subseteq S$ and, by inductive hypothesis and Theorem 20 (since τ is applicable at P and P_X) applied to τ, $u = s|_S$ for some $s \in b$. Therefore, $u \models \ell$ and a is applicable at b_X.

Now, assume that τ' is possible in P_X for each $X \in \mathcal{F}$. We need to show that τ' is possible in P. If $\ell =$ '$X = x$' is a precondition of a, then $\ell \in Pre(a)_S$ and ℓ holds at each state $u \in b_X$. If $s \in b$ then, by inductive hypothesis and Theorem 20 applied to τ, there is $u \in b_X$ such that $s|_S = u$. Thus, $s \models \ell$ and a is applicable at b.

Case: $\tau' = \langle \tau, \langle a, \ell\rangle\rangle$. First, assume that τ' is possible in P; i.e., $b^{a,\ell}$ is non-empty. We need to show that $b_X^{a,\ell}$ is non-empty as well for each $X \in \mathcal{F}$. We have

$$s \in b^{a,\ell} \overset{①}{\Longrightarrow} s \in b \wedge s \models W_a(\ell) \overset{②}{\Longrightarrow} s|_S \in b_X \wedge s|_S \models W_a(\ell)_S \overset{③}{\Longrightarrow} s|_S \in b_X^{a,\ell}$$

where ① is by the definition of $b^{a,\ell}$, ② by inductive hypothesis and Theorem 20, and ③ by the definition of $b_X^{a,\ell}$. Hence, $b_X^{a,\ell}$ is non-empty.

Finally, assume that τ' is possible in each P_X; i.e., $b_X^{a,\ell}$ is non-empty for each $X \in \mathcal{F}$. We need to show that $b^{a,\ell}$ is non-empty. Let $X \in \mathcal{F}$ be such that $W_a(\ell)_S = W_a(\ell)$ for $S = Ctx(X)$; it exists because of the fourth assumption on the problem P and Lemma 2. We have

$$\begin{aligned} u \in b_X^{a,\ell} &\overset{①}{\Longrightarrow} u \in b_X \wedge u \models W_a(\ell)_S \overset{②}{\Longrightarrow} \exists s\big[u \in b_X \wedge u \models W_a(\ell)_S \wedge s \in b \wedge u = s|_S\big] \\ &\overset{③}{\Longrightarrow} \exists s\big[u \in b_X \wedge s \models W_a(\ell) \wedge s \in b \wedge u = s|_S\big] \\ &\overset{④}{\Longrightarrow} \exists s\big[s \models W_a(\ell) \wedge s \in b\big] \overset{⑤}{\Longrightarrow} \exists s\big[s \in b^{a,\ell}\big] \end{aligned}$$

where ① is by the definition of $b_X^{a,\ell}$, ② by inductive hypothesis and Theorem 20, ③ by $W_a(\ell)_S = W_a(\ell)$, and ⑤ by the definition of $b^{a,\ell}$. Hence, $b^{a,\ell}$ is non-empty.

Part 2. Let τ be a possible execution in P, and hence, by part 1, also possible in P_X. Let b and b_X be the beliefs that result from τ in P and P_X respectively. By Theorem 20, $\Pi_X\, b_X = \Pi_X\, b$. Therefore, $X = x$ (or $X \neq x$) holds at b_X iff it holds at b. □

Theorem 22. Flat belief tracking *over each of the projected problems P_X for X being a precondition or goal variable in P, provides a sound and complete* factored algorithm *for belief tracking over P that is time and space exponential in the width of P.*

Proof. This is direct from Theorem 21. Let τ be an execution and b_τ and $b_{X,\tau}$ be the beliefs that result of executing τ in P and P_X respectively. Then, τ is possible in P iff it is possible at each P_X. Therefore, flat belief tracking for the subproblems P_X tells whether τ is possible or not in P. Furthermore, for each precondition or goal variable X, $X = x$ holds at b_τ iff it holds at $b_{X,\tau}$. Thus, flat belief tracking for the subproblems P_X is sufficient to determine when an action is applicable or a goal belief has been reached.

By Theorem 5, flat belief tracking for subproblem P_X is exponential in $|Ctx(X) \cap V_U|$. Therefore, flat belief tracking for all subproblems P_X (simultaneously) is exponential only in $\max_X |Ctx(X) \cap V_U|$ where the max ranges over the precondition or goal variables X. This latter expression is the one that defines $w(P)$. □

Proposition 19. *If an execution $\langle a_0, o_0, a_1, o_1, \ldots\rangle$ is possible in P, then it is also possible in P_X for any state variable X in P.*

Proof. If X is a precondition or goal variable, then the claim follows from Theorem 21. So, assume that X is a state variable that does not appear as a precondition or goal. We show using induction on the length of the (generalized) execution τ that if τ is possible in P then it is also possible in P_X. The base case for empty executions is direct. Consider an execution τ' of length $n+1$ that extends an execution τ of length n. Let b and b_X be the result of applying the execution τ in P and P_X respectively, and let $S = Ctx(X)$.

Case: $\tau' = \langle \tau, a\rangle$. Let ℓ = '$Y = y$' be a precondition in $Pre(a)_S$ and $T = Ctx(Y)$. Then, $Y \in S$ and $Ctx(Y) \subseteq Ctx(X)$ because Y is relevant to X. By Lemma 5 (below), $b_Y \supseteq \Pi_T b_X$.

On the other hand, by Theorem 21, $s \models \ell$ for every $s \in b_Y$. Therefore, ℓ holds at each state s in b_X, a is applicable at b_X, and τ' is possible at P_X.

Case: $\tau' = \langle \tau, \langle a, \ell \rangle \rangle$. If $W_a(\ell)_S = true$, then $b_X^{a,\ell} = b_X$ which is non-empty by inductive hypothesis and thus τ' is possible at P_X. If $W_a(\ell)_S \neq true$ and $\ell =$ '$Z = z$', then Z is relevant to X. Since by assumption there is a precondition or goal variable Y such that Z is relevant to Y, it is not difficult to show that X is relevant to Y. Thus, $Ctx(X) \subseteq Ctx(Y)$ and $b_X \supseteq \Pi_S b_Y$ by Lemma 5. Since τ' is possible in P and $b_Y^{a,\ell}$ is non-empty by Theorem 21, then $b_X^{a,\ell}$ is non-empty and τ' is possible in P_X. □

A.3 Causal Belief Tracking

Lemma 5 (Soundness of Causally-Closed Decompositions). *Let $D = \langle T, B \rangle$ be a decomposition whose beams are causally closed, and let P_X^D be the subproblem corresponding to the projection of P on the variables in $B(X)$ for $X \in T$. For any target variable $X \in T$, if b and b_X are the beliefs resulting from an execution on P and P_X^D respectively, then $b_X \supseteq \Pi_{B(X)} b$.*

Proof. The proof is by induction on the length of the executions. For the empty execution, the claims holds since I contains only unit clauses. Let τ' be an execution of length τ' that augments τ. In the following, b and b_X denote the beliefs in P and P_X^D resulting after execution τ, and S denotes $B(X)$.

Case: $\tau' = \langle \tau, a \rangle$. Let $u' \in \Pi_S b_a$. Then, there is $s \in b$ such that $u' \in \Pi_S F(a, s)$. By Lemma 1, $u' \in F_S(a, s|_S)$. Thus, since $s|_S \in b_X$ by inductive hypothesis, $u' \in b_{X,a}$.

Case: $\tau' = \langle \tau, \langle a, \ell \rangle \rangle$. Let $s|_S \in \Pi_S b^{a,\ell}$. We have $s \in b$ and $s \models W_a(\ell)$. By inductive hypothesis, $s|_S \models W_a(\ell)_S$ and $s|_S \in b_X$. Thus, $s|_S \in b_X^{a,\ell}$. □

Theorem 28. *Decoupled CBT runs in time and space that are exponential in $w_c(P)$, and it is sound but not complete. That is, for any target variable X in the causal decomposition, if b and b_X are the beliefs resulting from an execution on P and P_X^C respectively, then $b_X \supseteq \Pi_{B_C(X)} b$ is necessarily true, but $b_X \subseteq \Pi_{B_C(X)} b$ is not.*

Proof. Soundness follows directly from Lemma 5. The bounds on time and space are also direct because the size of each beam $B_C(X)$ is bounded by the causal width $w_c(P)$. □

Theorem 30. *CBT is* space *exponential in the* causal width *of the problem, and* time *exponential in its* width.

Proof. CBT maintains beliefs over the beams of the causal decomposition whose size are bounded by the causal width of the problem. The join-project operation in CBT can be performed across time, by considering one valuation at a time, without the need to first compute and store the full joint. This is done by recursively iterating over all the beliefs $(b_Y)_a^o$ that participate in the join in (5), combining partial valuations from each belief, and then storing its projection on the resulting belief b_X^{i+1}. The number of valuations in the join in (5) is bounded by $2^{O(w(P))}$ as any variable $Z \in B_C(Y)$, for Y relevant to X, is relevant to X and thus $Z \in Ctx(X)$. □

It only remains to show Theorem 33 (stated below). The proof is not straightforward so we split it in two parts. The first part reformulates CBT into an algorithm called Wide (Causal) Belief Tracking (WBT), that is like CBT but performs the join operation over the beliefs for *all* the variables in the problem and not only for the variables that are relevant to X, and shows the soundness and completeness of WBT. In the second part, we show that CBT is simply WBT applied to the subproblem $P_{Ctx(X)}$ associated to the variable X in the factored decomposition F, and then use the soundness and completeness of the factored decomposition to finish the proof. The first part of the proof consists of Lemmas 6–8, while the second part consists of Lemma 9 and Theorem 33.

WBT works on the causal decomposition $C = \langle T_C, B_C \rangle$ like CBT. The beliefs at time 0 for WBT are the same as for CBT: they are the initial belief projected into the causal beams $B_C(X)$ for $X \in T_C$. Beliefs at later times are associated with executions τ' that augment executions τ. If we denote the belief for variable $X \in T_C$ and execution τ with $b_{X,\tau}$, then the update equations for WBT are:

$$b_{X,\langle\tau,a\rangle} = \Pi_{B_C(X)} \bowtie \{F_T(a, b_{Y,\tau}) \,:\, Y \in T_C\}\,, \tag{8}$$

$$b_{X,\langle\tau,\langle a,\ell\rangle\rangle} = \Pi_{B_C(X)} \bowtie \{Filter(W_a(\ell)_T, b_{Y,\tau}) \,:\, Y \in T_C\} \tag{9}$$

where $T = B_C(Y)$ is the beam for Y, $F_T(a, U)$ is the set $\cup_{u \in U} F_T(a, u)$, and $Filter(\varphi, U)$ is the set $\{u \in U : u \models \varphi\}$. These equations are essentially the equation (5) for CBT, where progression and filtering had been separated, except that the join is performed over all target variables instead of joining only the target variables that are relevant to X.

The following basic facts about joins, projections and filtering are easily shown and will be used in the proofs. (We do not include their proofs here.) In their statements, the sets U and U_i refer to sets of valuations, $\mathcal{S}$ refers to a collection of subset of variables, S refers to a subset of variables and $S_i = Vars(U_i)$, and φ refers to a logical formula. The facts are:

BF1. $U \subseteq \bowtie\{\Pi_S U : S \in \mathcal{S}\}$,

BF2. For collection $\{U_i\}_{i \in I}$, $\bowtie\{\Pi_{S_i} \bowtie \{U_i : i \in I\} : i \in I\} = \bowtie\{U_i : i \in I\}$,

BF3. $\Pi_S Filter(\varphi, U) \subseteq Filter(\varphi_S, \Pi_S U)$.

Definition and Lemma 6. *A decomposition $D = \langle T, B \rangle$* factors *a set U of V-valuations iff $U = \bowtie\{\Pi_{B(X)} U : X \in T\}$.*

A decomposition $D = \langle T, B \rangle$ preserves transitions *in a set U of V-valuations iff for each pair of variables $X, Y \in T$, and $Z \in B(X) \cap B(Y)$, either i) Z is known in U (i.e., $u[Z] = u'[Z]$ for each $u, u' \in U$), ii) $B(X) \cup B(Y) \subseteq B(W)$ for some variable $W \in T$, or iii) for every action a, the transition function $F_S(a, \cdot)$ is 1-1 for variable Z in U, where S is the causal closure of Z.*

Let $D = \langle T, B \rangle$ be a decomposition such that $V = \cup_{X \in T} B(X)$ and $B(X)$ is causally closed for each $X \in T$, U be a set of V-valuations, and φ be a V-formula. The following claims hold:

1. If D factors U, then

$$F(a, U) \subseteq \bowtie\{\Pi_{B(X)} F(a, U) : X \in T\} = \bowtie\{F_{B(X)}(a, \Pi_{B(X)} U) : X \in T\}\,.$$

2. If D factors and preserves transitions in U, then

$$F(a,U) = \bowtie\{\Pi_{B(X)}F(a,U) : X \in T\} = \bowtie\{F_{B(X)}(a,\Pi_{B(X)}U) : X \in T\}\,.$$

3. If D factors U and there is $X \in T$ such that $\varphi_{B(X)} = \varphi$, then

$$\mathit{Filter}(\varphi,U) = \bowtie\{\Pi_{B(X)}\mathit{Filter}(\varphi,U) : X \in T\} = \bowtie\{\mathit{Filter}(\varphi_{B(X)},\Pi_{B(X)}U) : X \in T\}\,.$$

Proof. Part 1. The containment is direct by BF1, while the equality follows directly from $\Pi_{B(X)}F(a,U) = F_{B(X)}(a,\Pi_{B(X)}U)$ by Lemma 1.

Part 2. The second equality and the forward inclusion for the first equality are the same as in Part 1. We thus only need to show $F(a,U) \supseteq \bowtie\{\Pi_{B(X)}F(a,U) : X \in T\}$. Let u' be an element in the right-hand side of this expression and $X \in T$. Then, $u'|_{B(X)} \in \Pi_{B(X)}F(a,U)$ and so (by Lemma 1) there is $u_X \in \Pi_{B(X)}U$ such that $u'|_{B(X)} \in F_{B(X)}(a,u_X)$. We claim that $\{u_X\}_{X\in T}$ is a consistent collection of valuations. Indeed, if it is not, there are valuations u_X, u_Y and variable Z such that $u_X[Z] \neq u_Y[Z]$. Clearly, Z is not known in U. If $B(X) \cup B(Y) \subseteq B(W)$ for some $W \in T$, then we can exchange u_X and u_Y by $u_W|_{B(X)}$ and $u_W|_{B(Y)}$ respectively. Otherwise, we see that the function $F_S(a,\cdot)$, where S is the causal closure for Z, is not 1-1 for Z, contradicting the assumptions. Therefore, there is a valuation u such that $u|_{B(X)} = u_X$ for all $X \in T$ (i.e., $u \in \bowtie\{\Pi_{B(X)}U : X \in T\}$) and thus, by the assumption, $u \in U$. Finally, since $\Pi_{B(X)}F(a,u) = F_{B(X)}(a,u|_{B(X)}) = F_{B(X)}(a,u_X)$ by Lemma 1, we have $u'|_{B(X)} \in \Pi_{B(X)}F(a,u)$ and $u' \in F(a,U)$.

Part 3. First, observe that BF1 and BF3 imply the chain of containments

$$\mathit{Filter}(\varphi,U) \subseteq \bowtie\{\Pi_{B(X)}\mathit{Filter}(\varphi,U) : X \in T\} \subseteq \bowtie\{\mathit{Filter}(\varphi_{B(X)},\Pi_{B(X)}U) : X \in T\}\,.$$

We finish by showing that equality holds through by proving that the last subset is contained in the first. Let u' be an element in the last subset. This u' belongs to $\bowtie\{\Pi_{B(X)}U : X \in T\}$ and also to U since D factors U. We thus only need to show that $u' \models \varphi$. This is direct since by assumption there is $X \in T$ with $\varphi_{B(X)} = \varphi$, and thus $u'|_{B(X)} \models \varphi_{B(X)} = \varphi$. □

Lemma 7 (Soundness of WBT)**.** *WBT is sound. That is, if $C = \langle T_C, B_C\rangle$ is the causal decomposition of problem P, $\{b_X\}_{X\in T_C}$ are the local beliefs at time i, and b is the global belief at time i, then $b \subseteq \bowtie\{b_X : X \in T_C\}$ and $\Pi_{B_C(X)}b \subseteq b_X$ for $X \in T_C$.*

Proof. We really only need to proof the first claim $b \subseteq \bowtie\{b_X : X \in T_C\}$ because the second follows directly from it by observing that b_X is a belief over the variables in $B_C(X)$.

The proof of the first claim is by induction on the length of the executions. The base case for the empty execution is easily verified. Assume that the claims hold for executions of length n and let τ' be an execution of length $n+1$ that augments an execution τ of length n. Observe that C factors $U = \bowtie\{b_{Y,\tau} : Y \in T_C\}$ by BF2, and $b_\tau \subseteq U$ by inductive hypothesis.

Case: $\tau' = \langle\tau, a\rangle$.

$$\bowtie\{b_{X,\tau'} : X \in T_C\} \overset{①}{=} \bowtie\{\Pi_{B_C(X)} \bowtie\{F_{B_C(Y)}(a, b_{Y,\tau}) : Y \in T_C\} : X \in T_C\}$$

$$\overset{②}{\supseteq} \bowtie\{\Pi_{B_C(X)} \bowtie\{F_{B_C(Y)}(a, \Pi_{B_C(Y)}U) : Y \in T_C\} : X \in T_C\}$$
$$\overset{③}{\supseteq} F(a,U) \overset{④}{\supseteq} F(a, b_\tau) = b_{\tau'}$$

where ① is by Eq. 8, ② because $b_{Y,\tau} \supseteq \Pi_{B_C(Y)}U$, ③ by part 1 of Lemma 6, and ④ by inductive hypothesis.

Case: $\tau' = \langle \tau, \langle a, \ell \rangle \rangle$.

$$\bowtie\{b_{X,\tau'} : X \in T_C\} \overset{⑨}{=} \bowtie\{\Pi_{B_C(X)} \bowtie\{Filter(W_a(\ell)_{B_C(Y)}, b_{Y,\tau}) : Y \in T_C\} : X \in T_C\}$$
$$\overset{⑩}{\supseteq} \bowtie\{\Pi_{B_C(X)} \bowtie\{Filter(W_a(\ell)_{B_C(Y)}, \Pi_{B_C(Y)}U) : Y \in T_C\} : X \in T_C\}$$
$$\overset{⑪}{=} Filter(W_a(\ell), U) \overset{⑫}{\supseteq} Filter(W_a(\ell), b_\tau) = b_{\tau'}$$

where ⑨ is by Eq. 9, ⑩ because $b_{Y,\tau} \supseteq \Pi_{B_C(Y)}U$, ⑪ by part 3 of Lemma 6, and ⑫ by inductive hypothesis. □

Lemma 8 (Completeness of WBT). *Let $C = \langle T_C, B_C \rangle$ be the causal decomposition of problem P. If C preserves transitions in every* reachable *belief state, then WBT is complete. That is, if $\{b_X\}_{X \in T_C}$ are the local beliefs at time i, and b is the global belief at time i, then $b = \bowtie\{b_X : X \in T_C\}$ and $\Pi_{B_C(X)}b = b_X$ for $X \in T_C$.*

Proof. The proof is by induction on the length of the executions. The base case for the empty execution is easily verified since I contains only unit clauses. Assume that the claims hold for executions of length n and let τ' be an execution of length $n+1$ that augments an execution τ of length n. Observe that C factors $U = \bowtie\{b_{Y,\tau} : Y \in T_C\}$ by BF2, and the inductive hypothesis implies $b_\tau = U$ and $b_{Y,\tau} = \Pi_{B_C(Y)}U$. The proof of the first claim $b = \bowtie\{b_X : X \in T_C\}$ is exactly like the proof of Lemma 7 except that all the containments are replaced by equalities, by either using the part 2 of Lemma 6 or the inductive hypothesis.

For the second claim, we make a similar induction (in tandem with the first induction). Again, the base case of the induction is easily verified. For the inductive step,

Case: $\tau' = \langle \tau, a \rangle$.

$$b_{X,\tau'} \overset{①}{=} \Pi_{B_C(X)} \bowtie\{F_{B_C(Y)}(a, b_{Y,\tau}) : Y \in T_C\} \overset{②}{=} \Pi_{B_C(X)} \bowtie\{F_{B_C(Y)}(a, \Pi_{B_C(Y)}U) : Y \in T_C\}$$
$$\overset{③}{=} \Pi_{B_C(X)}F(a,U) \overset{④}{=} \Pi_{B_C(X)}F(a, b_\tau) = \Pi_{B_C(X)}b_{\tau'}$$

where ① is by Eq. 8, ② and ④ by inductive hypothesis, and ③ by part 2 of Lemma 6.

Case: $\tau' = \langle \tau, \langle a, \ell \rangle \rangle$.

$$b_{X,\tau'} \overset{⑤}{=} \Pi_{B_C(X)} \bowtie\{Filter(W_a(\ell)_{B_C(Y)}, b_{Y,\tau}) : Y \in T_C\}$$
$$\overset{⑥}{=} \Pi_{B_C(X)} \bowtie\{Filter(W_a(\ell)_{B_C(Y)}, \Pi_{B_C(Y)}U) : Y \in T_C\}$$
$$\overset{⑦}{=} \Pi_{B_C(X)}Filter(W_a(\ell), U) \overset{⑧}{=} \Pi_{B_C(X)}Filter(W_a(\ell), b_\tau) = \Pi_{B_C(X)}b_{\tau'}$$

where ⑤ is by Eq. 9, ⑥ and ⑧ by inductive hypothesis, and ⑦ by part 3 of Lemma 6. □

The following lemma shows that the tracking that CBT does on a variable X is equivalent to the tracking that WBT does on the *subproblem* P_X of the factored decomposition (i.e., $P_X = P_{B_F(X)}$ for the factored decomposition $F = \langle T_F, B_F \rangle$).

Lemma 9. *Let $F = \langle T_F, B_F \rangle$ and $C = \langle T_C, B_C \rangle$ be the factored and causal decompositions for problem P. If τ is an execution and $X \in T_C$ is a state variable, then $b_X^C = b_X^W$ where b_X^C denotes the local belief after τ for variable X that is computed by CBT on the problem P, and b_X^W denotes the local belief after τ for variable X that is computed by WBT on the subproblem $P_{B_F(X)}$.*

Proof sketch. This is a simple but tedious proof, so we just provide the sketch. Let $C_X = \langle T_X, B_X \rangle$ be the causal decomposition of the subproblem $P_{B_F(X)}$ for $X \in T_F$ (i.e., the causal decomposition of the subproblem associated with variable $X \in T_F$ in the factored decomposition). The beams that participate in the join in CBT are the beams for the variables $Y \in T_C$ that are relevant to X; all such variables appear in T_X as well. T_X has other variables however: observable variables Y that are not relevant to X. Yet, since all the state variables in $P_{B_F(X)}$ are relevant to X, the projected formulas $W_a(Y = y)_{B_F(X)}$ for such variables Y are all equal to true. Hence, the beams for such variables are over an empty set of variables and just contain the empty valuation. Therefore, such beams can be removed from the join that defines WBT on the problem $P_{B_F(X)}$ without altering its value. The resulting join for WBT on $P_{B_F(X)}$ just contain the beams for variables $Y \in T_C$ that are relevant to X.

Once this fact is observed, the proof consists of a simple induction on the length of the executions. An induction that is left as an exercise. □

Theorem 33. *Causal belief tracking is always sound. It is complete for causally decomposable problems.*

Proof. Let $F = \langle T_F, B_F \rangle$ and $C = \langle T_C, B_C \rangle$ be the factored and causal decompositions for problem P, and $C_X = \langle T_X, B_X \rangle$ be the causal decomposition of the subproblem $P_{B_F(X)}$ for $X \in T_F$ (notice that X is a state variable as T_F is only comprised of such). Further, let τ be an execution, let b_X^C and b_X^W be the local beliefs after τ for variable X that are computed by CBT on problem P and WBT on problem $P_{B_F(X)}$ respectively, let b_X^F be the local belief after τ for variable X that is computed by factored belief tracking on problem P, and let b be the (global) belief after τ on problem P.

If no observable variable is relevant to X, then $B_C(X) = B_F(X)$ and CBT on X is equal to factored belief tracking on X which is sound and complete by Theorem 20. If there are observable variables that are relevant to X, then first notice that

$$b_X^C \;=\; b_X^W \;\supseteq\; \Pi_{B_C(X)} b_X^F \;=\; \Pi_{B_C(X)} b \tag{10}$$

because Lemma 9, the soundness of WBT (cf. Lemma 7), and the soundness and completeness of FBT (cf. Theorem 20). Therefore, CBT is sound.

If the causal decomposition C_X preserves transitions for every reachable belief state in problem $P_{B_F(X)}$, then the containment in (10) is an equality and CBT is complete as well. We thus finish the proof by showing that the decomposition C_X for causally-decomposable problems is a decomposition that preserves transitions for each reachable belief in $P_{B_F(X)}$.

Let $X \in T_C$ be a variable, let b_X be a reachable belief in problem $P_{B_F(X)}$, let X' and X'' be two variables in T_X (the target variables for the causal decomposition of problem $P_{B_F(X)}$), and let Z be a variable in $B_C(X') \cap B_C(X'')$. We will show that either 1) Z is known in b_X, 2) $B_C(X') \cup B_C(X'') \subseteq B_C(W)$ for some variable $W \in T_X$, or 3) for every action a, the transition function $F_S(a, \cdot)$ is 1-1 for variable Z in b_X where S is the causal closure of Z. In such a case, the causal decomposition C_X preserves transitions for every reachable belief in problem $P_{B_F(X)}$.

We consider two cases:

Case: X' or X'' is observable. First, apply the causal-decomposability of P to conclude that either there is a variable $W \in T_C$ that is relevant to X' or X'' with $B_C(W) \supseteq B_C(X') \cup B_C(X'')$, or that Z is a memory variable. In the former case, W is relevant to X and thus belongs to T_X. In the latter case, we will show that either Z is known in b_X or the transition function $F_S(a, \cdot)$ is 1-1 for Z in b_X, where S is the causal closure of Z and a is any action applicable at b_X.

Indeed, for a proof by contradiction let us suppose that Z is not known in b_X and that the transition function is not 1-1. Then, there are two valuations $s_1, s_2 \in b_X$ and two progressions $s'_1 \in F_X(a, s_1)$ and $s'_2 \in F_X(a, s_2)$ such that $s_1[Z] \neq s_2[Z]$ and $s'_1[Z] = s'_2[Z]$. Therefore, from observing the value of Z at state s'_1 and knowing the initial belief and the actions in the execution $\langle \tau, a \rangle$ (where τ is the execution that leads to b_X), one cannot infer the value of Z at b_X because there are two different such values that are compatible with the observation, namely $s_1[Z]$ and $s_2[Z]$. Hence, Z is not a memory variable contradicting the assumed causal-decomposability of P.

Case: X' and X'' are not observables. We further divide this case in two subcases on whether both variables X' and X'' are causal ancestors of X or not. In the affirmative subcase, $B_C(X') \cup B_C(X'') \subseteq B_C(X)$. In the negative subcase, assume without loss of generality that X' is not a causal ancestor of X. Then, there is an observable variable Y such that X' is a causal ancestor of Y and Y is relevant to X. Hence, $B_C(Y) \supseteq B_C(X')$ which implies $Z \in B_C(Y) \cap B_C(X'')$ and this case is reduced to the previous case. □

References

Albore, A., Palacios, H., & Geffner, H. (2009). A translation-based approach to contingent planning. In *Proc. 21st Int. Joint Conf. on Artificial Intelligence*, pp. 1623–1628, Pasadena, California.

Albore, A., Ramirez, M., & Geffner, H. (2010). Compiling uncertainty away in non-deterministic conformant planning. In *Proc. 19th European Conf. on Artificial Intelligence*, pp. 465–470, Lisbon, Portugal.

Albore, A., Ramirez, M., & Geffner, H. (2011). Effective heuristics and belief tracking for planning with incomplete information. In *Proc. 21st Int. Conf. on Automated Planning and Scheduling*, pp. 2–9, Freiburg, Germany.

Amir, E., & Russell, S. (2003). Logical filtering. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence*, pp. 75–82, Acapulco, Mexico.

Bertoli, P., Cimatti, A., Roveri, M., & Traverso, P. (2001). Planning in nondeterministic domains under partial observability via symbolic model checking. In Nebel, B. (Ed.), *Proc. 17th Int. Joint Conf. on Artificial Intelligence*, pp. 473–478, Seattle, WA. Morgan Kaufmann.

Bonet, B., & Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In Chien, S., Kambhampati, S., & Knoblock, C. (Eds.), *Proc. 5th Int. Conf. on Artificial Intelligence Planning Systems*, pp. 52–61, Breckenridge, CO. AAAI Press.

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

Bonet, B., & Geffner, H. (2012a). Action selection for MDPs: Anytime AO* vs. UCT. In *Proc. 26th AAAI Conf. on Artificial Intelligence*, pp. 1749–1755, Toronto, Canada.

Bonet, B., & Geffner, H. (2012b). Width and complexity of belief tracking in non-deterministic conformant and contingent planning. In *Proc. 26th AAAI Conf. on Artificial Intelligence*, pp. 1756–1762, Toronto, Canada.

Bonet, B., & Geffner, H. (2013). Causal belief decomposition for planning with sensing: Completeness results and practical approximation. In *Proc. 23rd Int. Joint Conf. on Artificial Intelligence*, pp. 2275–2281, Beijing, China.

Bonet, B., & Geffner, H. (2014). Flexible and scalable partially observable planning with linear translations. In *Proc. 28th AAAI Conf. on Artificial Intelligence*, pp. 2235–2241, Québec City, Canada.

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. In Cooper, G., & Moral, S. (Eds.), *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, pp. 33–42, Madison, WI. Morgan Kaufmann.

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

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

Choi, A., & Darwiche, A. (2006). An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proc. 21st Nat. Conf. on Artificial Intelligence*, pp. 1107–1114.

Cimatti, A., Roveri, M., & Bertoli, P. (2004). Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*, *159*, 127–206.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*, 229–264.

Dechter, R., & Beek, P. V. (1997). Local and global relational consistency. *Theoretical Computer Science*, *173*(1), 283–308.

Doucet, A., Freitas, N. D., Murphy, K., & Russell, S. (2000). Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proc. 16th Conf. on Uncertainty in Artificial Intelligence*, pp. 176–183.

Edelkamp, S. (2001). Planning with pattern databases. In Cesta, A. (Ed.), *Proc. 6th European Conf. on Planning*, pp. 13–24, Toledo, Spain. Springer: LNCS.

Goldman, R. P., & Boddy, M. S. (1996). Expressive planning and explicit knowledge. In Drabble, B. (Ed.), *Proc. 3rd Int. Conf. on Artificial Intelligence Planning Systems*, pp. 110–117, Edinburgh, Scotland. AAAI Press.

Hoffmann, J., & Brafman, R. I. (2005). Contingent planning via heuristic forward search with implicit belief states. In Biundo, S., Myers, K., & Rajan, K. (Eds.), *Proc. 15th Int. Conf. on Automated Planning and Scheduling*, pp. 71–80, Monterey, CA. Morgan Kaufmann.

Hoffmann, J., & Brafman, R. I. (2006). Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, *170*, 507–541.

Kaelbling, L. P., Littman, M., & Cassandra, A. R. (1999). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Kaye, R. (2000). Minesweeper is NP-Complete. *Mathematical Intelligencer*, *22*(2), 9–15.

Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Proc. 17th European Conf. on Machine Learning*, pp. 282–293. Springer.

Lin, W., Buffet, O., Lee, C., & Teytaud, O. (2012). Optimistic heuristics for Minesweeper. In *Proc. of the Int. Computer Symposium (ICS-12)*. At http://hal.inria.fr/docs/00/75/05/77/PDF/mines3.pdf.

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

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Ramirez, M., & Geffner, H. (2007). Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In *Proc. 13th Int. Conf. on Principles and Practice of Constraint Programming*, pp. 605–619. Springer.

Rintanen, J. (2008). Regression for classical and nondeterministic planning. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., & Avouris, N. M. (Eds.), *Proc. 18th European Conf. on Artificial Intelligence*, pp. 568–572, Patras, Greece.

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

Scott, A., Stege, U., & Rooij, I. V. (2011). Minesweeper may not be NP-Complete but is Hard nonetheless. *Science+Business Media, LLC*, *33*(4), 5–17.

Shani, G., & Brafman, R. I. (2011). Replanning in domains with partial information and sensing actions. In *Proc. 22nd Int. Joint Conf. on Artificial Intelligence*, pp. 2021–2026, Barcelona, Spain.

Shani, G., Poupart, P., Brafman, R. I., & Shimony, S. (2008). Efficient ADD operations for point-based algorithms. In Rintanen, J., Nebel, B., & J. C. Beck, E. A. H. (Eds.), *Proc. 18th Int. Conf. on Automated Planning and Scheduling*, pp. 330–337, Sydney, Australia.

Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Proc. 24th Annual Conf. on Advances in Neural Information Processing Systems*, pp. 2164–2172.

Sipser, M. (2006). *Introduction to Theory of Computation* (2nd edition). Thomson Course Technology, Boston, MA.

Smith, D., & Weld, D. (1998). Conformant graphplan. In Mostow, J., & Rich, C. (Eds.), *Proc. 15th Nat. Conf. on Artificial Intelligence*, pp. 889–896, Madison, WI. AAAI Press / MIT Press.

To, S. T., Pontelli, E., & Son, T. C. (2011). On the effectiveness of CNF and DNF representations in contingent planning. In *Proc. 22nd Int. Joint Conf. on Artificial Intelligence*, pp. 2033–2038, Barcelona, Spain.

Tran, V., Nguyen, K., Son, T. C., & Pontelli, E. (2013). A conformant planner based on approximation: CpA(H). *ACM Trans. on Intelligent Systems and Technology*, *4*(2), 36.

Weld, D., Anderson, C., & Smith, D. (1998). Extending Graphplan to handle uncertainty and sensing actions. In *Proc. 15th Nat. Conf. on Artificial Intelligence*, pp. 897–904. AAAI Press.