

Causal Graphs and Structurally Restricted Planning

Hubie Chen

Dept. de Tecnologies de la Informació i les Comunicacions
Universitat Pompeu Fabra
Passeig de Circumval·lació, 8
08003 Barcelona, Spain
hubie.chen@upf.edu

Omer Giménez

Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Jordi Girona, 1-3
08034 Barcelona, Spain
omer.gimenez@upc.edu

Abstract

The causal graph is a directed graph that describes the variable dependencies present in a planning instance. A number of papers have studied the causal graph in both practical and theoretical settings. In this work, we systematically study the complexity of planning restricted by the causal graph. In particular, any set of causal graphs gives rise to a subcase of the planning problem. We give a complete classification theorem on causal graphs, showing that a set of graphs is either polynomial-time tractable, or is not polynomial-time tractable unless an established complexity-theoretic assumption fails; our theorem describes which graph sets correspond to each of the two cases. We also give a classification theorem for the case of reversible planning, and discuss the general direction of structurally restricted planning.

Introduction

Background. It is well-acknowledged that real-world planning problems exhibit structure that can be exploited to solve them more efficiently than a worst-case analysis would suggest. Understanding how to exploit such structure, either explicitly or implicitly, has been the focus of much planning research, both from the practical and theoretical viewpoints. In this vein, a feature of planning instances that has attracted significant attention is the *causal graph*, which is a directed graph whose edges describe variable dependencies (Knoblock 1994). For example, the causal graph has been used as the basis for domain-independent heuristics in the Fast Downward planning system (Helmert 2006) and to factor planning problems (Knoblock 1994; Brafman and Domshlak 2006).

A natural research issue is to study the complexity of planning restricted to different types of causal graphs, that is: which causal graphs support polynomial-time tractable planning and which do not? This issue was taken up by a number of papers, of which we describe a sampling. Polynomial-time algorithms for classes of problems whose definition involves restricting the causal graph were given by Jonsson and Bäckström (1998), Brafman and Domshlak (2003), and Jonsson (2007). The intractability of concrete families of causal graphs was shown by Domshlak

and Dinitz (2001), Helmert (2004), and Giménez and Jonsson (2008). Complexity results based on structural features of the causal graph were given for sequentially-optimal planning by Katz and Domshlak (2007) and planning with bounded local depth by Brafman and Domshlak (2006).

Contributions. In this paper, we give a complete classification of the complexity of planning under causal graph restrictions. That is, we give a theorem (Theorem 4) that establishes, for any set of causal graphs \mathcal{C} , either that planning over \mathcal{C} is polynomial-time tractable, or that it is not (unless an established complexity-theoretic assumption fails); the theorem also describes the graph sets corresponding to each of the two cases. More specifically, this theorem shows that, for any set of causal graphs \mathcal{C} , if there exists a constant k bounding the size of connected components in the undirected graphs induced by \mathcal{C} , then plan generation over \mathcal{C} is polynomial-time solvable; otherwise, plan existence over \mathcal{C} is not polynomial-time decidable. Note that we do not place any restrictions on the sizes of variable domains. This theorem subsumes and places into a unified framework some complexity results established by Domshlak and Dinitz (2001), Helmert (2004) and Giménez (2008).

This classification theorem closes the question of which tractable classes may be explained to be tractable solely via the causal graph. It is quite possible, however, that causal graph restrictions may be combined with other problem features to present tractability results, for instance, bounded domain size. Our second main theorem (Theorem 23) addresses a feature which we refer to as reversibility, which is present in many benchmark domains. For planning problems that are reversible, we also obtain a complete complexity classification of all sets of causal graphs. For this second classification, the property that determines tractability is the existence of a constant bound on the size of strongly connected components. The intractability results of our classification theorems were inspired by and are methodologically based on the proof of Grohe’s theorem on constraint satisfaction (Grohe 2007).

After presenting our two classification theorems, we define and begin to study an object associated to each planning instance which we call the *interaction network*. Roughly speaking, the interaction network of an instance is the object obtained by “removing” the values from the operators. This

object captures relationships among the variables in a finer way than the causal graph does; indeed, the causal graph of an instance can be derived from its interaction network.

This definition draws, by way of analogy, from the work on structurally restricted constraint satisfaction (Gottlob, Leone, and Scarcello 2000; Grohe 2007), and we believe that its study could be construed as a program of structurally restricted planning. We initiate investigation of the interaction network by presenting a tractability result in this framework whose algorithm appears to be qualitatively different than those given for causal graphs. We also present the research problem of obtaining a classification of interaction networks; we look forward to future research along these lines.

Preliminaries

We use the notation $[n]$ and $[i, n]$ to respectively refer to the sets $\{1, \dots, n\}$ and $\{i, \dots, n\}$. If $z = (z_1, \dots, z_k)$ is a tuple, we write $\pi_i(z)$ to denote the i -th component z_i of z .

Planning

An instance of the planning problem is a tuple $\Pi = (V, \text{init}, \text{goal}, A)$ whose components are described as follows.

- V is a finite set of variables, where each variable $v \in V$ has an associated finite domain $D(v)$. Note that variables are not necessarily propositional, that is, $D(v)$ may have any finite size. A *state* is a mapping s defined on the variables V such that $s(v) \in D(v)$ for all $v \in V$. A *partial state* is a mapping p defined on a subset $\text{vars}(p)$ of the variables V such that for all $v \in \text{vars}(p)$, it holds that $p(v) \in D(v)$.
- init is a state called the *initial state*.
- goal is a partial state.
- A is a set of *operators*; each operator $a \in A$ consists of a *precondition* $\text{pre}(a)$, which is a partial state, as well as a *postcondition* $\text{post}(a)$, also a partial state. We sometimes denote an operator a by $\langle \text{pre}(a); \text{post}(a) \rangle$.

Note that when s is a state or partial state, and W is a subset of the variable set V , we will use $(s \upharpoonright W)$ to denote the partial state resulting from restricting s to W . We say that a state s is a *goal state* if $(s \upharpoonright \text{vars}(\text{goal})) = \text{goal}$.

We define a *plan* (for an instance Π) to be a sequence of operators $P = a_1, \dots, a_n$. Starting from a state s , we define the state resulting from s by applying a plan P , denoted by $s[P]$, inductively as follows. For the empty plan $P = \epsilon$, we define $s[\epsilon] = s$. For non-empty plans P , denoting $P = P', a$, we define $s[P', a]$ as follows.

- If $(s[P'] \upharpoonright \text{vars}(\text{pre}(a))) \neq \text{pre}(a)$ (that is, the precondition of a does not hold in $s[P']$) then $s[P', a] = s[P']$.
- Otherwise, $s[P', a]$ is the state equal to $\text{post}(a)$ on variables $v \in \text{vars}(\text{post}(a))$, and equal to $s[P']$ on variables $v \in V \setminus \text{vars}(\text{post}(a))$.

We say that a state s is *reachable* (in an instance Π) if there exists a plan P such that $s = \text{init}[P]$. A plan P is a *solution plan* if $\text{init}[P]$ is a goal state.

We are concerned with two computational problems. The first is *plan existence*: given an instance $\Pi = (V, \text{init}, \text{goal}, A)$, decide if there exists a solution plan. The second is *plan generation*: given an instance $\Pi = (V, \text{init}, \text{goal}, A)$, output a solution plan if one exists.

Graphs

A *directed graph* is a pair (V, E) where V is the *vertex set* and $E \subseteq V^2$ is the *edge set*. An *undirected graph* is a pair (V, E) where V is the *vertex set* and $E \subseteq \binom{V}{2}$ is the *edge set*. By the notation $\binom{V}{k}$, we denote the set of all size k subsets of V . (We adhere to the convention that $\binom{V}{k}$ is the empty set in the case that V has size strictly smaller than k .) When G is a graph, we will sometimes use the notation V_G to refer to its vertex set, and E_G to refer to its edge set.

For a directed graph $C = (V, E)$, we use $U(C)$ to denote the undirected graph (V, E') where $E' = \{\{v, v'\} \mid v \neq v' \text{ and } (v, v') \in E\}$. That is, $U(C)$ is the undirected graph naturally induced by C by “forgetting” orientations of edges.

A *connected component* of an undirected graph G is a set of vertices S such that, for any two vertices $s, s' \in S$, there exists an undirected path from s to s' . A *strongly connected component* of a directed graph C is a set of vertices S such that, for any two vertices $s, s' \in S$, there exists a directed path from s to s' .

A *clique* of an undirected graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that for any two distinct elements $s, s' \in S$, it holds that $\{s, s'\} \in E$.

The *causal graph* of a planning instance $\Pi = (V, \text{init}, \text{goal}, A)$ is the directed graph (V, E) where E contains all pairs (u, v) such that $u \neq v$, and there exists an operator $a \in A$ such that $u \in \text{vars}(\text{pre}(a)) \cup \text{vars}(\text{post}(a))$ and $v \in \text{vars}(\text{post}(a))$.

Parameterized Complexity

The intractability results of this paper are established using the framework of parameterized complexity. Here, we present the elements of parameterized complexity that will be used in the paper; we refer the reader to the book by Flum and Grohe (2006) for more information. Note, however, that the reductions that we give are presented in a way that does not make any reference to parameterized complexity, and can be read and understood in a relatively self-contained fashion.

A *parameterization* of a decision problem is a polynomial-time computable mapping κ that maps each instance I of the problem to a *parameter* $\kappa(I)$. A *parameterized problem* is a decision problem A along with a parameterization of A . As an example, consider the CLIQUE problem: given a pair (G, k) where G is an undirected graph, decide if G has a clique of size k . The *parameterized clique problem*, denoted p-CLIQUE is the CLIQUE problem with respect to the parameterization $\kappa(G, k) = k$.

A parameterized problem is *fixed-parameter tractable* if there exists a computable function f mapping into the natural numbers, a constant c , and an algorithm that decides an instance x of the problem in time $f(\kappa(x))|x|^c$. This notion may be viewed as a relaxation of polynomial-time

tractability; while for each fixed value of the parameter the running time must be polynomial with degree c , the running time may have an exponential (or worse) dependence on the parameter. The class FPT is defined to contain all parameterized problems that are fixed-parameter tractable. A non-uniform extension of this class is defined as follows. A parameterized problem is *non-uniformly fixed-parameter tractable* if there is a constant c such that, for every parameter value k , there is an algorithm that decides the instances x of the problem having $\kappa(x) = k$ in time $O(|x|^c)$. The class nu-FPT is defined to contain all parameterized problems that are non-uniformly fixed-parameter tractable.

A hierarchy of parameterized complexity classes, the so-called W-hierarchy, has been defined. The first class in this hierarchy is called W[1], and can be thought of as an parameterized analog of NP. It is widely believed that W[1] is not contained in FPT nor nu-FPT. A completeness theory for W[1] has been developed and the following is a fundamental result.

Theorem 1. (Downey and Fellows 1995) *The parameterized problem p-CLIQUE is W[1]-complete under fpt-reductions.*

For our purposes here, it is not necessary to define fpt-reductions; instead, we refer the reader to a reference (Flum and Grohe 2006).

Let us say that a parameterized problem (A, κ) *non-uniformly reduces into* a decision problem B if there exists a constant c such that for every parameter value k there is a reduction from the instances x of A having $\kappa(x) = k$ to the problem B running in time $O(|x|^c)$. By a reduction, we simply mean a mapping that sends an instance x of A to an equivalent instance y of B where by *equivalent*, we simply mean that $x \in A$ if and only if $y \in B$. In the context of discussing reductions, we will use the term “equivalent” in this fashion.

The type of reduction just introduced will be our tool for demonstrating hardness results. In particular, we will use the following theorems.

Theorem 2. *Let (A, κ) be a parameterized problem and let B be a decision problem. If (A, κ) non-uniformly reduces into B and B is polynomial-time decidable, then (A, κ) is in nu-FPT.*

Proof. Let c be the constant from the reduction, and let d be a constant such that an instance x of B can be decided in time $O(|x|^d)$. The problem (A, κ) is non-uniformly fixed-parameter tractable with respect to $c' = cd$: for any k , this is via the algorithm that, on an instance x with $\kappa(x) = k$, invokes the reduction to obtain an instance y of B , and then invokes the decision algorithm for B on y . The instance y has size that is $O(|x|^c)$, and hence the decision algorithm requires time at most $O((|x|^c)^d) = O(|x|^{cd}) = O(|x|^{c'})$. \square

Theorem 3. *If the parameterized problem p-CLIQUE non-uniformly reduces into B and B is polynomial-time decidable, then $W[1] \subseteq \text{nu-FPT}$.*

Proof. (idea) By Theorem 2, it follows that p-CLIQUE is in nu-FPT. Since nu-FPT is closed under fpt-reductions, it

follows from Theorem 1 that every parameterized problem in W[1] is in nu-FPT. \square

We will use Theorem 3 to give evidence that problems are not polynomial-time decidable, by exhibiting reductions from p-CLIQUE. Thus, our intractability results are relative to the complexity-theoretic assumption $W[1] \not\subseteq \text{nu-FPT}$. This is an assumption that is widely believed to hold; we remark that it is the same assumption relative to which (a version of) Grohe’s theorem (Grohe 2007) is proved.

Classification of Causal Graphs

For a set \mathcal{C} of directed graphs, define $\text{PlanExist}(\mathcal{C})$ to be the problem of deciding, given a planning instance Π whose causal graph is in \mathcal{C} , whether or not a solution plan exists. Similarly, $\text{PlanGen}(\mathcal{C})$ is defined as the problem of outputting, given a planning instance Π whose causal graph is in \mathcal{C} , a solution plan, if one exists. In other words, $\text{PlanExist}(\mathcal{C})$ and $\text{PlanGen}(\mathcal{C})$ are the plan existence and generation problems restricted to instances whose causal graphs are elements of \mathcal{C} .

For an undirected graph G , we define the *connected component size* of G , denoted $\text{cc-size}(G)$, to be the size of the largest connected component of G . For a directed graph C , we define $\text{cc-size}(C)$ to be $\text{cc-size}(U(C))$; recall that $U(C)$ is defined as the undirected graph induced by C .

For a set of directed graphs \mathcal{C} , say that $\text{cc-size}(\mathcal{C})$ is *bounded* if there exists a constant k such that for all $C \in \mathcal{C}$, it holds that $\text{cc-size}(C) \leq k$; otherwise, we say that $\text{cc-size}(\mathcal{C})$ is *unbounded*.

Theorem 4. *Let \mathcal{C} be a set of directed graphs.*

- *If $\text{cc-size}(\mathcal{C})$ is bounded, then the problem $\text{PlanGen}(\mathcal{C})$ is polynomial-time solvable;*
- *otherwise, the problem $\text{PlanExist}(\mathcal{C})$ is not polynomial-time decidable (unless $W[1] \subseteq \text{nu-FPT}$).*

For a directed graph C , we define the *pre-degree* of a vertex v , denoted by $\text{pre-degree}(v)$, to be the number of vertices other than v that have a (directed) path to v . We define the *pre-degree* of the graph C , denoted by $\text{pre-degree}(C)$, to be the maximum value of $\text{pre-degree}(v)$ over all vertices v of C . We say that a set of directed graphs \mathcal{C} has *bounded pre-degree* if there exists a constant k such that $\text{pre-degree}(C) \leq k$ for all $C \in \mathcal{C}$; otherwise, we say that \mathcal{C} has *unbounded pre-degree*.

Discussion: why parameterized complexity? The reader may at this point ask: why are our classification theorems proved relative to a complexity-theoretic assumption that comes from parameterized complexity, and not simply the assumption that P does not equal NP? In response to this question, we point out that there exist sets of directed graphs \mathcal{C} such that $\text{PlanExist}(\mathcal{C})$ is *NP-intermediate*, by which we mean that $\text{PlanExist}(\mathcal{C})$ is not in P nor NP-hard assuming that P does not equal NP. This can be proved by the technique presented in Sections 4 and 5 of the paper by Bodirsky and Grohe (2008); in particular, Theorem 5 in that paper gives essentially the result that we desire, but is phrased

in terms of the constraint satisfaction problem. Such NP-intermediate problems justify the use of the finer-grained parameterized complexity theory; we cannot prove that they are NP-hard nor that they are in P without proving that P equals NP!

Tractability

Theorem 5. *There is an algorithm that solves instances of problem PlanGen(C), where C is a set of directed graphs with connected component size bounded by k , in time $O(n^{k+3})$, where n is the total size of the input planning problem.*

The idea behind the algorithm is to solve each connected component independently.

Intractability: Unbounded pre-degree

Theorem 6. *There is a polynomial-time algorithm R that, given an instance (G, k) of the Clique problem and a directed graph C with $\text{pre-degree}(C) \geq k$, produces an equivalent instance $R(G, k, C)$ of PlanExist(C) in time polynomial in $|G|$, $|C|$ and k .*

Recall that by an *equivalent instance*, we mean an instance that is a “yes” instance of PlanExist(C) if and only if the given (G, k) is a “yes” instance of the Clique problem.

Before giving the formal description of the reduction, we give an intuitive description. The variable set of the created planning instance will be $\{v_1, \dots, v_k, d\}$. Each variable v_i has the opportunity to commit to a vertex in V_G ; once it does this, the vertex committed to is stored in the first component of v_i . The variables can also propagate messages; messages are held in the second components of variables. The variable d checks, for each edge $\{v_i, v_j\}$, that the variables v_i, v_j committed to adjacent vertices of G .

Reduction 1 (Unbounded pre-degree). The planning instance $R(G, k, C) = (V, \text{init}, \text{goal}, A)$ is defined as follows. Let $d \in V_C$ be a vertex with $\text{pre-degree}(d) \geq k$, and let $v_1, \dots, v_k \in V_C$ be vertices (distinct from d and each other) such that, for any $v \in \{v_1, \dots, v_k\}$, there is a directed path from v to d in C that only goes through vertices of $\{v_1, \dots, v_k\}$. Under these conditions, there exists a subgraph T of C , defined on vertices $\{v_1, \dots, v_d, d\}$, that it is a directed tree where all edges point to d .

The variable set is $V = \{v_1, \dots, v_k, d\}$. The domains of the variables are

- $D(v_i) = (V_G \cup \{\text{start}\}) \times (\{(v_j \sim w) \mid j \in [k], w \in V_G\} \cup \{\text{blank}\})$ for $i \in [k]$, and
- $D(d) = \{(\text{checked } i) \mid i \in [0, M]\} \cup \{(\text{checking } i, a_i \sim w) \mid i \in [M], w \in V_G\}$, where $M = \binom{k}{2}$.

The initial state init is $\text{init}(v_i) = (\text{start}, \text{blank})$ for $i \in [k]$, and $\text{init}(d) = (\text{checked } 0)$.

The goal state goal is defined only on d and has $\text{goal}(d) = (\text{checked } M)$.

Let $v, v' \in \{v_1, \dots, v_k\}$, $w \in V_G$ and $(x, y), (x', y') \in D(v)$. Then, the operators of A changing v are

- $\text{commit}(v, w, y) = \langle v = (\text{start}, y); v = (w, y) \rangle$,

- $\text{initmessage}(v, w, y) = \langle v = (w, y); v = (w, (v \sim w)) \rangle$, and
- $\text{passmessage}(v, x, y, v', x', y') = \langle v = (x, y), v' = (x', y'); v' = (x', y) \rangle$ if $(v, v') \in E_T$.

Let $M = \binom{k}{2}$ and let $e_1 = \{a_1, b_1\}, \dots, e_M = \{a_M, b_M\}$ be an enumeration of all 2-element subsets of $\{v_1, \dots, v_k\}$, where we assume that $a_i < b_i$ for all $i \in [M]$ with respect to any fixed ordering of $\{v_1, \dots, v_k\}$. Let $i \in [M]$, let $w, w' \in V_G$, and let $v \in \{v_1, \dots, v_k\}$ such that $(v, d) \in E_T$. Then, the operators of A changing d are

- $\text{recorda}(i, v, x, w) = \langle d = (\text{checked } i-1), v = (x, (a_i \sim w)); d = (\text{checking } i, a_i \sim w) \rangle$, and
- $\text{recordb}(i, v, x, w, w') = \langle d = (\text{checking } i, a_i \sim w), v = (x, (b_i \sim w')); d = (\text{checked } i) \rangle$ if $(w, w') \in E_G$.

It is straightforward to verify that the causal graph of this instance $R(G, k, C)$ is a subgraph of C . By augmenting it with extra variables and unusable operators, a new instance whose causal graph is exactly C may be readily computed from $R(G, k, C)$. We omit the details.

Proposition 7. *Suppose that (G, k, C) is a triple on which the algorithm R is defined. If G has a clique of size k , then the planning instance $R(G, k, C)$ is solvable.*

To prove the converse of Proposition 7 we introduce the following notation. If s, s' are total states, we write $s' \leq s$ to denote that there exists some plan P , possibly empty, such that $s'[P] = s$. Finally, if $v, v' \in \{v_1, \dots, v_k\}$, we write $v' \leq v$ if the unique path in T from v' to d passes through v .

Lemma 8 (Commitments are hard). *Let s, s' be reachable states such that $s' \leq s$ and $\pi_1(s'(v)) = w$ for some $v \in \{v_1, \dots, v_k\}$, $w \in V_G$. Then $\pi_1(s(v)) = w$.*

Proof. The only operators a that modify the first component of v are commit operators, and $\pi_1(\text{pre}(a)(v)) = \text{start}$. Hence we cannot apply any such operator a in state s' . By induction, the same holds for all s such that $s' \leq s$. \square

Lemma 9 (Messages are truthful). *Let s be a reachable state such that $\pi_2(s(v)) = (v' \sim w)$ for some $v, v' \in \{v_1, \dots, v_k\}$ and $w \in V_G$. Then $v' \leq v$ and $\pi_1(s(v')) = w$.*

Proof. Let P be a plan such that $\text{init}[P] = s$, and let a be the first operator in P such that $\pi_2(\text{post}(a)(v)) = (v' \sim w)$. Let s' be the state before applying operator a during the execution of plan P . Note that $s' \leq s$.

We prove the lemma by induction on the number of ancestors of v in T , that is, on the number of vertices v_0 such that $v_0 \leq v$.

If v has no ancestor, then a must be one of the operators $\text{initmessage}(v, w, \cdot)$. This implies that $v = v'$ and, by the pre-condition of a , $\pi_1(s'(v)) = w$. Thus, by Lemma 8, we obtain $\pi_1(s(v)) = w$, which proves the base case.

Assume now that v has some ancestors in T . If a is some operator $\text{initmessage}(v, w, \cdot)$, then we apply the same reasoning as in the previous case. Assume now that a is one of the other possible operators $\text{passmessage}(v_0, \cdot, (v' \sim w), v, \cdot, \cdot)$, where $(v_0, v) \in V_C$, that is, v_0 is a direct ancestor

of v in T . The pre-conditions of a hold in state s' , so it follows that $\pi_2(s'(v_0)) = (v' \sim w)$. The variable v_0 has fewer ancestors than v . By induction, v' is a descendant of v_0 and $\pi_1(s'(v')) = w$. Hence $v' \leq v_0 \leq v$ and, by Lemma 8, $\pi_1(s(v')) = w$, completing the proof. \square

Proposition 10. *Suppose that (G, k, C) is a triple on which the algorithm R is defined. If the planning instance $R(G, k, C)$ is solvable, then G has a clique of size k .*

Proof. By design, during the course of some plan P solving $R(G, k, C)$ variable d must have held values (checked i) for all $i \in [M]$. Consequently, at some time of the plan operators $a = \text{recorda}(i, v, x, w)$ and $a' = \text{recordb}(i, v', x', w, w')$ have been applied, where $i \in [M]$, v, v' are such that $(v, d), (v', d) \in E_T$, and $w, w' \in V_G$ such that $(w, w') \in E_G$.

Let s be the state before applying operator a during the execution of plan P . The pre-conditions of a imply that $\pi_2(s(v)) = (a_i \sim w)$, so by Lemma 9 it follows $\pi_1(s(a_i)) = w$. The same reasoning applied to operator a' implies that $\pi_1(s(b_i)) = w'$. Most importantly, operator a' requires that $(w, w') \in E_G$.

Now, let $\sigma : \{v_1, \dots, v_k\} \rightarrow V_G$ be defined by $\sigma(a_i) = w$, $\sigma(b_i) = w'$ for all $i \in [M]$. Lemma 8 implies that function σ is well defined. Moreover, for $v, v' \in \{v_1, \dots, v_k\}$, let $j \in [M]$ be such that $e_j = \{v, v'\}$. It follows that $(\sigma(v), \sigma(v')) \in E_G$, hence σ is a total, injective function whose image defines a clique of size k in G . \square

Theorem 11. *Let \mathcal{C} be a set of directed graphs with unbounded pre-degree. The problem $\text{PlanExist}(\mathcal{C})$ is not polynomial-time decidable unless $W[1] \subseteq \text{nu-FPT}$.*

Proof. By Theorem 3, it suffices to show that p-CLIQUE non-uniformly reduces into $\text{PlanExist}(\mathcal{C})$. Let c be a constant such that $O(n^c)$ is a time bound for the algorithm of Theorem 6. Fix a natural number k . There exists a graph $C \in \mathcal{C}$ having $\text{pre-degree}(C) \geq k$, as \mathcal{C} has unbounded pre-degree. Then, instances of p-CLIQUE having the form $x = (G, k)$ may be reduced to $\text{PlanExist}(\mathcal{C})$ simply by invoking the algorithm of Theorem 6 on (G, k) and C ; as C is fixed, this process takes time $O(|x|^c)$. \square

Intractability: Bounded pre-degree

We now investigate families of graphs \mathcal{C} where $\text{cc-size}(\mathcal{C})$ is unbounded and \mathcal{C} has bounded pre-degree. We begin by showing that such graph families must contain certain structures which we call *source-sink configurations*.

Definition 12. Let C be a directed graph. A *source-sink configuration* of C is a triple (A, B, g) where A and B are disjoint non-empty subsets of V_C , and $g : B \rightarrow \binom{A}{1} \cup \binom{A}{2}$ is a mapping such that the following properties hold:

- For every vertex $b \in B$ and for every $a \in g(b)$, there is a directed path (in C)

$$a = v_0, v_1, \dots, v_n, v_{n+1} = b$$

from a to b such that none of the intermediate vertices $p(a, b) = \{v_1, \dots, v_n\}$ are in A nor B , that is, $p(a, b) \cap (A \cup B) = \emptyset$.

Moreover, the sets $p(a, b)$ of intermediate vertices (over $b \in B, a \in g(b)$) are pairwise disjoint.

- The undirected graph with vertex set A and edge set $\binom{A}{2} \cap \{g(b) \mid b \in B\}$ is connected in the sense that for any vertices $a, a' \in A$, there is a path from a to a' .

The size of a source-sink configuration is $|B|$.

Example 13. For $n \geq 2$, let Z_n be the ‘‘zig-zag’’ graph with vertex set $V_{Z_n} = \{b_1, \dots, b_n\} \cup \{a_1, \dots, a_{n-1}\}$ and edge set $E_{Z_n} = \{(a_i, b_i) \mid i \in [n-1]\} \cup \{(a_i, b_{i+1}) \mid i \in [n-1]\}$. It is readily verified that the family $\mathcal{Z} = \{Z_n \mid n \geq 2\}$ has pre-degree 2 and $\text{cc-size}(\mathcal{Z})$ unbounded (as a graph Z_n has component size $|V_{Z_n}| = 2n - 1$). It is readily verified that the graph Z_n has source-sink configuration $(\{a_1, \dots, a_{n-1}\}, \{b_1, \dots, b_n\}, g_n)$ with g_n defined by $g_n(b_i) = \{a_{i-1}, a_i\}$ for $i \in \{2, \dots, n-1\}$, $g_n(b_1) = \{a_1\}$, and $g_n(b_n) = \{a_{n-1}\}$.

The following is a graph-theoretic structure theorem; its proof is omitted.

Theorem 14. *Let \mathcal{C} be a set of directed graphs having bounded pre-degree and $\text{cc-size}(\mathcal{C})$ unbounded. For every $m \geq 1$, there exists a graph $C \in \mathcal{C}$ having a source-sink configuration (A, B, g) with $|B| = m$.*

Theorem 15. *There is an algorithm R that, given an instance (G, k) of the Clique problem and a directed graph C with a source-sink configuration $S = (A, B, g)$ with $|B| = \binom{k}{2}$, produces an equivalent instance $R(G, k, C, S)$ of $\text{PlanExist}(\mathcal{C})$ in time polynomial in $|G|$ and $|C|$.*

We give the formal description of the reduction, followed by an intuitive description.

Reduction 2 (Unbounded source-sink configuration). The planning instance $R(G, k, C, S) = (V, \text{init}, \text{goal}, O)$ is defined as follows. Let us denote the elements of B by $\{b_x \mid x \in \binom{\{v_1, \dots, v_k\}}{2}\}$, and let $T = V_C \setminus (A \cup B)$.

The variable set is $V = V_C$. The domains of the variables are

- $D(a) = \{\text{start}\} \cup \{(a \approx (v_j \sim w)) \mid w \in V_G, j \in [k]\}$ for all $a \in A$,
- $D(t) = \{\text{blank}\} \cup \{(a \approx (v_j \sim w)) \mid a \in A, j \in [k], w \in V_G\}$ for all $t \in T$,
- $D(b) = D_1(b) \times D_2(b)$ for all $b \in B$, where $D_1(b) = \{v_0\}$ if $|g(b)| = 1$, $D_1(b) = \{v_0, v_1, \dots, v_k\}$ if $|g(b)| = 2$, and $D_2(b) = \{\text{blank}, \text{edge}\} \cup V_G$

The initial state init is $\text{init}(a) = \text{start}$ for all $a \in A$, $\text{init}(t) = \text{blank}$ for all $t \in T$, and $\text{init}(b) = (v_0, \text{blank})$ for all $b \in B$.

The goal state goal is defined on all $b \in B$ as $\text{goal}(b) = (v_0, \text{edge})$ if $|g(b)| = 1$, and as $\text{goal}(b) = (v_k, \text{edge})$ if $|g(b)| = 2$.

Let $a \in A, i \in [k-1], w, w' \in V_G$. Then, the operators of O changing a are

- $\text{firstcommit}(a, w') = \langle a = \text{start}; a = (a \approx (v_1 \sim w')) \rangle$,
- $\text{commit}(a, i, w, w') = \langle a = (a \approx (v_i \sim w)); a = (a \approx (v_{i+1} \sim w')) \rangle$.

Let $t \in T$, p be such that $(p, t) \in E_C$, $a \in A$, $v \in \{v_1, \dots, v_k\}$ and $w \in V_G$. Then, the operators of O changing t are

- $\text{passmessage}(p, t, a, v, w) = \langle p = (a \approx (v \sim w)); t = (a \approx (v \sim w)) \rangle$.

Let $b \in B$ with $b = b_{\{v_i, v_j\}}$ (assume $i < j$), let p, p' be such that $(p, b), (p', b) \in E_C$, $a, a' \in A$, $w, w' \in V_G$, $(q_1, q_2) \in D(b)$. Then, the operators of O changing b are

- $\text{advance}(p, p', b, k, w, q_2) = \langle p = (a \approx (v_k \sim w)), p' = (a' \approx (v_k \sim w)), b = (v_{k-1}, q_2); b = (v_k, q_2) \rangle$ if $g(b) = \{a, a'\}$,
- $\text{remember}(p, b, a, w, q_1) = \langle b = (q_1, \text{blank}), p = (a \approx (v_i \sim w)); b = (q_1, w) \rangle$, and
- $\text{match}(p, b, a, w, w', q_1) = \langle b = (q_1, w), p = (a \approx (v_j \sim w')) \rangle; b = (q_1, \text{edge}) \rangle$ if $\{w, w'\} \in E_G$.

It is easily verified that the causal graph of $R(G, k, C, S)$ is a subgraph of C .

We give a brief intuitive description of this construction. Each of the variables $a \in A$ passes through a sequence of k values, having the form $(a \approx (v_1 \sim w_1)), \dots, (a \approx (v_k \sim w_k))$ for some vertices $w_1, \dots, w_k \in V_G$. Each of these variables $a \in A$ can be thought of as a ‘‘broadcaster’’ that sends messages to the variables $b \in B$ (having $a \in g(b)$); the variables $t \in T$ simply relay these messages. It is possible to achieve the goal state only if (1) the variables $a \in A$ broadcast exactly the same sequence of vertices w_1, \dots, w_k and (2) these vertices form a k -clique in G .

Each variable $b_{\{v_i, v_j\}} \in B$ is responsible for ensuring that the broadcast values w_i, w_j are joined by an edge in G . It accomplishes this by first remembering the value w_i in its second coordinate (via a remember operator) and then by changing its second coordinate to the value edge if it receives a value w_j that is adjacent to w_i in G . The variables $b \in B$ with $|g(b)| = 2$ also have another responsibility: to make sure that the two variables in $g(b) \subseteq A$ broadcast the same sequence w_1, \dots, w_k . The first coordinate of a variable $b \in B$ may be advanced to $v_i \in \{v_1, \dots, v_k\}$ only if both variables in $g(b)$ broadcast the same value w_i . Hence, such a first coordinate may reach v_k only if both variables in $g(b)$ broadcast the same sequence of w_i values; this is used in conjunction with the connectivity requirement in the definition of source-sink configuration to ensure that a goal state is reached if and only if all variables in A broadcast the same w_i sequence.

We now formally justify the construction.

Proposition 16. *Suppose that (G, k, C, S) is a tuple on which the algorithm R is defined. If G has a clique of size k , then the planning instance $R(G, k, C, S)$ is solvable.*

Proof. Let $\sigma : \{v_1, \dots, v_k\} \rightarrow V_G$ be a function describing a clique of size k of G . We describe a plan P solving $R(G, k, C, S)$. Plan P is the concatenation of subplans P_1, \dots, P_k . Fix some $i \in [k]$, and let $v = v_i$. Each plan P_i starts by committing each variable $a \in A$ to value $(a \approx (v \sim \sigma(v)))$ by employing $|A|$ operators of type $\text{firstcommit}(a, \sigma(v))$ if $i = 1$, and $|A|$ actions of type $\text{commit}(a, i - 1, \cdot, \sigma(v))$ otherwise.

The plan P_i proceeds by considering vertices $b \in B$ with $g(b) = \{a, a'\}$ of size 2, one at a time. Due to part (a) of Definition 12, there exist two disjoint paths from a to b and from a' to b , say, paths $a = v_0^a, v_1^a, \dots, v_n^a = v^a$ and $a' = v_0^{a'}, v_1^{a'}, \dots, v_m^{a'} = v^{a'}$, with $(v^a, b), (v^{a'}, b) \in E_C$. Then, plan P_i uses operators of type $\text{passmessage}(v_{j-1}^a, v_j^a, a, v_i, \sigma(v))$ for $j \in [n]$ to propagate the value $(a \approx (v \sim \sigma(v)))$ up to variable v^a , and similar operators to propagate $(a' \approx (v \sim \sigma(v)))$ up to variable $v^{a'}$. Plan P_i then uses an operator $\text{advance}(v^a, v^{a'}, b, i, \sigma(v), \cdot)$ that moves the first component of variable b from v_{i-1} to v_i , one step closer to its goal.

In addition, for each $b = b_{\{v_i, v_j\}} \in B$, the plan P_i contains an extra operator of type $\text{remember}(v^a, b, a, \sigma(v_i), \cdot)$ if $i < j$, or an extra operator of type $\text{match}(v^a, b, a, \sigma(v_i), \sigma(v_j), \cdot)$ if $i > j$. Note that we can apply operator match for any pair $\{v_i, v_j\}$ because σ describes a clique. Plan P_i repeats this process for all variables $b \in B$.

After the execution of plan P_i the first component of all variables $b \in B$ with $|g(b)| = 2$ is v_i , and after the execution of $P_{\max\{i, j\}}$ the second component of the variable $b = b_{\{v_i, v_j\}}$ is edge. Hence, after executing plan P_k all variables $b \in B$ are in a goal state. \square

Lemma 17. *Let P be a plan of $R(G, k, C, S)$. Then, the variable $a \in A$, over the execution of P , takes a sequence of values $\text{start}, (a \approx (v_1 \sim w_1)), (a \approx (v_2 \sim w_2)), \dots, (a \approx (v_j \sim w_j))$, in that order, for some $j \in [0, k]$ and $w_1, \dots, w_j \in V_G$*

Proof. The only operators changing the value of a are $\text{firstcommit}(a, \cdot)$ and $\text{commit}(a, i, \cdot, \cdot)$ for $i \in [k - 1]$. By design, the first operator changing a in plan P , if any, must be of type $\text{firstcommit}(a, \cdot)$, and subsequent ones must be of type $\text{commit}(a, i, \cdot, \cdot)$ where i ranges from 1 to j , for some $j \in [0, k]$. \square

Definition 18. Let w_i, j be as in Lemma 17. For a plan P of $R(G, k, C, S)$ and a variable $a \in A$, we define $f_{(P, a)} : \{v_1, \dots, v_k\} \rightarrow V_G$ as the partial function $f_{P, a}(v_i) = w_i$ if $i \in [j]$, and undefined otherwise.

Note that this function is not defined anywhere if the only value taken on by a is start , in which case $j = 0$; and is a total function in the case that $j = k$.

Lemma 19 (Messages are truthful). *Let s be a reachable state such that $s(t) = (a \approx (v \sim w))$ for some $a \in A$, $v \in \{v_1, \dots, v_k\}$ and $w \in V_G$. Then $f_{(P, a)}(v) = w$ for any plan P such that $\text{init}[P] = s$.*

Proof. The lemma is true if $t = a$, by definition of $f_{(P, a)}$. If not, let P be any plan such that $\text{init}[P] = s$. Clearly, P contains an operator $o = \text{passmessage}(p, t, a, v, w)$ for some p such that $(p, t) \in E_C$. Let s_p be the state resulting of the application of plan P up to operator o . Then, the lemma holds by induction on state s_p and variable p . \square

Lemma 20. *Let P be a plan solving $R(G, k, C, S)$. Then all the mappings $f_{(P, a)}$ over $a \in A$ are equal and total.*

Proof. A plan P solving $R(G, k, C, S)$ reaches a state $s = \text{init}[P]$ satisfying the goal condition, so $s(b) = (v_k, \text{edge})$ for any $b \in B$ with $g(b) = \{a, a'\}$ of size 2. We prove that $f_{(P,a)}$ and $f_{(P,a')}$ are equal and both total. This is sufficient to prove the lemma since, by the second property in Definition 12, the whole set of vertices A is connected by edges of the form $g(b)$, for some $b \in B$.

Clearly, the only operators changing the first component of variable b onto v_i for $i \in [k]$ are those of type $\text{advance}(\cdot, \cdot, b, i, \cdot, \cdot)$, which require as pre-condition $b = (v_{i-1}, q)$ for any $q \in D_2(b)$. Hence, plan P must contain operators of this type for all $i \in [k]$.

Consider the operator $o_i = \text{advance}(p, p', b, i, w, q)$ on plan P . Let s_i be the state resulting on the application of plan P up to operator o_i . By the pre-condition of o_i it holds that $s_i(p) = (a \approx (v_i \sim w))$ and $s_i(p') = (a' \approx (v_i \sim w))$ and, by Lemma 19, it follows that $f_{(P,a)}(v_i) = f_{(P,a')}(v_i) = w$. \square

Proposition 21. *Suppose that (G, k, C, S) is a tuple on which the algorithm R is defined. If the planning instance $R(G, k, C, S)$ is solvable, then G has a clique of size k .*

Proof. Let P be a plan solving $R(G, k, C)$. We want to show that G has a clique of size k . By Lemma 20, we have that the mappings $f_{(P,a)}$ over $a \in A$ are equal and total. In light of this, we simply use f_P to denote this single mapping.

Let us turn our attention to the second coordinate of the value of a variable $b = b_{\{v_i, v_j\}} \in B$. (We assume that $i < j$.) The initial value $\pi_2(\text{init}(b))$ is blank, and the goal value $\pi_2(\text{goal}(b))$ is edge. To achieve this change, P must contain both a $\text{remember}(\cdot, b, \cdot, \cdot, \cdot)$ operator and a $\text{match}(\cdot, b, \cdot, \cdot, \cdot)$. (It is clear from inspection that these are the only two actions that may change the second coordinate of the value of b .) Suppose that after the remember action, we have $b = (q, w)$. By the precondition, there is some variable p such that $p = (a \approx (v_i \sim w))$, from which we obtain, by Lemma 19, that $f_P(v_i) = w$. From there, a match action on b may only be executed if $p = (a \approx (v_j \sim w'))$ for some w' with $\{w, w'\} \in E_G$. Again, by Lemma 19, if such a match action is executed, we have $f_P(v_j) = w'$ and hence $\{f_P(v_i), f_P(v_j)\} \in E_G$.

In the previous paragraph, the variable $b_{\{v_i, v_j\}} \in B$ was chosen arbitrarily. Hence, for all pairs $i < j$ with $i, j \in [k]$, we have $\{f_P(v_i), f_P(v_j)\} \in E_G$, from which it follows that $\{f_P(v_1), f_P(v_2), \dots, f_P(v_k)\}$ is a clique of size k in G . \square

Theorem 22. *Let \mathcal{C} be a set of directed graphs with source-sink configurations of unbounded size. The problem $\text{PlanExist}(\mathcal{C})$ is not polynomial-time decidable unless $W[1] \subseteq \text{nu-FPT}$.*

Proof. By Theorem 3, it suffices to show that p-CLIQUE non-uniformly reduces into $\text{PlanExist}(\mathcal{C})$. Let c be a constant such that $O(n^c)$ is a time bound for the algorithm of Theorem 15. Fix a natural number k . By Theorem 14, there exists a graph $C \in \mathcal{C}$ having a source-sink configuration $S = (A, B, g)$ of size $|B| = \binom{k}{2}$. Then, a reduction from the instances of p-CLIQUE having the form $x = (G, k)$ to the problem $\text{PlanExist}(\mathcal{C})$ is the algorithm of Theorem 15

executed on $x = (G, k)$ and C with source-sink configuration S . As C and S are fixed, this reduction takes time $O(|x|^c)$. \square

From Theorems 14 and 22, it follows that for any set of directed graphs \mathcal{C} having bounded pre-degree and $\text{cc-size}(\mathcal{C})$ unbounded, the problem $\text{PlanExist}(\mathcal{C})$ is not polynomial-time decidable unless $W[1] \subseteq \text{nu-FPT}$.

Reversible Planning

We say that a planning instance $\Pi = (V, \text{init}, \text{goal}, A)$ is *reversible* if from any reachable state s , there exists a plan P such that $s[P] = \text{init}$.

For a set of directed graphs \mathcal{C} , we define $\text{RevPlanExist}(\mathcal{C})$ and $\text{RevPlanGen}(\mathcal{C})$ are the plan existence and generation problems restricted to reversible planning instances whose causal graphs are elements of \mathcal{C} .

For a directed graph C , we define the *strongly connected component size* of C , denoted $\text{scc-size}(C)$, to be the size of the largest strongly connected component of C .

For a set of directed graphs \mathcal{C} , say that $\text{scc-size}(\mathcal{C})$ is *bounded* if there exists a constant k such that for all $C \in \mathcal{C}$, it holds that $\text{scc-size}(C) \leq k$.

Theorem 23. *Let \mathcal{C} be a set of directed graphs.*

- *If $\text{scc-size}(\mathcal{C})$ is bounded, then the problem $\text{RevPlanGen}(\mathcal{C})$ is polynomial-time solvable (under a succinct plan representation);*
- *otherwise, the problem $\text{RevPlanExist}(\mathcal{C})$ is not polynomial-time decidable (unless $W[1] \subseteq \text{nu-FPT}$).*

Tractability

Theorem 24. *There is an algorithm that solves instances of problem $\text{RevPlanGen}(\mathcal{C})$, where \mathcal{C} is a set of directed graphs with strongly connected component size bounded by k , in time $O(n^{k+2})$, where n is the total size of the input planning problem.*

This algorithm is essentially equivalent to algorithms presented by Knoblock (1994, Section 3.2) and Helmert (2006, Section 5.2).

Intractability

To establish the intractability part of this classification, we demonstrate a reduction from $\text{PlanExist}(\mathcal{C})$ to $\text{RevPlanExist}(\mathcal{C})$, for all graph sets \mathcal{C} where $\text{scc-size}(\mathcal{C})$ is unbounded. For such graph sets, the problem $\text{PlanExist}(\mathcal{C})$ is intractable by Theorem 4. In particular, we establish the following.

Theorem 25. *There is an algorithm R that, given a planning instance Π of $\text{PlanExist}(\mathcal{C})$ where \mathcal{C} is a strongly connected graph, produces an equivalent instance $R(\Pi)$ of $\text{RevPlanExist}(\mathcal{C})$ in time polynomial in Π .*

The intuition behind this reduction is the following. For such an instance Π of $\text{PlanExist}(\mathcal{C})$, the fact that \mathcal{C} is strongly connected allows us to augment Π with a message-passing mechanism that makes it always possible for Π to be “reset” to its initial state. This augmentation is done in a way that preserves solvability of the instance.

Structurally Restricted Planning

In this section, we define and begin to study an object that, as with the causal graph, is defined for each planning instance. We call this object the *interaction network*; it is essentially what one obtains by “forgetting” the values in the precondition and postcondition of each operator, and retaining just the variables themselves.

Definition 26. An *interaction network* is a pair (V, E) where V is a finite set and $E \subseteq \wp(V) \times \wp(V)$. (We use $\wp(V)$ to denote the power set of V .) The *interaction network* of a planning instance $\Pi = (V, \text{init}, \text{goal}, A)$ is the pair (V, E) where E is the set of pairs

$$\{(\text{vars}(\text{pre}(a)), \text{vars}(\text{post}(a))) \mid a \in A\}.$$

For a set \mathcal{N} of interaction networks, define $\text{PlanExist}(\mathcal{N})$ to be the problem of deciding, given a planning instance Π whose interaction network is in \mathcal{N} , whether or not a solution plan exists. Similarly, $\text{PlanGen}(\mathcal{N})$ is defined as the problem of outputting, given a planning instance Π whose interaction network is in \mathcal{N} , a solution plan, if one exists. These definitions give rise to a new research problem: classify the complexity of the problems $\text{PlanExist}(\mathcal{N})$ and $\text{PlanGen}(\mathcal{N})$ over sets of interaction networks \mathcal{N} . We expect that investigation of this problem will give rise to phenomena much richer than those that appeared in the classification of causal graphs.

The causal graph of an instance Π can be derived from the interaction network of Π . For an interaction network $N = (V, E)$, define $\text{cg}(N)$ to be the directed graph (V, E') where E' contains all pairs (u, v) such that $u \neq v$, and there exists a pair $(X, Y) \in E$ such that $u \in X \cup Y$ and $v \in Y$. It is straightforward to verify that for any instance Π with interaction network N , it holds that the causal graph of Π is equal to $\text{cg}(N)$. For a set of interaction networks \mathcal{N} , we define $\text{cg}(\mathcal{N}) = \{\text{cg}(N) \mid N \in \mathcal{N}\}$.

The interaction network contains more information than the causal graph. In particular, one can give tractability results on problems of the form $\text{PlanGen}(\mathcal{N})$ in cases where $\text{PlanGen}(\text{cg}(\mathcal{N}))$ is intractable.

Example 27. Let \mathcal{N}_C denote the set of interaction networks $N = (V, E)$ where (1) for all $e \in E$, it holds that $e = (\{u\}, \{v\})$ for $u, v \in V$ and (2) the directed graph (V, E') with $E' = \{(u, v) \mid (\{u\}, \{v\}) \in E\}$ is a directed path. In Theorem 28, we show that $\text{PlanGen}(\mathcal{N}_C)$ is polynomial-time solvable; however, it follows from Theorem 4 that $\text{PlanGen}(\text{cg}(\mathcal{N}_C))$ is intractable.

Theorem 28. *The problem $\text{PlanGen}(\mathcal{N}_C)$ is polynomial-time solvable.*

The tractability of precondition-free planning, obtained by Bylander (1994), can also be cast within the interaction network framework. In particular, let \mathcal{N}_P denote the set of interaction networks (V, E) where $E \subseteq \{\emptyset\} \times \wp(V)$. The instances of $\text{PlanGen}(\mathcal{N}_P)$ are then exactly those which have empty preconditions, and we may state this tractability result as follows.

Theorem 29. (Bylander 1994) *The problem $\text{PlanGen}(\mathcal{N}_P)$ is polynomial-time solvable.*

Note that for a network $N = (V, E) \in \mathcal{N}_P$, each pair (\emptyset, S) induces a clique over S in $\text{cg}(N)$. Consequently, the problem $\text{PlanGen}(\text{cg}(\mathcal{N}_P))$ is intractable (for example, by Theorem 4).

References

- Bodirsky, M., and Grohe, M. 2008. Non-dichotomies in constraint satisfaction complexity. To appear in ICALP 2008.
- Brafman, R., and Domshlak, C. 2003. Structure and Complexity in Planning with Unary Operators. *Journal of Artificial Intelligence Research* 18:315–349.
- Brafman, R., and Domshlak, C. 2006. Factored Planning: How, When, and When Not. In *Proceedings of the 21st National Conference on Artificial Intelligence*.
- Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69:165–204.
- Domshlak, C., and Dinitz, Y. 2001. Multi-Agent Off-line Coordination: Structure and Complexity. In *Proceedings of the 6th European Conference on Planning*, 277–288.
- Downey, R., and Fellows, M. 1995. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science* 141.
- Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer-Verlag.
- Giménez, O., and Jonsson, A. 2008. The Complexity of Planning Problems with Simple Causal Graphs. *Journal of Artificial Intelligence Research* 31:319–351.
- Gottlob, G.; Leone, N.; and Scarcello, F. 2000. A comparison of structural CSP decomposition methods. *Artif. Intell.* 124(2):243–282.
- Grohe, M. 2007. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM* 54(1).
- Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, 161–170.
- Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26:191–246.
- Jonsson, P., and Bäckström, C. 1998. Tractable plan existence does not imply tractable plan generation. *Annals of Mathematics and Artificial Intelligence* 22(3–4):281–296.
- Jonsson, A. 2007. The Role of Macros in Tractable Planning Over Causal Graphs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1936–1941.
- Katz, M., and Domshlak, C. 2007. Structural patterns of tractable sequentially-optimal planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling*.
- Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.