Solving planning domains with polytree causal graphs is NP-complete

Omer Giménez *

Abstract

We show that solving planning domains on binary variables with polytree causal graph is NP-complete. This is in contrast to a polynomial-time algorithm of Domshlak and Brafman that solves these planning domains for polytree causal graphs of bounded indegree.

1 Introduction

It is well known that the planning problem (namely, the problem of obtaining a valid sequence of transformations that moves a system from an initial state to a goal state) is intractable in general [3]. However, it is widely believed that many real-life problems have a particular structure, and that by exploiting this structure general planners will be able to efficiently handle more meaningful problems.

One of the most fruitful tools researchers have been using to characterize structure in planning problems is the so called *causal graph* ([6]). In short, the causal graph of a problem instance is a graph that captures the degree of interdependence among the state variables of the problem.The causal graph has been used both as a tool for describing tractable subclasses of planning problems (e.g., [7], [2], [4]) and as a key property which algorithms that adress the general planning problem take into consideration [5].

In the present work we show that solving planning domains where the causal graph is a polytree (that is, the underlying undirected graph is acyclic) is NP-complete, even if we restrict to domains with binary variables and unary operators. This result closes the complexity gap that appears in [4], where it is shown that plan existence is NP-complete for planning domains with singly connected causal graphs, and that plan generation is polynomial for planning domains with polytree causal graphs of bounded indegree.

Additionally, it is known that solving unary operator planning problems on binary variables is essentially equivalent to solving dominance queries for binary-valued CP-nets (see [1]). Under this reformulation the causal graph becomes the CP-net, so the present work also shows that dominance testing for binary-valued polytree CP-nets is NP-complete.

*Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona. omer.gimenez@upc.edu

2 Definitions

In this section we define planning problems and causal graphs according to the SAS+ formalism, and we introduce a short-cut notation to describe unary operators on binary variables.

Let $\mathcal{V}$ be a set of (state) variables. The *domain* D_v of variable v is the set of values that v can take. A *(partial) state* S defined on the set $\mathcal{C}(S) \subseteq \mathcal{V}$ is a mapping of the variables $\mathcal{C}(S)$ onto values of their respective domains. When $\mathcal{C}(S) = \mathcal{V}$ we say that the state S is *total*. We write $S \subseteq S'$ when $\mathcal{C}(S) \subseteq \mathcal{C}(S')$ and both assignments coincide in $\mathcal{C}(S)$, and $S \oplus S'$ to denote the state defined on $\mathcal{C}(S) \cup \mathcal{C}(S')$ obtained by merging the assigments of S and S' but giving preference to S' for variables on $\mathcal{C}(S) \cap \mathcal{C}(S')$.

An *operator* α is a tuple of partial states $(\mathbf{prv}, \mathbf{pre}, \mathbf{post})$, where $\mathbf{prv}$ (prevail conditions), $\mathbf{pre}$ (pre-conditions), $\mathbf{post}$ (post-conditions) satisfy $\mathcal{C}(\mathbf{pre}) = \mathcal{C}(\mathbf{post})$ and $\mathcal{C}(\mathbf{prv}) \cap \mathcal{C}(\mathbf{pre}) = \emptyset$. An operator is *unary* when $|\mathcal{C}(\mathbf{pre})| = 1$. To apply an operator α onto a (total) state S we require that $\mathbf{prv} \subseteq S$ and $\mathbf{pre} \subseteq S$; when this holds, we define $\alpha(S)$ as $S \oplus \mathbf{post}$.

A planning domain instance P is a tuple $(\mathcal{V}, \mathcal{O}, I, G)$ where $\mathcal{V}$ is the set of variables, $\mathcal{O}$ is the set of *operators*, I is the (total) *initial state* and G is the (possibly partial) *goal state*. A *plan* π for P is a sequence of operators $\alpha_1, \alpha_2, \cdots, \alpha_t$ such that we are allowed to apply α_i onto state S_i for all $i \leq t$, where $S_i = \alpha_{i-1}(S_{i-1})$ and $S_1 = I$, the initial state. An *action* is a particular occurrence of an operator in a plan, and a plan is *valid* when $G \subseteq S_{t+1}$.

The *planning problem* is the problem of obtaining a valid plan π for a planning domain instance P. We may consider several variations on the problem, like obtaining optimal valid plans, or simply deciding whether a plan exists or not.

The *causal graph* of a problem $P = (\mathcal{V}, \mathcal{O}, I, G)$ is a directed graph that has $\mathcal{V}$ as the set of vertices and a directed edge from x to y if and only if there is an operator $\alpha = (\mathbf{prv}, \mathbf{pre}, \mathbf{post})$ in $\mathcal{O}$ such that $y \in \mathcal{C}(\mathbf{post})$ and $x \in \mathcal{C}(\mathbf{pre}) \cup \mathcal{C}(\mathbf{prv})$. Hence a directed edge from x to y means that we may need to take into account the value of x when considering operators that change y.

In the present work we restrict to binary domains (that is, $\mathcal{D}_v = \{0, 1\}$ for all variables v) and unary operators ($|\mathbf{post}| = 1$). Under these circumstances, and assuming that no operator has equal pre-condition and post-condition, the pre-condition of an operator α can be deduced from its post-condition, so we will simply write $\alpha = \langle \mathbf{prv}, \mathbf{post} \rangle$, or even $\alpha = \langle \mathbf{post} \rangle$ if $\mathbf{prv} = \emptyset$. In addition, we write post-conditions using the assignment notation *variable* $\leftarrow$ *value* to emphasize that post-conditions modify the state. For instance, operators $(\{x = 1, y = 1\}, \{z = 0\}, \{z = 1\})$ and $(\emptyset, \{z = 1\}, \{z = 0\})$ will be written $\langle \{x = 1, y = 1\}, z \leftarrow 1 \rangle$ and $\langle z \leftarrow 0 \rangle$.

3 Main result

We prove NP-hardness by showing a reduction between 3-CNF-SAT and our class of planning domains. As an example of the reduction, Figure 1 shows the causal graph of the planning domain P_F that corresponds to a formula F of three variables and three clauses. (The precise definition of P_F is given in Proposition 3.2.)

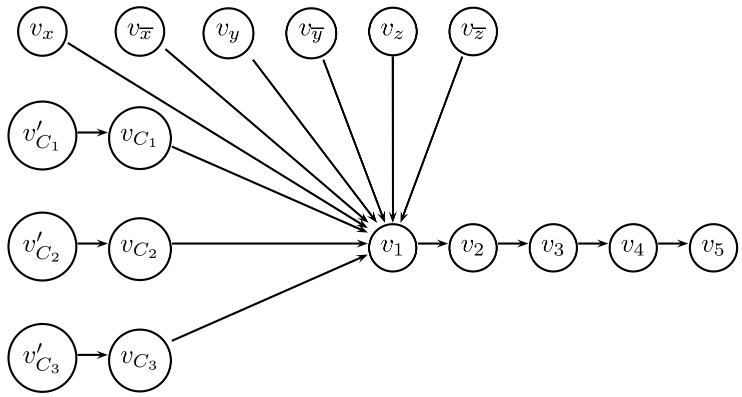

Figure 1: Causal graph of P_F when $F = C_1 \wedge C_2 \wedge C_3$ on three variables x, y, z.

Let us describe briefly the idea behind the reduction. The planning domain P_F has two different parts. The first part (state variables $v_x, v_{\overline{x}}, \ldots,$ $v_{C_1}, v'_{C_1}, \ldots,$ and v_1) depends on the formula F, and has the property that a plan may change the value of v_1 from 0 to 1 as many times as the number of clauses of F that a truth assignment can satisfy. However, this condition on v_1 can not be stated as a planning domain goal. We overcome this difficulty by introducing a gadget (state variables $v_1, v_2, \ldots, v_t$) that translates it to a regular planning domain goal.

We describe this last part. Let P be the planning domain $(\mathcal{V}, \mathcal{O}, I, G)$ where $\mathcal{V}$ is the set of variables $\{v_1, \ldots, v_{2k-1}\}$, and $\mathcal{O}$ is the set of $4k-2$ operators $\{\alpha_1, \ldots, \alpha_{2k-1}, \beta_1, \ldots, \beta_{2k-1}\}$. Operators α_1 and β_1 are defined as $\langle v_1 \leftarrow 0 \rangle$ and $\langle v_1 \leftarrow 1 \rangle$; for $i > 1$, operators α_i and β_i are respectively $\langle \{v_{i-1} = 0\}, v_i \leftarrow 0 \rangle$ and $\langle \{v_{i-1} = 1\}, v_i \leftarrow 1 \rangle$. All variables are 0 in the initial state I, and the goal state G is $v_i = 0$ when i is even, $v_i = 1$ when i is odd.

Lemma 3.1. *Any valid plan for the planning domain P changes at least k times the variable v_1 from 0 to 1. There is a valid plan that achieves this minimum.*

Proof. Let A_i be the sequence of actions $\alpha_1, \ldots, \alpha_i$, and let B_i be the sequence of actions $\beta_1, \ldots, \beta_i$. It is easy to check that the plan B_{2k-1}, A_{2k-2},B_{2k-3} , ..., B_3,A_2,B_1 is valid: after finishing a sequence of actions A_i or B_i, the variable v_i is in its goal state (0 if i is even, 1 if i is odd). Subsequent actions in the plan do not modify v_i, so the variable remains in its goal state until the end. The action β_1 appears k times in the plan, thus v_1 changes k times from state 0 to 1.

We proceed to show that k is the minimum. Consider a valid plan π, and let λ_i be the number of actions α_i and β_i that appear in π. (That is, λ_i is the number of times that variable v_i changes value, either from 0 to 1 or from 1 to 0. Note that the number of actions β_i has to be either equal or exactly one more than the number of actions α_i.) We will show that $\lambda_{i-1} > \lambda_i$. Since λ_{2k-1} has to be at least one, $\lambda_{i-1} > \lambda_i$ implies that $\lambda_1 \geq 2k-1$. In consequence, there are at least k actions β_i in plan π, finishing the proof.

We show that $\lambda_{i-1} > \lambda_i$ for valid plans. To begin with, let π be any plan (not necessarily a valid one) and consider only the subsequence made out of actions α_i and β_i in π. It starts with β_i (since the initial state is

$v_i = 0$), and the same action can not appear twice consecutively in the sequence. Thus this sequence alternates β_i and α_i. Moreover, since β_i (for $i > 1$) has $v_{i-1} = 1$ as prevail condition, and α_i has $v_{i-1} = 0$, there must be at least one action α_{i-1} in the plan π betweeen any two actions β_i and α_i. For the same reason we must have at least one action β_{i-1} between any two actions α_i and β_i, and an action β_{i-1} before the first action β_i. This shows that, in any plan π, not necessarily valid, we have $\lambda_{i-1} \geq \lambda_i$. If, in addition, π is valid, we require an extra action: when v_i changes state for the last time and attains its goal state, we have that $v_{i-1} = v_i$, so v_{i-1} is not in its goal state by parity. Hence a valid plan must have an extra action α_{i-1} or β_{i-1} after all occurrences of α_i and β_i. Thus $\lambda_{i-1} > \lambda_i$ for valid plans. □

Proposition 3.2. *3-*Sat *reduces to plan existance for planning domains on binary variables with a polytree causal graph.*

Proof. Let F be a CNF formula with k clauses and n variables. We produce a planning domain P_F on $2n + 4k - 1$ state variables and $2n + 14k - 3$ operators. The planning problem has two state variables v_x and $v_{\overline{x}}$ for every variable x in F, two state variables v_C and v'_C for every clause C in F, and $2k - 1$ additional variables $v_1, \ldots, v_{2k-1}$. All variables are 0 in the initial state. The (partial) goal state is $v_i = 0$ when i is even, $v_i = 1$ when i is odd, like in problem P of Lemma 3.1. The operators are:

(1.) Operators $\langle v_x \leftarrow 1\rangle$ and $\langle v_{\overline{x}} \leftarrow 1\rangle$ for every variable x of F.

(2.) Operators $\langle v'_C \leftarrow 1\rangle$, $\langle\{v'_C = 0\}, v_C \leftarrow 1\rangle$ and $\langle\{v'_C = 1\}, v_C \leftarrow 0\rangle$ for every clause C of F.

(3.) There are 7 operators for every clause C, one for each of the 7 different partial assignments that satisfy C. The post-condition of these operators is $v_1 \leftarrow 1$, and they all have 7 prevail conditions: the condition $v_C = 1$, and six conditions to ensure that the values of the state variables v_x and $v_{\overline{x}}$ associated to a variable x that appears in C are in agreement with the partial assignment.

For example, the operator related to the clause $C = x \vee \overline{y} \vee \overline{z}$ and the satisfying partial assigment $\{x = 0, y = 0, z = 1\}$ is

$$\langle\{v_x = 0, v_{\overline{x}} = 1, v_y = 0, v_{\overline{y}} = 1, v_z = 1, v_{\overline{z}} = 0, v_C = 1\}, v_1 \leftarrow 1\rangle$$

(4.) An operator $\langle\{v_C = 0 \mid \forall C\}, v_1 \leftarrow 0\rangle$.

(5.) Operators $\alpha_i = \langle\{v_{i-1} = 0\}, v_i \leftarrow 0\rangle$ and $\beta_i = \langle\{v_{i-1} = 1\}, v_i \leftarrow 1\rangle$ for $2 \leq i \leq 2k - 1$. (That is, the same operators that in problem P but for α_1 and β_1.)

We note some simple facts about problem P_F. For any variable x, state variables v_x and $v_{\overline{x}}$ in P_F start at 0, and by using the actions in (1.), they can change into 1, but they can not go back to 0. In particular, a plan π can not reach both partial states $\{v_x = 1, v_{\overline{x}} = 0\}$ and $\{v_x = 0, v_{\overline{x}} = 1\}$ during the course of its execution.

Similarly, if C is a clause of F, the state variable v_C can change from 0 to 1 and, by first changing v'_C into 1, v_C can go back to 0. No further changes are possible, since no action brings back v'_C to 0.

Now we interpret actions in (3.) and (4.), which are the only actions that affect v_1. To change v_1 from 0 to 1 we need to apply one of the actions of (3.), thus we require $v_C = 1$ for a clause C. But the only way to bring back v_1 to 0 is applying the action (4.), so that $v_C = 0$. We deduce that every time that v_1 changes its value from 0 to 1 and then back to 0 in the plan π, at least one of the k state variables v_C is *used up*, in the sense that v_C has been brought from 0 to 1 and then back to 0, and cannot be used again for the same purpose.

We show that F is in 3-SAT if and only if there is a valid plan for problem P_F. Let σ be a truth assignment that satisfies F. By Lemma 3.1 we can extend a plan π' that switches variable v_1 from 0 to 1 at least k times to a plan π that sets all variables v_i to their goal values. The plan π' starts by setting the all state variables v_x and $v_{\overline{x}}$ in correspondence with the truth assignment σ using the actions of (1.). Then, for every of the k state variables v_C, we set $v_C = 1$, we apply the action of (3.) that corresponds to σ restricted to the variables of clause C, and we move back v_C to 0 so that we can apply the action (4.). By repeating this process for all clauses C of F we are switching the state variable v_1 exactly k times from 0 to 1.

We show the converse, namely, that the existence of a valid plan π in P_F implies that F is satisfiable. By Lemma 3.1 the state variable v_1 has to change from 0 to 1 at least k times. This implies that k actions of (3.), all of them corresponding to different clauses, have been used to move v_1 from 0 to 1. Hence we can define a satisfying assigment σ by setting $\sigma(x) = 1$ if the partial states $\{v_x = 1, v_{\overline{x}} = 0\}$ appears during the execution of π, and $\sigma(x) = 0$ otherwise. $\square$

Theorem 3.3. *Plan existence for planning problems with a polytree causal graph is NP-complete.*

Proof. Due to Proposition 3.2 we only need to show that the problem is NP. But in [4] it is shown that this holds in the more general setting of planning problems with causal graphs where each component is singly connected. Their proof uses the non-trivial result that solvable planning problems on binary variables with a singly connected causal graph have plans of polynomial length. (This is not true for non-binary variables, or unrestricted causal graphs.)

$\square$

4 Concluding remarks

The given reduction constructs a planning domain with a partial goal state. If desired, we can make the goal state total by adding the restrictions $v_x = 1, v_{\overline{x}} = 1$ and $v_C = 1, v'_C = 0$ for all variables x and clauses C. On the other hand, the author has found no way to avoid an operator like the one in (4.), with an unbounded number of prevail conditions. Hence we should not rule out the existence of a polynomial-time algorithm that solves polytree planning domains where all operators have prevail conditions bounded by a constant.

Planning domains with unary operators on binary variables can be formulated as dominance queries in binary CP-nets, and the translation is polynomial-size preserving provided two technical conditions (see [1] for the details): we must allow partially specified CPTs in the CP-net, and no two operators with opposing post-conditions may share the same prevail conditions in the planning domain. Clearly the given reduction satisfies this last condition (actions α_1 and β_1 in problem P do not satisfy the last requirement, but they are not present in problem P_F) hence dominance testing in binary polytree CP-nets (with partially specified CPTs) is NP-complete.

Brafman and Domshlak show in [4] that, if we restrict to binary variables and unary operators, we can generate valid plans in time roughly $O(|\mathcal{V}|^{2\kappa})$ for planning domains where the causal graph is a polytree with indegree bounded by κ. The same authors show in [2] how to solve in time roughly $O(|V|^{\omega\delta})$ planning domains with local depth δ and causal graphs of tree-width ω. The planning domains of our reduction have $\kappa = 2n + k$, $\omega = 1$ and $\delta = 2k - 1$, where n and k stand for the number of variables and clauses of the 3-CNF formula. This vindicates the fact that the algorithms are exponential in, respectively, κ and δ, so that, unless $P = NP$, we cannot hope to improve them in a significant way for polytree planning domains.

Acknowledgements

The author is in debt to Carmel Domshlak and Anders Jonsson for useful suggestions and careful reading of this paper.

References

[1] Boutilier, C., Brafman, R., Domshlak, C., Hoos, H. and Poole, C. (2004). CP-nets: A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements. *Journal of Artificial Intelligence Research* 21:135–191.

[2] Brafman, R., and Domshlak, C. (2006). Factored Planning: How, When, and When Not. *Proceddings of AAAI'06*.

[3] Chapman, D. (1987). Planning for Conjunctive Goals. *Artificial Intelligence* 32(3):333–377.

[4] Domshlak, C., and Brafman, R. (2003). Structure and Complexity in Planning with Unary Operators. *Journal of Artificial Intelligence Research* 18:315–349.

[5] Helmert, M. (2004). A planning heuristic based on causal graph analysis. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 161–170.

[6] Knoblock, C.A. (1994). Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

[7] Williams, B. and Nayak, P. (1997). A reactive planner for Model-based Execution. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence.*