Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Causal Belief Decomposition for Planning with Sensing: Completeness Results and Practical Approximation

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

Hector Geffner
ICREA & Universitat Pompeu Fabra
08018 Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

Belief tracking is a basic problem in planning with sensing. While the problem is intractable, it has been recently shown that for both *deterministic* and *non-deterministic* systems expressed in compact form, it can be done in time and space that are exponential in the problem *width*. The width measures the maximum number of state variables that are all *relevant* to a given precondition or goal. In this work, we extend this result both theoretically and practically. First, we introduce an alternative decomposition scheme and algorithm with the same time complexity but different completeness guarantees, whose space complexity is much smaller: exponential in the *causal width* of the problem that measures the number of state variables that are *causally relevant* to a given precondition, goal, or observable. Second, we introduce a fast, meaningful, and powerful approximation that trades completeness by speed, and is both *time and space exponential in the problem causal width*. It is then shown empirically that the algorithm combined with simple heuristics yields state-of-the-art real-time performance in domains with high widths but low causal widths such as Minesweeper, Battleship, and Wumpus.

1 Introduction

Planning with sensing is a search problem in belief space that involves two tasks: keeping track of beliefs and selecting the action to do next. In this paper, we consider the first task in a logical, non-probabilistic setting, where *beliefs* stand for *sets of states*. While belief tracking for problems expressed in compact form is computationally intractable, it has been recently shown that for *deterministic* problems, the problem can be solved in time and space that are exponential in a problem *width* parameter that in many of the existing benchmarks is bounded and small [Palacios and Geffner, 2009; Albore *et al.*, 2009]. The same result has been extended to *non-deterministic* problems where the task of keeping track of global beliefs b over all problem variables is replaced by the task of keeping track of local beliefs b_X for each precondition and goal variable X, each of which involves the variables that are *relevant* to X. The width $w(X)$ of a variable X is the number of variables that are *relevant* to X, and the *width of the problem*, $w(P)$, is the maximum number of variables that are all relevant to a precondition or goal variable. The algorithm that tracks beliefs in time and space that are exponential in the problem width is called *factored belief tracking* [Bonet and Geffner, 2012].

A limitation of these accounts is that many meaningful domains have large, unbounded widths. In this work, we aim to extend these results so that such problems can be handled effectively. For this, we introduce an alternative decomposition scheme and belief tracking algorithm with the same time complexity as factored belief tracking but whose space complexity is often much smaller: exponential in the *causal width* of the problem. The causal width measures the number of state variables that are all *causally relevant* to a given precondition, goal, or observable. We also determine the conditions under which the new belief tracking scheme is complete, and use the new decomposition to define an incomplete but meaningful and powerful approximation algorithm that is practical enough, as it is both *time and space exponential in the problem causal width.*

The paper is organised as follows. After reviewing the basic model and notions from Bonet and Geffner [2012], we introduce the new notion of causal width, a new decomposition scheme, and the new algorithm. We then test the algorithm experimentally.

2 Planning with Sensing

We review the standard model for planning with sensing, and a language for representing these models in compact form.

2.1 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. 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_G is a non-empty set of goal states, A is a set of actions with $A(s)$ denoting the actions applicable in state $s \in S$, and F is a non-deterministic transition function such that $F(a, s)$ denotes the non-empty set of possible successor states that follow action a in state s, $a \in A(s)$.

Conformant planning can be cast as a path finding problem over *beliefs* defined as the sets of states that are deemed possible at any one point [Bonet and 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:

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

where it is assumed that the action a is applicable at each state s in b. *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 $O(s, a)$ that maps 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. Executions in the contingent setting are sequences of action-observation pairs $a_0, o_0, a_1, o_1, \ldots$. 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 Eq. 1, and the belief $b_{i+1} = b_a^o$ that follows from observing the token o is:

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

An execution $a_0, o_0, a_1, o_1, \ldots$ is deemed *possible* in P if starting from the initial belief b_0, each action a_i is applicable at the belief b_i (i.e., $a_i \in A(s)$ for all $s \in b_i$), and b_{i+1} is non-empty. The contingent model is similar to POMDPs but with uncertainty encoded through sets rather than probability distributions.

2.2 Syntax

Syntactically, conformant problems can be expressed in *compact form* through a set of *state variables*, which for convenience we assume to be *multi-valued*. 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 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 a set of conditional effects $C \rightarrow E_1|\ldots|E_n$, where C and each E_i is a set (conjunction) of V-literals. The conditional effect is *non-deterministic* if $n > 1$; else it is deterministic. A conformant 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 valuations over the variables in V, S_0 and S_G are the set of valuations that satisfy I and G, $A(s)$ is the set of operators whose preconditions are true in s, and $F(a, s)$ is determined by the conditional effects of a in the standard way.

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 set of state variables V (i.e., some state variables may be observable), and formulas $W_a(Y = y)$ over a subset of state variables, for each action a and each possible value y of each observable variable Y. The formula $W_a(Y = y)$ encodes the states over which the observation $Y = y$ is possible when a is the last action. A contingent problem is a tuple $P = \langle V, I, A, G, V', W\rangle$ that defines a contingent model that is given by the conformant model $\langle S, S_0, S_G, A, F\rangle$ determined by the first four components in P, and the sensor model $O(s, a)$ determined by the last two components, where $o \in O(s, a)$ 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 simplicity and without loss of generality, we make three simplifying assumptions [Bonet and Geffner, 2012]; namely, that the initial situation I is given by a set of literals, that non-deterministic conditional effects involve just one variable in their heads, and that every observable variable is relevant to a variable appearing in some precondition or goal. If these assumptions are not true for a problem, they can be enforced by means of simple, equivalence-preserving transformations.[1] Extensions of this basic language featuring *defined variables* and *state constraints* are discussed below.

3 Belief Tracking for Planning

A real execution is an interleaved sequence of actions and observations $a_0, o_0, a_1, o_1, \ldots$, where a_i is an action from the problem and o_i is a full valuation over the observable variables. We will find it convenient, however, to consider *generalized executions* $a_0, o_0, a_1, o_1, \ldots$ where the o_i's denote partial valuations, and in particular, *observation literals* $Y = y$. Any real execution can be expressed as a generalized execution provided that the observation literals that arise from the same (full) observation are separated by dummy NO-OP actions with no effects. The *belief tracking for planning* problem can then be expressed as follows:

Definition 1. *The* belief tracking for planning (BTP) problem *is the problem of determining whether a generalized execution $\tau : a_0, o_0, a_1, o_1, \ldots, a_k, o_k$ is possible for a problem P, and whether it achieves the goal.*

In other words, for a planner to be complete, it just needs to determine which observations are possible after an execution, which actions are applicable, and whether the goal has been achieved. There is no need, on the other hand, to determine whether an arbitrary formula is true after an execution. This is an important distinction which is not exploited by the baseline algorithm for BTP, which is called *flat belief tracking* and is the result of the iterative application of Equations 1 and 2. The complexity of flat belief tracking is exponential in the number of state variables $|V|$. Often, however, the value of some variables is not uncertain and such variables do not add up to the complexity of tracking beliefs. We call such variables *determined*. Formally, a *set* of state variables X is *determined* when they are all initially known, they appear in the heads of deterministic effects only, and the variables appearing in the body of such effects, if any, are in the set as well. The set of determined variables for problem P is the maximal set of variables that is determined.

The BTP problem, however, remains intractable in the worst case:

Theorem 2. *The belief tracking for planning (BTP) problem is NP-hard and coNP-hard.*

[1] A longer version of the paper provides the proofs and necessary details.

4 Width and Factored Belief Tracking

Factored belief tracking [Bonet and Geffner, 2012] improves flat belief tracking by focusing on the beliefs that are strictly necessary for solving BTP, and by exploiting the structure of the problem. For this, a variable X is regarded as an *immediate cause* of a variable X' in a problem P, written $X \in Ca(X')$, if $X \neq X'$, and either X occurs in the body C of a conditional effect $C \rightarrow E_1|\cdots|E_n$ such that X' occurs in a head E_i, $1 \leq i \leq n$, or X occurs in a formula $W_a(X' = x')$ for an observable variable X' and value $x' \in D_{X'}$. *Causal relevance* is the transitive closure of this relation:

Definition 3. *X is* causally relevant *to X' in P if $X = X'$, $X \in Ca(X')$, or X is causally relevant to a variable Z that is causally relevant to X'.*

In the presence of observations, however, relevance does not flow only *causally*, but like in Bayesian Networks [Pearl, 1988], it also flows *evidentially:*

Definition 4. *X is* evidentially relevant *to X' in P if X' is causally relevant to X and X is an observable variable.*

Relevance is the transitive closure of the causal and evidential relations taken together:

Definition 5. *X is* relevant *to X' if X is causally or evidentially relevant to X', or X is relevant to a variable Z that is relevant to X'.*

The *width of a variable X* and *the width of the problem* are defined in terms of the relevance relation:

Definition 6. *The* width *of a variable X, $w(X)$, is the number of state variables that are relevant to X and are not determined. The width of the problem P, $w(P)$, is $\max_X w(X)$, where X ranges over the variables that appear in a goal or action precondition.*

The result obtained by Bonet and Geffner [2012] is an algorithm that solves the BTP problem in time and space that are exponential in the problem width. For this, *factored belief tracking* does not track the global belief b over all the problem variables after an execution, but tracks the 'local beliefs' b_X over the state variables that are relevant to each variable X appearing in action preconditions or goals. This is sufficient for solving BTP. The *local beliefs* b_X are defined in turn as the *global beliefs* that result from an execution over a problem that is like P but with all state variable that are not relevant to X, projected away. These *projected problems* are defined as:

Definition 7. *The projection of problem $P = \langle V, I, A, G, V', W\rangle$ on a subset of state variables $S \subseteq V$ is $P_S = \langle V_S, I_S, A_S, G_S, V'_S, W_S\rangle$ where V_S is S, I_S and G_S are the initial and goal formulas I and G (logically) projected over the variables in S, A_S is A but with 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 projected on the variables in S.*[2]

[2]The logical projection of 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 and Marquis, 2002].

For convenience, the projection P_S of P where S is the set of state variables relevant to X is abbreviated as P_X. The projections P_X for variables X appearing in action preconditions or goals ensure three key properties: first, that the real and generalized executions $a_0, o_0, a_1, o_1, \ldots$ that are possible in P are possible in P_X; second, that belief tracking over the projections P_X is time and space exponential in $w(P)$, and finally, that the beliefs b_X and b that result from such executions over P_X and P are *equivalent* over the set of variables relevant to X. *Factored belief tracking* is the algorithm that solves the belief tracking problem over P by keeping track of the local beliefs b_X for each precondition and goal variable X using *flat belief tracking* over each of the projections P_X:

Theorem 8 (Bonet and Geffner [2012]). *Factored belief tracking solves the belief tracking problem for planning in time and space that are exponential in the problem width.*

5 Causal vs. Factored Decompositions

While there are many domains that have a bounded and low width [Palacios and Geffner, 2009; Albore *et al.*, 2009], there are also many meaningful domains that do not. The contribution of this paper is a reformulation of the above results that lead to a fast, meaningful, and powerful approximation belief tracking algorithm that applies effectively to a much larger class of problems. For this, we introduce the idea of *decompositions*, cast factored belief tracking in terms of one such decomposition, and introduce an alternative decomposition leading to different algorithms.

Definition 9. *A* decomposition *of a problem P is a pair $D = \{T, B\}$, 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 such target variables such that $B(X)$ is a set of* state variables *from P.*

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

Definition 10. *The* factored decomposition $F = \{T_F, B_F\}$ *of P is such that T_F stands for the set of variables X appearing in action preconditions or goals, and $B(X)$ is the set of state variables* relevant *to X.*

Factored belief tracking is *flat belief tracking* applied to the subproblems determined by the *factored decomposition*. The complexity of the algorithm is exponential in the problem width, which bounds the size of the beams in the decomposition. The algorithms that we introduce next are based on a different decomposition:

Definition 11. *The* causal decomposition $C = \{T_C, B_C\}$ *of P is such that T_C stands for the action precondition, goal, and* observable *variables in P, and $B_C(X)$ is the set of state variables that are* causally relevant *to X.*

The *causal decomposition* determines a larger number of subproblems, as subproblems are generated also for the observable variables X, but the beams $B_C(X)$ associated with these subproblems are smaller, as they contain only the state

variables that are *causally relevant* to X as opposed to the *relevant* variables. The *causal width* of a problem is defined in terms of the largest beam in the causal decomposition:

Definition 12. *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 new *causal decomposition* is what we call *Decoupled Causal Belief Tracking* or Decoupled CBT, which runs in time and space that are exponential in the causal width:

Definition 13. *Decoupled CBT is* flat belief tracking *applied independently to each of the problems P^C_X determined by the* causal decomposition *of P.*

Since causal width is never greater than width and is often much smaller, Decoupled CBT will run much faster than factored belief tracking in general. This, however, comes at a price, that we express using relational operators for *projections* and *joins*, exploiting that beliefs are tables or relations whose rows are states, and whose columns are variables.[3] The subproblem of P determined for a target variable X in the *causal* decomposition is denoted as P^C_X; i.e., $P^C_X = P_{B_C(X)}$.

Theorem 14. *Decoupled CBT is sound and runs in time and space that are exponential in $w_c(P)$ but it is not complete; i.e., 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^C_X 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 of Decoupled CBT is that the beliefs b_X associated with different target variables X are not independent. Indeed, a variable Z may appear in two beams of the causal decomposition, with some variables relevant to Z in one beam, and other variables relevant to Z in the other. In the factored decomposition this cannot happen, as all the variables that are relevant to Z will appear in all the beams that contain Z. 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 not independent.

6 Causal Belief Tracking

In *causal belief tracking*, the local beliefs are not tracked independently over each of the subproblems P^C_X of the causal decomposition; rather, the local beliefs are first progressed and filtered *independently*, but then are merged and projected back onto each of the beams, making them consistent with each other. This consistency operation is expressed using the projection and join operators:

Definition 15. Causal belief tracking *is the belief tracking algorithm that operates on the causal decomposition $C = \langle T_C, B_C \rangle$, setting the beliefs b^0_X at time 0 over each beam $B_C(X)$, $X \in T_C$, to the projection of the initial belief over the beam, and the successive beliefs b^{i+1}_X as:*

$$b^{i+1}_X = \textstyle\prod_{B_C(X)} \bowtie \{(b^i_Y)^o_a : Y \text{ in } T_C \text{ and relevant to } X\} \quad (3)$$

where $a = a_i$ and $o = o_i$ are the action and obs. at time i in the execution, and $(b^i_Y)^o_a$ is b^o_a from Eqs. 1–2 with $b = b^i_Y$.

Progressing and filtering the local beliefs in the causal decomposition is time and space exponential in the problem *causal width*, but the consistency operation captured by the join-project operation in (3) is time exponential in the number of state variables that are relevant to X. As a result:

Theorem 16. *CBT is* space *exponential in the problem* causal width*, and* time *exponential in the problem* width*.*

CBT is sound but still *not* complete. However, the range of problems for which CBT is complete, unlike Decoupled CBT, is large and meaningful enough, and it includes for example the three domains to be considered in the experimental section: Minesweeper, Battleship and Wumpus. We express the completeness conditions for CBT by introducing the notions of *memory variables* and *causally decomposable problems.* A *state variable* X is a *memory variable* in a problem P when the value X^k of the variable at any time point k can be *determined uniquely* from an observation of the value of the variable X^i at any other time point i, the actions done in the execution, and the initial belief state of the problem. Thus, *static variables* are memory variables, as much as variables that are *determined* (Section 3). For the first, $X^k = X^i$, while for the second, X^k is determined by the initial belief and the actions before time k. All the variables in *permutation domains* [Amir and Russell, 2003] are also memory variables.

Definition 17. *A problem P is* causally decomposable *when for every pair of beams $B_C(X)$ and $B_C(X')$ in the causal decomposition, then 1) the variables in the intersection of the two beams are all memory variables, or 2) there is a beam in the decomposition that includes both beams.*

Theorem 18. *CBT is always sound and it is complete for* causally decomposable *problems.*

7 Approximation: Beam Tracking

The last algorithm, *beam tracking* is also defined over the new *causal decomposition* but it is not aimed at being complete but rather efficient and effective. It combines the low complexity of Decoupled CBT, which is time and space exponential in the causal width of the problem, with a form of *local consistency* that approximates the *global consistency* enforced in CBT:

Definition 19. Beam tracking *is the belief tracking algorithm that operates on the causal decomposition $C = \langle T_C, B_C \rangle$, setting the beliefs b^0_X at time 0 over each beam $B_C(X)$, $X \in T_C$, to the projection of the initial belief over the beam, and setting the successive beliefs b^{i+1}_X in two steps. First, they are set to b^o_a for $b = b^i_X$, $a = a_i$, 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:*

$$b^{i+1}_X \;:=\; \Pi_{B_c(X)}(\, b^{i+1}_X \;\bowtie\; b^{i+1}_Y)\,. \quad (4)$$

[3] 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 $X = 1, Y = 1, Z = 2$.

The filtering captured by the iterative updates in Eq. 4 defines a form of *relational arc consistency* [Dechter and 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 is *sound* but *not complete*. In problems that satisfy the conditions in Theorem 18, however, the incompleteness is the sole result of replacing global consistency by local consistency.

8 Extensions

Before considering the experiments, we discuss briefly two simple but useful extensions of the language and the algorithms. The first involves *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 (e.g., Z true when $X = Y$ is known to be true). Variables defined in this way can then be used in action preconditions or goals, and the immediate causes of such variables are the state variables appearing in its definition. All the results above carry to domains with defined variables once they are added to the set of target variables in the factored and causal decompositions. The second extension is *state constraints*. In Battleship, for example, they are used to express the conditions under which the neighbor of a cell containing a ship, will contain the same ship. A state constraint represented by a formula C can be encoded by means of a dummy observable variable Y that is always observed to be true, and can be observed to be true only in states where C is true; i.e., $W_a(Y = true) = C$. For the *implementation*, however, it pays off to treat such constraints C as relations, and to include them in all the 'joins' over 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 (4) to

$$b_X^{i+1} := \Pi_{B_c(X)}(b_X^{i+1} \bowtie b_Y^{i+1} \bowtie C) \quad (5)$$

where C stands for 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 every 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 can be ignored.

9 Experimental Results

We have tested the beam tracking algorithm over three domains, Battleship, Minesweeper, and Wumpus, in combination with simple heuristics for selecting actions. Belief tracking in these domains is difficult [Kaye, 2000; Scott *et al.*, 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 simpler and fully-solvable version of Wumpus where the comparison with off-line and on-line contingent planners is feasible. The results have been obtained on a Xeon 'Woodcrest' 5140 CPU running at 2.33 GHz with 8GB of RAM.[4]

dim	policy	#ships	#torpedos	avg. time per decision	avg. time per 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 1: Battleship. Averages over 10,000 runs. Time is in seconds.

Battleship

The problem is encoded with 6 variables per cell:[5] $hit_{x,y}$ tells if a torpedo has been fired at the cell, $sz_{x,y}$ is the size of the ship occupying the cell (0 if no such ship), $hz_{x,y}$ encodes if ship is placed horizontally or vertically, $nhits_{x,y}$ is the number of hits for ship at cell, and $anc_{x,y}$ is the relative position of the ship on the cell. The observable variable is $water_{x,y}$ with sensor model given by $W_{fire(x,y)}(water_{x,y} = true) = (sz_{x,y} = 0)$. In addition, state constraints are used to describe how a ship at a given cell constrains the variables associated with neighboring cells. The goal of the problem is to achieve the equality $nhits_{x,y} = sz_{x,y}$ over the cells that may contain a ship. 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 dimensions.

Table 1 shows results for two policies: a *random* policy that fires at any 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* computed by beam tracking. 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 on average, matching quite closely the results of Silver and Veness [2010] obtained with a combination of UCT [Kocsis and Szepesvári, 2006] for action selection, and a particle filter [Doucet *et al.*, 2000] tuned to the domain for belief tracking. Their approach involves 65,000 simulations per action and takes in the order of 2 seconds per game, while ours takes 0.0096 seconds per game.

Minesweeper

We consider Minesweeper instances over $m \times n$ grids containing k randomly placed mines. The problem can be described 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 $D =$

[4] A real-time animation for Minesweeper can be found at https://www.youtube.com/watch?v=U98ow4n87RA.

[5] This is a rich encoding that allows to observe when a ship has been fully sunk. In the experiments, this observation was not used in order to compare with the work of Silver and Veness [2010].

dim	#mines	density	%win	#guess	avg. time per decision	avg. time per 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 2: Minesweeper. Averages over 1,000 runs. Time is in seconds.

$\{0, \ldots, 9\}$. There are two type of actions $open(x, y)$ and $flag(x, y)$ where the first has no precondition and the second $\neg mine_{x,y}$. The sensor model is given by formulas that specify the integer that receives the agent when opening a cell in terms of the value $mine_{x',y'}$ for the surrounding cells. The goal of the problem is to get the disjunction $flagged_{x,y} \vee opened_{x,y}$ for each cell (x, y), provided that opening a cell with a mine causes termination. The causal beams contain at most 9 boolean variables, even though the width of the problem is $3mn$.

Table 2 shows results for the three standard levels of the game and a much larger instance. As in Battleship, the greedy policy uses the beliefs computed by beam tracking, flagging or opening a cell when certain about its content, else selecting the cell with the most extreme probability, and flagging or opening it according to its likely content. The results shown in the table are competitive with those recently reported by Lin *et al.* [2012], which are achieved 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 with 10 mines, $74.4 \pm 0.5\%$ for the 16×16 with 40 mines, and $38.7 \pm 1.8\%$ for the 16×30 with 99 mines. The authors do not report times, but as in Battleship, the time required for selecting actions is likely to be orders-of-magnitude larger than the time taken by our algorithm.

Wumpus

An $m \times n$ instance of Wumpus [Russell and Norvig, 2009] is described with known state variables for the agent's position and orientation, and hidden boolean variables for each cell that tell whether there is a pit, a wumpus, or nothing at the cell. One more 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 6 while the problem width grows with the grid size.

Table 3 shows results for different grid sizes and number of pits and wumpus for a greedy policy based on a heuristic that returns the minimum cost of a 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. As the table shows, very large instances are solved in real time. Moreover, the unsolved instances were all proven to be unsolvable using the information gathered by the agent and a SAT solver.

We also considered a simpler version of Wumpus that has been used as a benchmark for testing off-line and on-line contingent planners. In this version, there are no pits, the agent is known to start at the bottom-left corner, the gold is known to be at the opposite corner, and a wumpus is known to be on the cell below or to the left of each diagonal cell, except for the first two cells. The problem has a full contingent solution for grids $n \times n$ for $n \geq 3$. Off-line planners have been shown to scale up to $n = 7$ [Albore *et al.*, 2009; To *et al.*, 2011], while on-line planners to $n = 20$ [Albore *et al.*, 2009; Brafman and Shani, 2012]. We tried our K-replanner [Bonet and Geffner, 2011] that is domain-independent, relies on a very effective form of belief representation based on literals and invariants, but is however less powerful than beam tracking. The K-replanner manages to scale up to $n = 40$. Beam tracking with the heuristic above yields solutions to much larger grids, e.g., $n > 100$, in real-time. Unfortunately, since it's not possible to try the other planners with the same action selection mechanism, it is not possible to say how much of the gap in performance is due to the belief representation, and how much to action selection. It must also be kept in mind that none of the planners handle action nondeterminism which poses no problem for beam tracking.

dim	#p/#w	%den	#decisions	%win	avg. time per decision	avg. time per 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
40×40	128 / 128	16.0	4,471,168	7.3	2.8E-3	12.7
50×50	512 / 512	40.9	7,492,503	0.1	1.3E-2	100.4

Table 3: Wumpus. Averages over 1,000 runs. Column '#p/#w' refers to number of hidden pits and wumpus, and '%den' to their density in the grid. Time is in seconds.

10 Summary

We have introduced a causal decomposition scheme in belief tracking for planning along with a new algorithm, causal belief tracking, that is *provably complete* for a broad range of domains. CBT is time exponential in the problem *width*, and space exponential in the often much smaller *causal width*. We then introduced a second algorithm, beam tracking, as a meaningful approximation of CBT that is both time and space exponential in the problem causal width. The empirical results in Battleship, Minesweeper, and Wumpus, where very large instances are solved in real-time with a greedy policy that is defined on top of these beliefs, suggest that the algorithm may be practical enough, managing to track beliefs effectively and efficiently over large deterministic and nondeterministic spaces.

Acknowledgements

This work was partially supported by EU FP7 Grant# 270019 (Spacebook) and MICINN CSD2010-00034 (Simulpast). We thank Gabriel Detoni for his Tewnta framework used to implement graphical interfaces for the three games and James Biagioni for his command-line `wumpuslite` simulator.

References

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

[Amir and Russell, 2003] E. Amir and S. Russell. Logical filtering. In *Proc. IJCAI*, pages 75–82, 2003.

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

[Bonet and Geffner, 2011] B. Bonet and H. Geffner. Planning under partial observability by classical replanning: Theory and experiments. In *Proc. IJCAI*, pages 1936–1941, 2011.

[Bonet and Geffner, 2012] B. Bonet and H. Geffner. Width and complexity of belief tracking in non-deterministic conformant and contingent planning. In *Proc. AAAI*, pages 1756–1762, 2012.

[Brafman and Shani, 2012] R. I. Brafman and G. Shani. A multi-path compilation approach to contingent planning. In *Proc. AAAI*, pages 9–15, 2012.

[Darwiche and Marquis, 2002] A. Darwiche and P. Marquis. A knowledge compilation map. *JAIR*, 17:229–264, 2002.

[Dechter and Beek, 1997] R. Dechter and P. Van Beek. Local and global relational consistency. *Theoretical Computer Science*, 173(1):283–308, 1997.

[Doucet *et al.*, 2000] A. Doucet, N. De Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proc. UAI*, pages 176–183, 2000.

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

[Kocsis and Szepesvári, 2006] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proc. ECML*, pages 282–293, 2006.

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

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

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

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

[Scott *et al.*, 2011] A. Scott, U. Stege, and I. Van Rooij. Minesweeper may not be NP-Complete but is Hard nonetheless. *Science+Business Media, LLC*, 33(4):5–17, 2011.

[Silver and Veness, 2010] D. Silver and J. Veness. Monte-carlo planning in large POMDPs. In *Proc. NIPS*, pages 2164–2172, 2010.

[To *et al.*, 2011] S. T. To, E. Pontelli, and Tran Cao Son. On the effectiveness of CNF and DNF representations in contingent planning. In *Proc. IJCAI*, pages 2033–2038, 2011.