Parameterising the Complexity of Planning by the Number of Paths in the Domain-transition Graphs

Christer Bäckström [1]

Abstract. We apply the theory of parameterised complexity to planning, using the concept of fixed-parameter tractability (fpt) which is more relaxed than the usual tractability concept. The parameter we focus on is the maximal number of paths in the domain-transition graphs, and we show that for this parameter, optimal planning is fpt for planning instances with polytree causal graphs and acyclic domain-transition graphs. If this parameter is combined with the additional parameters of domain size for the variables and the treewidth of the causal graph, then planning is fpt also for instances with arbitrary causal graphs. Furthermore, all these parameters are fpt to test in advance. These results also imply that delete-relaxed planning is fpt, even in its recent generalisation to non-binary variables.

1 INTRODUCTION

Early papers on the complexity of planning focussed on analysing subclasses defined by various combinations of (primarily) syntactical properties like the number of preconditions and effects [4, 9]. Later research has often focussed on the properties of the casusal graph of a planning instance, usually in combination with other properties. For instance, it is known that planning remains **NP**-hard even when the causal graph is a directed star (fork or inverted fork) [12], a directed path (chain) [21], a fence or a polypath [3] or a polytree [20]. Tractability results exist in all these cases, usually with restrictions on the variable domains to two or three values and/or restrictions on the degree of the graph. Identifying tractable fragments of planning is important for the theoretical understanding of when planning is hard and when it is easy, which can be exploited also for designing algorithms. It has also proven very useful for designing heuristics, as pointed out by Katz and Domshlak [25]:

> Computational tractability can be an invaluable tool even for dealing with problems that fall outside all the known tractable fragments of planning. For instance, tractable fragments of planning provide the foundations for most (if not all) rigorous heuristic estimates employed in planning as heuristic search.

For instance, Helmert [22] used a planning algorithm for a simpler restricted problem to compute heuristic values for subproblems and then combine these values. Similarly, the highly successful delete-relaxation heuristic (h^+) [23] exploits that planning is simpler with only positive effects and uses this as a relaxation for computing a heuristic value. Also Katz and Domshlak [26] exploit tractable fragments to build pattern database heuristics.

We will analyse some cases of planning using parameterised complexity analysis rather than standard complexity analysis. Parameterised complexity analysis was invented to enable a more fine-grained analysis than standard complexity analysis allows, by treating some selected parameter(s) as independent of the instance rather than being a part of it. Somewhat simplified, the idea is as follows. Consider an instance of some problem and let n denote the instance size. We usually consider a problem as tractable if it can be solved by some algorithm in $O(n^c)$ time, that is, in polynomial time. For many problems, like the **NP**-hard problems, we do not know of any significantly faster way to solve them than doing brute-force search, which typically requires exponential, or at least super-polynomial, time in n. In practice the search is often not exponential in the size of the whole instance, but only in some smaller hard part of it. Then the complexity may rather be something like $O(2^k n^c)$ where k is a parameter that is typically independent of the instance size n. The combinatorial explosion is thus confined to the parameter k. A problem is *fixed-parameter tractable (fpt)* if it can be solved in this way. This is the essence of parameterised complexity theory and it provides a more relaxed tractability concept than the usual one, while correlating better with tractability in practice for real-world problems. The theory also offers various classes for problems that are not fpt, for example $\mathbf{W}[1]$ and $\mathbf{W}[2]$. Parameterised complexity analysis has contributed fundamental new insights into complexity theory [15, 19]. It is nowadays a very common technique in many areas of computer science, including many subareas of AI, like non-monotonic reasoning, constraints, social choice and argumentation. There has also been an increasing interest in parameterised complexity within planning. Some early results on STRIPS appeared in Downey, Fellows and Stege [14]. Bäckström *et. al.* [2] used plan length as parameter and studied the complexity of planning under various restrictions previously considered for classical complexity analysis in the literature. Kronegger *et. al.* [28] provided an extensive analysis of the complexity of planning for all combinations of a number of parameters, mostly parameters based on previously studied restrictions. De Hahn *et. al.* [10] applied parameterised analysis to plan reuse.

In this paper we study planning instances where the domain-transition graphs (DTGs) of the variables are acyclic and focus primarlily on a parameter k, which denotes the maximum number of paths in a DTG. This is a natural parameter that appears in the literature (cf. Katz and Keyder [27]), but it has never been used in parameterised analysis. Most other parameters previously used are either straightforward syntactical parameters, e.g the number of preconditions or the size of the variable domains, or implicit properties of the actual solution, e.g the length of an optimal plan or the local work parameter (Δ) [8]. The value for parameters of the first type are usually trivial to test, while parameters of the second type are typically as hard to test as planning itself. For instance, we show in this paper that testing the Δ parameter is **PSPACE**-complete under classical analy-

[1] Department of Computer and Information Science, Linköping University, SE-58183 Linköping, Sweden. Email: christer.backstrom@liu.se

sis and **W**[2]-complete under parameterised analysis, which makes it moderately interesting to use as a parameter. In contrast, our parameter k is not an immediate explicit property of a planning instance, but an implicit one. Yet, it is fixed-parameter tractable to test. Furthermore, Brafman and Domshlak [6] studied the complexity of planning when the number of paths in the causal graph is restricted. Since both the causal graph and the DTG are important tools for describing and analysing planning, it is interesting to consider the same restrictions for both graph types, even though they model different things.

Our main results are: (1) planning with acyclic DTGs and polytree causal graphs is fpt in parameter k and (2) planning with acyclic DTGs and no restrictions on the causal graph is fpt if parameter k is combined with the domain size d of the variables and the treewidth w of the causal graph. These results still hold even if we ask for the length of an optimal plan. The proofs are based on translating planning to constraint satisfaction (CSP) instances since parameter k has favourable properties that makes this translation natural without blowing up the domain sizes exponentially in the instance size.

The rest of the paper is organised as follows. Section 2 provides some necessary definitions from graph theory, parameterised complexity and planning. Section 3 discusses the parameters and assumptions used for the results. Section 4 presents the main fpt results for planning with arbitrary causal graphs and with polytree causal graphs. Section 5 shows that these fpt results still hold if we ask for an optimal plan. Finally, Section 6 contains a discussion on the restrictions, on the relations to some other work in the literature and on how to go forward from these results.

2 PRELIMINARIES

This section provides some basic definitions for graph theory, parameterised complexity and the planning framework used in this paper.

2.1 Graphs

Given a directed graph $G = \langle V, E\rangle$, the notation $U(G)$ refers to the undirected version of G, that is, $U(G) = \langle V, E_U\rangle$ where $E_U = \{\{v, w\} \mid \langle v, w\rangle \in E\}$. Although many special types of graphs appear in the literature on causal graphs, there is only one that is of particular interest in this paper, the *polytree* graph. A directed graph G is a polytree if $U(G)$ is a tree, i.e. if we ignore the direction of the edges then G must be connected and contain no cycles.

The *treewidth* of a graph is a concept that is commonly used, especially in parameterised analysis of problems.

Definition 1. A *tree decomposition* of a graph $G = \langle V, E\rangle$ is a tuple $\langle N, T\rangle$ where $N = \{N_1, \ldots, N_n\}$ is a family of subsets of V and T is a tree with nodes $N_1, \ldots, N_n$, satisfying the following properties (the term node is used to refer to a vertex of T to avoid confusion with vertices of G):

1. The union of all sets N_i equals V. That is, each graph vertex is contained in at least one tree node.
2. For each vertex $v \in V$, the tree nodes of N containing vertex v form a connected subtree of T.
3. For every edge $\{v, w\}$ in the graph, there is a subset N_i that contains both v and w. That is, vertices are adjacent in the graph only when the corresponding subtrees have a node in common.

The *width* of a tree decomposition is the size of its largest set N_i minus one. The *treewidth* of a graph G is the minimum width among all possible tree decompositions of G. Perhaps not surprisingly, every tree has treewidth 1.

2.2 Parameterised complexity

We define the basic notions of parameterised complexity and refer to other sources [15, 19] for an in-depth treatment. A *parameterised problem* is a set of pairs $\langle \mathbb{I}, k\rangle$, the *instances*, where $\mathbb{I}$ is the main part and k the *parameter*. The parameter is usually one or more non-negative integers. A parameterised problem is *fixed-parameter tractable (fpt)* if there exists an algorithm that solves any instance $\langle \mathbb{I}, k\rangle$ of size n in time $f(k) \cdot n^c$ where f is an arbitrary computable function and c is a constant independent of both n and k. That is, the expression can be separated into a function $f(k)$ that depends only on the parameter(s) and a polynomial function n^c that depends only on the instance size. **FPT** is the class of all fixed-parameter tractable decision problems. A parameter is not the same thing as assuming that the value is a constant. For instance, although $O(n^k)$ is tractable in the classical sense if k is a constant, the expression is not fpt when k is the parameter since it is not separable. Although the parameter value is often assumed much smaller than the instance size, it need not even be polynomially bounded in the instance size. Parameterised complexity offers a completeness theory, similar to the theory of **NP**-completeness, based on a hierarchy of complexity classes $\mathbf{FPT} \subseteq \mathbf{W}[1] \subseteq \mathbf{W}[2] \subseteq \mathbf{W}[3] \subseteq \cdots$, where the class **W**[1] is usually considered as a parameterised analogue of **NP**, although neither class is included in the other. We will not go further into this since we primarily prove tractability results in this paper.

2.3 Planning framework

We use the SAS$^+$ planning framework [4]. Let $V = \{v_1, \ldots, v_n\}$ be a finite set of *variables*, with an implicit order $v_1, \ldots, v_n$. A *domain function* D for V assigns a finite domain $D(v_i)$ to each variable $v_i \in V$. The *state space* for V and D is $S(V, D) = D(v_1) \times \ldots \times D(v_n)$ and the members of $S(V, D)$ are called *(total) states*. The value of a variable v_i in a state s is called the *projection* of s onto v_i and is denoted $s[v_i]$. This can be viewed as a total function over V such that $s[v_i] \in D(v_i)$ for all $v_i \in V$. A *partial state* is similarly a partial function over V such that for each $v_i \in V$, either $s[v_i]$ is undefined or $s[v_i] \in D(v_i)$. The notation $\text{vars}(s)$ denotes the set of variables $v_i \in V$ such that $s[v_i]$ is defined. Projection is extended to sets of variables such that if $V' \subseteq V$, then $s[V']$ is a partial state that agrees with s on all variables in $\text{vars}(s) \cap V'$ and is otherwise undefined.

A *planning instance* is a tuple $\mathbb{P} = \langle V, D, A, s_I, s_G\rangle$ where V is a set of variables, D is a domain function for V and A is a set of *actions*. The *initial state* s_I and the *goal* s_G are total states over $S(V, D)$. Each action $a \in A$ has two associated partial states, the *precondition* $\text{pre}(a)$ and the *effect* $\text{eff}(a)$. Let $a \in A$ and let s be a total state. Then a is *valid in* s if $\text{pre}(a)[v] = s[v]$ for all $v \in \text{vars}(\text{pre}(a))$. Furthermore, the *result of* a *in* s is a state $t \in S(V, D)$ such that for all $v \in V$, $t[v] = \text{eff}(a)[v]$ if $v \in \text{vars}(\text{eff}(a))$ and $t[v] = s[v]$ otherwise.

Let $s_0, s_\ell \in S(V, D)$ and let $\omega = \langle a_1, \ldots, a_\ell\rangle$ be a sequence of actions. Then ω is a *plan from* s_0 *to* s_ℓ if either (1) $\omega = \langle\rangle$ and $\ell = 0$ or (2) there are states $s_1, \ldots, s_{\ell-1} \in S(V, D)$ such that for all i ($1 \leq i \leq \ell$), a_i is valid in s_{i-1} and s_i is the result of a_i in s_{i-1}. An action sequence ω is a *plan for* $\mathbb{P}$ if it is a plan from s_I to s_G.

We will study the following problems, where C is some class of planning instances and π is a list of parameters. $\textsc{PlanSat}(C, \pi)$ takes a planning instance $\mathbb{P}$ in C and values for the parameters in π as input and answers *yes* if $\mathbb{P}$ has a plan and otherwise answers *no*. $\textsc{PlanOpt}(C, \pi)$ has the same input but its output is either the length

of the shortest plan for $\mathbb{P}$ or *no* if there is no plan.

Projections are extended as follows. Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance and let $V' \subseteq V$. Then, for each action $a \in A$, define $a[V']$ as the restriction a' of a where $\text{pre}(a') = \text{pre}(a)[V']$ and $\text{eff}(a') = \text{eff}(a)[V']$. That is, we treat a and $a[V']$ as different variants of the same action. Also define $A[V'] = \{a[V'] \mid a \in A \text{ and } \text{eff}(a[V']) \neq \varnothing\}$. Then $\mathbb{P}[V'] = \langle V', A[V'], s_I[V'], s_G[V'] \rangle$. The projection of an action sequence $\omega = \langle a_1, \ldots, a_\ell \rangle$ over A onto V' is denoted $\omega[V']$ and defined as follows. First define the sequence $\omega' = \langle a'_1, \ldots, a'_\ell \rangle$ such that $a'_i = a_i[V']$ for all i ($1 \leq i \leq \ell$). Then define $\omega[V']$ as the subsequence of ω' that contains only those a'_i where $\text{eff}(a'_i) \neq \varnothing$. Also here, we consider $\omega[V']$ to be a subsequence of ω, i.e. it consists of actions from ω although in a restricted variant. For all cases, we also define projection onto a single variable v such that $a[v] = a[\{v\}]$ etc. The following result is known in the literature (cf. Helmert [22]).

Proposition 2. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance and let $V' \subseteq V$. If ω is a plan for $\mathbb{P}$, then $\omega[V']$ is a plan for $\mathbb{P}[V']$.*

We define the *transition graph* for a planning instance $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ as the labelled directed graph $\text{TG}(\mathbb{P}) = \langle S, E \rangle$, where $S = S(V, D)$ and $E \subseteq S \times A \times S$ such that for all $s, t \in S$ and $a \in A$, $\langle s, a, t \rangle \in E$ if a is valid in s and t is the result of a in s. Obviously, the paths from s_I to s_G in $\text{TG}(\mathbb{P})$ correspond to the plans for $\mathbb{P}$. In particular, we define the *domain transition graph (DTG)* for a variable $v \in V$ as $\text{DTG}(v) = \text{TG}(\mathbb{P}[v])$, i.e. the paths from $s_I[v]$ to $s_G[v]$ in $\text{DTG}(v)$ describe all possible ways to go from the initial state to the goal for this particular variable treated in isolation. DTGs are sometimes ornamented with further information in the literature, but this definition is sufficient for our needs. Both the transition graph and the DTGs are multigraphs since different actions can induce different edges between the same pair of vertices.

The *causal graph* for a planning instance describes how the variables of the instance depend on each other, as implicitly defined by the actions. Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance. Then the *causal graph* of $\mathbb{P}$ is the directed graph $CG(\mathbb{P}) = \langle V, E \rangle$ where E contains the edge $\langle v, w \rangle$ for every pair of distinct vertices $v, w \in V$ such that (1) $v \in \text{vars}(\text{pre}(a)) \cup \text{vars}(\text{eff}(a))$ and (2) $w \in \text{vars}(\text{eff}(a))$ for some action $a \in A$.

3 PARAMETERS AND ASSUMPTIONS

We will consider the following parameters in this paper:

d: The maximum domain size of the variables.
k: The maximum number of paths in each DTG.
w: The treewidth of the causal graph.

Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance. Then parameter d is defined as $d = \max_{v \in V} |D(v)|$, which can be computed in polynomial time. For each v, let $\text{Paths}(v)$ be the set of all paths in $\text{DTG}(v)$ from $s_I[v]$ to $s_G[v]$. Then parameter k is defined as $k = \max_{v \in V} |\text{Paths}(v)|$. Computing k is fixed parameter tractable.

Lemma 3. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance such that $\text{DTG}(v)$ is acyclic for all $v \in V$ and let $k > 0$ be an integer. Checking if $|\text{Paths}(v)| \leq k$ for all $v \in V$ is fpt in k.*

Proof. Eppstein [16] has presented an algorith that finds the k shortest paths in a directed acyclic graph in time $O(m + k)$, where m is the number of edges. This algorithm works also for multigraphs and is thus applicable to DTGs. To check if $|\text{Paths}(v)| \leq k$ for a variable v, apply this algorithm but ask for the $k+1$ shortest paths. Obviously, $|\text{Paths}(v)| \leq k$ if and only if the algorithm returns fewer than $k + 1$ paths. Checking this for all n variables takes time $O((m+k)n)$. This is fpt in parameter k since $(m + k)n = mn + kn \leq (k + 1) \cdot mn$ and both m and n are upper bounded by the instance size. □

We define parameter w as the treewidth of $U(CG(\mathbb{P}))$ since there is no generally accepted concept of treewidth for directed graphs. Deciding the treewidth of a graph is **NP**-complete [1], but fpt under parameterised analysis [5].

Our results will concern planning instances where all variables have acyclic DTGs and we make the following observations.

Lemma 4. *Let $\text{DTG}(v)$ be an acyclic DTG with d vertices and at most k paths from $s_I[v]$ to $s_G[v]$. Then (1) each path has length at most $d - 1$, (2) if we remove all edges that are not on any of the paths, then at most $k(d - 1)$ edges remain and (3) for any path ω in $\text{Paths}(v)$, no action occurs more than once along the path.*

Proof. (1) and (2) are straightforward. For (3), note that if an action a occurs twice in a path in $\text{DTG}(v)$, then it must visit the vertex $\text{eff}(a)[v]$ twice, implying that $\text{DTG}(v)$ is not acyclic. □

The restriction to total goal states is not necessary but simplifies the proofs. Besides, if partial goals were allowed, then the number of paths would be at least the same as the number of reachable vertices in the DTG. Similarly, the restriction to acyclic DTGs is not necessary either. If paths are taken to mean non-simple paths that may have cycles, as in Eppsteins algorithm, then the number of paths will immediately be unbounded if any of the paths has a cycle. The acyclicity is thus rather a consequence of the bound.

4 PLAN SATISFIABILITY

This section provides the complexity results for two cases of the PLANSAT problem. We first prove the more general result for arbitrary causal graphs (Theorem 9) and then show that the result for polytrees (Corollary 10) follows. Before that, we first recapitulate the concept of CSP, since it will be central to the forthcoming proofs.

Definition 5. An instance of the *constraint satisfaction problem (CSP)* is a triple $\mathbb{C} = \langle X, D, C \rangle$, where X is a finite set of variables, D is a domain function assigning a finite domain to each variable and C is a finite set of constraints. Each constraint in C is a tuple $\langle t, R \rangle$ where t is a sequence $\langle x_{i_1}, \ldots, x_{i_r} \rangle$ of variables from X and R is a relation $R \subseteq D(x_{i_1}) \times \ldots \times D(x_{i_r})$. An *instantiation* of $\mathbb{C}$ is a mapping α that maps each $x_i \in X$ to an element in $D(x_i)$. A *solution* for $\mathbb{C}$ is an instantiation α that satisfies all constraints in C, i.e. $R(\alpha(x_{i_1}), \ldots, \alpha(x_{i_r}))$ holds for every constraint $\langle \langle x_{i_1}, \ldots, x_{i_r} \rangle, R \rangle$ in C. When all constraint relations are binary, the *constraint graph* for $\mathbb{C}$ is the graph $G = \langle X, E \rangle$ where E contains the edge $\{x_i, x_j\}$ whenever there is some constraint $\langle t, R \rangle$ such that $t = \langle x_i, x_j \rangle$ or $t = \langle x_j, x_i \rangle$.

Dechter and Pearl [11] have shown that a CSP instance $\mathbb{C} = \langle X, D, C \rangle$ where all constraints are binary and the constraint graph is a tree can be solved in time $O(d^2 n)$, where d is the maximum domain size and n is the number of variables.

The main theorem of this section is based on the following construction, which defines a CSP instance with certain properties for every planning instance.

Construction 6. Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance and let $\langle N, T \rangle$ be a tree decomposition of $CG(\mathbb{P})$. Define a corresponding CSP instance $\mathbb{C} = \langle X, D, C \rangle$ as follows. The set X contains one variable x_i for each node $N_i \in N$ where the domain $D(x_i)$ for x_i is the set of plans for $\mathbb{P}[N_i]$. For each pair of adjacent nodes N_i, N_j in T such that $i < j$, define the relation $R_{i,j} \subseteq D(x_i) \times D(x_j)$ such that $R_{i,j}$ contains exactly those tuples $\langle \omega_i, \omega_j \rangle$ where $\omega_i[N_i \cap N_j] = \omega_j[N_i \cap N_j]$, i.e. those tuples where ω_i and ω_j agree on the actions affecting the common variables.

A solution for $\mathbb{C}$ is a set of subplans for the projected instances and we want to guarantee that it is possible to merge these subplans into one single action sequence without any ordering conflicts.

Lemma 7. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance such that DTG(v) is acyclic for all $v \in V$, let $\langle N, T \rangle$ be a tree decomposition of $CG(\mathbb{P})$ and let $\mathbb{C} = \langle X, C \rangle$ be the corresponding CSP instance according to Construction 6. If $\mathbb{C}$ has a solution with value ω_i for each $x_i \in X$, then there is an action sequence ω_T such that (1) $\omega_T[N_i] = \omega_i$ for each $x_i \in X$ and (2) every action a in ω_T occurs in ω_i for some $x_i \in X$.*

Proof. Suppose $\mathbb{C}$ has a solution with value ω_i for each $x_i \in X$. According to Lemma 4 no action occurs more than once in each ω_i, but the same action a can occur in more than one ω_i if two or more variables in eff(a) occur in different tree nodes. We can thus merge all the subplans ω_i into a new action sequence ω_T which contains all action of the subplans, but all occurences of the same action in different ω_i are merged into one, thus synchronizing the subplans. It remains to prove that the action ordering of all subplans can be respected in ω_T. Proof by induction over the number of nodes in T.

Base case: If there is only one node N_0, then $\omega_T = \omega_0$.

Induction: Suppose the claim holds for p nodes and let $N = \{N_0, \ldots, N_p\}$. Without losing generality, assume that N_0 is a leaf node in T and that N_1 is its only adjacent node. By assumption, $\omega_1, \ldots, \omega_p$ can be merged into a plan ω' such that $\omega'[N_i] = \omega_i$ for $1 \leq i \leq p$ and every action a in ω' occurs in some of the subplans $\omega_1, \ldots, \omega_p$. It remains to prove that also ω_0 can be merged with ω'.

The subplan $\omega_0[N_0 \cap N_1]$ can trivially be merged with ω' since the constraints of $\mathbb{C}$ guarantee that it is identical to $\omega_1[N_0 \cap N_1]$, which must be a subsequence of ω'. Suppose $v \in N_0 \setminus N_1$, i.e. $v \in N_0$ but $v \notin N_1$. The node N_0 is a leaf so v cannot occur in any other node than N_0 since the definition of tree decompositions requires that all nodes containing v form a subtree of T. That is, $(N_0 \setminus N_1) \cap N_i = \varnothing$ for $1 \leq i \leq p$.

Let a be an action in ω_0 and let v be a variable such that $v \in$ vars(eff(a)). Suppose $v \in N_0 \setminus N_1$, i.e. v occurs only in N_0. Let u be a variable such that $u \in$ vars(pre(a)). Then the edge $\langle u, v \rangle$ must be in $CG(\mathbb{P})$ so there must be some node N_i such that $\{u, v\} \subseteq N_i$. However, then $N_i = N_0$ since v only occurs in N_0. Since this holds for all such actions it follows that $\omega_0[N_0 \setminus N_1]$ only needs to be ordered with respect to $\omega_0[N_0 \cap N_1]$. However, this is trivial since ω_0 is a plan for $\mathbb{P}[N_0]$ and $\omega_0[N_0 \setminus N_1]$ and $\omega_0[N_0 \cap N_1]$ thus already have consistent action orders. This ends the induction.

We thus conclude that all subplans $\omega_0, \ldots, \omega_p$ can be merged into a single sequence ω_T such that $\omega_T[N_i] = \omega_i$ for $0 \leq i \leq p$ and every action a in ω_T occurs in some ω_i $(0 \leq i \leq p)$. □

We can now prove that Construction 6 is indeed a reduction from planning to CSP.

Lemma 8. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance such that DTG(v) is acyclic for all $v \in V$, let $\langle N, T \rangle$ be a tree decomposition of $CG(\mathbb{P})$ and let $\mathbb{C} = \langle X, C \rangle$ be the corresponding CSP instance according to Construction 6. Then $\mathbb{P}$ has a solution if and only if $\mathbb{C}$ has a solution.*

Proof. if: Suppose $\mathbb{C}$ has a solution with value ω_i for each $x_i \in X$. Let ω_T be a corresponding merged plan, which must exist according to Lemma 7. We first prove by induction over all prefixes of ω_T that ω_T is valid in s_I.

Base case: The empty prefix is always valid in s_I.

Induction: Suppose all prefixes of ω_T of length at most p are valid in s_I and let $\langle a_1, \ldots, a_{p+1} \rangle$ be a prefix of ω_T. The sequence $\langle a_1, \ldots, a_p \rangle$ is valid in s_I by assumption, so there is a sequence of states $s_0, \ldots, s_p$, where $s_0 = s_I$, such that s_i is the result of $\langle a_1, \ldots, a_i \rangle$ in s_I for all i $(1 \leq i \leq p)$. We must thus prove that a_{p+1} is valid in s_p. This trivially holds if a_{p+1} has an empty precondition, so assume that vars(pre(a_{p+1})) is not empty. Let u and v be two arbitrary variables such that $u \in$ vars(pre(a_{p+1})) and $v \in$ vars(eff(a_{p+1})). Also let $x \in D(u)$ and $y \in D(v)$ be values such that pre$(a_{p+1})[u] = x$ and eff$(a_{p+1})[v] = y$. We must prove that $s_p[u] = x$. We do this by proving (1) that $s_q[u] = x$ for some $q \leq p$ and (2) that there is no r such that $q < r \leq p$ and $u \in$ vars(eff(a_r)) but eff$(a_r)[u] \neq x$

Condition (1) trivially holds if $s_I[u] = x$, so assume this is not the case. Obviously, $CG(\mathbb{P})$ contains the edge $\langle u, v \rangle$ so there must be some node N_i in T such that $\{u, v\} \subseteq N_i$. The action $a_{p+1}[N_i]$ must thus occur in ω_i and ω_i is a plan for $\mathbb{P}[N_i]$, so there must be some action a' such that eff$(a')[u] = x$ and a' is ordered before $a_{p+1}[N_i]$ in ω_i. It follows from Lemma 7 that there is some a_q in ω_T such that $1 \leq q \leq p$ and eff$(a_q)[u] = x$. Hence, $s_q[u] = x$.

For condition (2), assume the opposite, that there is also some r such that $q < r \leq p$ and eff$(a_r) = z$ for some value $z \in D(u)$ where $z \neq x$. It follows from Lemma 7 that ω_i must contain all actions in ω_T that affect u or v. Hence, the actions $a_q[N_i]$, $a_r[N_i]$ and $a_{p+1}[N_i]$ must occur in ω_i in that order. However, this contradicts that ω_i is a plan for $\mathbb{P}[N_i]$ so we conclude that there can be no r satisfying the assumption.

It follows that $s_p[u] =$ pre$(a_{p+1})[u]$. Since u was choosen arbitrarily in vars(pre(a_{p+1})) it follows that a_{p+1} must be valid in s_p and, thus, that $\langle a_1, \ldots, a_{p+1} \rangle$ is valid in s_I, ending the induction.

By definition each ω_i is a plan from $s_I[N_i]$ to $s_G[N_i]$. Since ω_T is valid in s_I also every subplan ω_i is valid in s_I. Hence, each ω_i is a plan from s_I to $s_G[N_i]$. Furthermore, $N_1 \cup \ldots \cup N_{|N|} = V$ so since the merge ω_T of all ω_i is valid in s_I, it follows that ω_T is a plan from s_I to s_G.

only if: Suppose ω is a plan for $\mathbb{P}$. Let N_i and N_j be two arbitrary adjacent nodes in T. Also let $\omega_i = \omega[N_i]$ and $\omega_j = \omega[N_j]$. It is a property of projections that ω_i is a plan for $\mathbb{P}[N_i]$ and ω_j is a plan for $\mathbb{P}[N_j]$. Obviously, $\omega_i[N_i \cap N_j] = \omega_j[N_i \cap N_j]$. It follows that $\mathbb{C}$ must have a solution with value $\omega[N_i]$ for each $x_i \in X$. □

This leads to the main theorem.

Theorem 9. *Let C be a class of planning instances such that for every instance $\mathbb{P} \in C$, all DTGs of $\mathbb{P}$ are acyclic. Then* PLANSAT$(C, \langle d, k, w \rangle)$ *is fixed-parameter tractable.*

Proof. First compute an optimal tree decomposition $\langle X, T \rangle$ for $CG(\mathbb{P})$, which has width w by assumption. This is fpt in w [5].

Then for each $N_i \in N$, construct the transition graph $\text{TG}_i = \text{TG}(\mathbb{P}[N_i])$ from the DTGs DTG(v) for $v \in N_i$. The set of paths from $s_I[N_i]$ to $s_G[N_i]$ in TG_i corresponds directly to the set of plans for $\mathbb{P}[N_i]$, so this set of paths is the domain $D(x_i)$ for the corresponding CSP variable x_i. Since $|N_i| \leq w + 1$, there are at most

d^{w+1} vertices in TG_i. We can assume that all irrelevant edges have been removed from the DTGs so all edges appear along some path. Since there are at most k paths in each DTG, there can be at most k^{w+1} paths in TG_i. Each of these paths is of length $d^{w+1} - 1$ at most, so TG_i has at most $(kd)^{w+1}$ edges. The set of paths can thus be computed in time $O((kd)^{w+1} + k^{w+1}) = O((kd)^{w+1})$, using Eppsteins algorithm [16]. If there are n variables in V then there are at most n nodes in T so computing the sets of paths for all nodes takes time $O((kd)^{w+1} \cdot n)$, which is fpt in $\langle k, d, w\rangle$ since n is upper bounded by the instance size.

For each pair of adjacent nodes $N_i, N_j \in N$, there are at most $(kd)^{2(w+1)}$ pairs of paths for TG_i and TG_j. Since the path length is bounded by d^{w+1} two paths can be compared in time $O(d^{w+1})$. It follows that the relation $R_{i,j}$ can be computed in time $O((kd)^{2(w+1)}d^{w+1}) \subseteq O((kd)^{3(w+1)})$. There are at most n such relations since T is a tree, so computing all relations takes time $O((kd)^{3(w+1)} \cdot n)$, which is fpt in $\langle k, d, w\rangle$.

The relations are precomputed and explicitly represented, so it suffices to use indices instead of actual paths, i.e. the relations of $\mathbb{C}$ can be over $\mathbb{N} \times \mathbb{N}$. All relations are binary and the constraint graph of $\mathbb{C}$ is a tree. As previously noted, the domain size of each variable is at most k^{w+1} and there are at most n variables. Hence, using the result by Dechter and Pearl [11] it follows that $\mathbb{C}$ can be solved in time $O((k^{w+1})^2 n) = O(k^{2(w+1)} \cdot n)$, which is fpt in $\langle k, w\rangle$.

This whole construction is thus fpt in the parameter $\langle k, d, w\rangle$ so it follows from Lemma 8 that PLANSAT is fpt in the parameter $\langle k, d, w\rangle$ for this case. □

The special case of polytree causal graphs follows.

Corollary 10. *Let C be a class of planning instances such that for every instance $\mathbb{P} \in C$, all DTGs are acyclic and $CG(\mathbb{P})$ is a polytree. Then* PLANSAT$(C, \langle k\rangle)$ *is fixed-parameter tractable.*

Proof. If $CG(\mathbb{P})$ is a polytree, then it has treewidth $w = 1$ since $U(CG(\mathbb{P}))$ is a tree. We will also use the fact that the domain size is upper bounded by the instance size. Each graph TG_i has at most $d^{w+1} = d^2$ vertices and at most $k^{w+1} = k^2$ paths, each of length $d^2 - 1$ at most. Hence, the set of paths can be computed in time $O(kd)^{w+1}n) = O((kd)^2 n) = O(k^2 \cdot d^2 n)$, which is fpt in k since d is upper bounded by the instance size. Computing the relations takes time $O((kd)^{3(w+1)}n) = O(k^6 \cdot d^6 n)$, which is fpt in k. Solving $\mathbb{C}$ takes time $O(k^{2(w+1)}n) = O(k^4 \cdot n)$, which is fpt in k. Hence, the whole computation is fpt in k. □

In the general case, d must be a parameter since it occurs in expressions on the form d^{w+1} and cannot be separated from parameter w. In the polytree case, this no longer holds since w is constant.

5 PLAN OPTIMISATION

In this section, we will show that the two previous fpt results hold also for finding the length of an optimal plan.

Definition 11. An instance of the *constraint satisfaction optimization problem (CSOP)* is a CSP instance $\mathbb{C} = \langle X, D, C, W\rangle$ with the additional parameter W which contains a weight (or cost) function $w : D(x) \to \mathbb{Q}_{\geq 0}$ for each $x \in X$. The weight of an instantiation α is defined as $w(\alpha) = \sum_{x \in X} w(\alpha(x))$. A solution to $\mathbb{C}$ is either the answer 'no', if there is no satisfying instantiation, or the minimum value of $w(\alpha)$ over all satisfying instantiations α.

Färnqvist [18, Corollary 13]) has shown that solving CSOP is fpt if the parameters are the maximum domain size of the variables and the treewidth of the constraint graph.

The optimisation variant of our main theorem relies on the same construction as previously, but with some additional details.

Theorem 12. *Let C be a class of planning instances such that for every instance $\mathbb{P} \in C$, all DTGs are acyclic. Then* PLANOPT$(C, \langle d, k, w\rangle)$ *is fixed-parameter tractable.*

Proof sketch. The subplans for the nodes in Construction 6 can overlap so we must avoid counting an action multiple times. Identify a subset $A_i \subseteq A$ with each node $N_i \in N$, defined as follows. For each action $a \in A$, arbitrarily choose one variable $v \in \text{vars}(\text{eff}(a))$ and arbitrarily choose one node $N_i \in N$ such that $v \in N_i$ and let a belong to A_i. Obviously, $\{A_i \mid N_i \in N\}$ is a partition on A. For each constraint variable x_i, define the weight function such that the weight of a solution ω_i for $\mathbb{P}[N_i]$ is the number of actions in ω_i that belong to A_i. This guarantees that every action that is used in some subplan in a solution is counted exactly once. Since the domain size of this CSOP is upper bounded by k^{w+1}, it follows from the proof of the previous theorem and Färnqvists Corollary that computing the optimal plan length is fpt in the parameter $\langle k, d, w\rangle$. □

The polytree case follows also here.

Corollary 13. *Let C be a class of planning instances such that for every instance $\mathbb{P} \in C$, all DTGs are acyclic and $CG(\mathbb{P})$ is a polytree. Then* PLANOPT$(C, \langle k\rangle)$ *is fixed-parameter tractable.*

6 DISCUSSION

Requiring acyclic DTGs may seem very restrictive, yet it is sufficient for certain important cases. For instance, some of the most successful heuristics for STRIPS planning are based on the *delete relaxation* heuristic (h^+) [23]. This method first removes all negative effects of actions and then takes the length of an optimal plan for this relaxed problem as the heuristic measure for the real plan length. However, even delete relaxed planning is **NP**-complete so it needs to be somehow approximated in practice. Generalisations of delete relaxed planning to non-binary variables have received attention in the literature recently [13, 24] and our parameter k seems more useful in this case, than in the STRIPS case. The DTGs are always acyclic for delete-relaxed instances but they may have loops, i.e. one-vertex cycles consisting of a single edge $\langle v, a, v\rangle$. The method described in this paper is still applicable, with a minor modification. First use Eppsteins algorithm as previously described, ignoring the loops. Then split each path into different paths whenever the loops give a choice, as follows. A loop $\langle v, a, v\rangle$ at a vertex v can only occur if there is an incoming edge $\langle u, a, v\rangle$. Furthermore, no action is required more than once in an optimal plan, so there is no need to follow a loop labelled with an action that already occurs in the path. Hence, for each vertex along the path where there is a choice of following loops or not, split the path into different paths for all possible choices, obeying that each action occurs at most once. During this process, ignore duplicates and keep track of the number of paths and break if a split results in a total number of paths larger than k. The trick here is that we never need to follow a loop more than once. Hence, our results imply that delete relaxed planning is fpt even in the generalised case, even though it is **NP**-complete already for binary domains.

Although not the topic of this paper, there are connections with so-called *factored planning*, which is the idea of partitioning the variables, solving the subproblems independently and then using CSP

techniques for finding a global solution. Our approach is clearly related since we use CSP techniques for the proofs, although CSP is just a tool that happens to be useful in our case. There is also the difference that papers on factored planning typically focus on algorithms that are intended to solve planning in general by employing CSP techniques [7], although there are also attempts to identify tractable subcases [17], in the classical sense. Brafman and Domshlak [8] provide more detailed complexity figures, using treewidth and other parameters, although not attempting a parameterised analysis and the expressions they arrive at are not fpt in these parameters. Some of the parameters are also difficult to test in advance. For instance, their local work parameter Δ is defined as

$$\Delta = \min_{\omega \in \mathit{Plans}(\mathbb{P})} \max_{v \in V} |\omega[v]|$$

in our notation, where $\mathit{Plans}(\mathbb{P})$ denotes the set of plans for $\mathbb{P}$. Finding the value of Δ is unfortunately as hard as planning itself.

Theorem 14. *Deciding the value of Δ is* **PSPACE***-complete and* ***W****[2]-complete.*

Proof sketch. Let $\mathbb{P}$ be an arbitrary planning instance. Construct a new instance $\mathbb{P}'$ by adding a new variable v that every action switches from 0 to 1. Also add a new action that can switch v back to 0. Any plan for $\mathbb{P}'$ must have twice as many actions as the corresponding plan for $\mathbb{P}$ and the value of Δ for $\mathbb{P}'$ equals the length of an optimal plan for $\mathbb{P}'$. Hence, deciding Δ is as hard as deciding the optimal plan length, which is **PSPACE**-complete [9] and **W**[2]-complete [2]. □

Since we have an fpt result for planning with arbitrary causal graphs, future work should focus on non-acyclic DTGs. The k parameter may still be useful, but additional parameters must be used. Katz and Keyder [27] also use the number of paths in a DTG, but consider general DTGs and first collapse all strongly connected components (SCCs) of the graph and then count the paths in the resulting graph. This can be applied also in our case, but it would be necessary to find additional parameters also for the SCCs, since different cycles in an SCC may require different preconditions for the actions, causing intricate interplay between the variables. Parameters that immediately come to mind are the maximum size of an SCC and the maximum number of cycles in an SCC. Another interesting approach is to allow paths to contain cycles under certain restrictions and then still count the number of such paths. It may then be beneficial to model the paths as regular expressions or even model the DTGs as automata, as is sometimes done in factored planning [17]. Yet another possibility is to allow restricted appearance of cycles in the DTGs and use a parameter to bound the number of times a cycle may be followed. In that case, a very simple straightforward method could be to replace each cycle with a path corresponding to this number of turns through the cycle, somewhat similar to loop unrolling in compilers.

REFERENCES

[1] Stefan Arnborg, Derek Corneil, and Andrzej Proskurowski, 'Complexity of finding embeddings in a k-tree', *SIAM J. Alg. Disc. Meth.*, **8**(2), 277–284, (1987).

[2] Christer Bäckström, Yue Chen, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider, 'The complexity of planning revisited - a parameterized analysis', in *Proc. 26th AAAI Conf. Artif. Intell. (AAAI 2012), Toronto, ON, Canada*, pp. 1735–1741, (2012).

[3] Christer Bäckström and Peter Jonsson, 'A refined view of causal graphs and component sizes: SP-closed graph classes and beyond', *J. Artif. Intell. Res.*, **47**, 575–611, (2013).

[4] Christer Bäckström and Bernhard Nebel, 'Complexity results for SAS^+ planning', *Computat. Intell.*, **11**, 625–656, (1995).

[5] Hans L. Bodlaender, 'A linear-time algorithm for finding tree-decompositions of small treewidth', *SIAM J. Comput.*, **25**(6), 1305–1317, (1996).

[6] Ronen I. Brafman and Carmel Domshlak, 'Structure and complexity in planning with unary operators', *J. Artif. Intell. Res.*, **18**, 315–349, (2003).

[7] Ronen I. Brafman and Carmel Domshlak, 'Factored planning: How, when, and when not', in *Proc. 21st Nat'l Conf. Artif. Intell. (AAAI 2006), Boston, MA, USA*, pp. 809–814, (2006).

[8] Ronen I. Brafman and Carmel Domshlak, 'On the complexity of planning for agent teams and its implications for single agent planning', *Artif. Intell.*, **198**, 52–71, (2013).

[9] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *Artif. Intell.*, **69**(1–2), 165–204, (1994).

[10] Ronald de Haan, Anna Roubícková, and Stefan Szeider, 'Parameterized complexity results for plan reuse', in *Proc. 27th AAAI Conf. Artif. Intell. (AAAI 2013), Bellevue, WA, USA*, (2013).

[11] Rina Dechter and Judea Pearl, 'Tree clustering for constraint networks', *Artif. Intell.*, **38**(3), 353–366, (1989).

[12] Carmel Domshlak and Yefim Dinitz, 'Multi-agent off-line coordination: Structure and complexity', in *Proc. 6th Eur. Conf. Planning (ECP'01), Toledo, Spain*, (2001).

[13] Carmel Domshlak and Adeline Nazarenko, 'The complexity of optimal monotonic planning: The bad, the good, and the causal graph', *J. Artif. Intell. Res.*, **48**, 783–812, (2013).

[14] R. Downey, M. Fellows, and U. Stege, *Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability*, volume 49 of *DIMACS Series in Disc. Math. Theor. Comput. Sci.*, 49–99, 1999.

[15] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Monographs in Computer Science, Springer, New York, 1999.

[16] David Eppstein, 'Finding the k shortest paths', *SIAM J. Comput.*, **28**(2), 652–673, (1998).

[17] Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux, 'Cost-optimal factored planning: Promises and pitfalls', in *Proc. 20th Int'l Conf. Automated Planning and Scheduling (ICAPS 2010), Toronto, ON, Canada, May 12-16, 2010*, pp. 65–72, (2010).

[18] Tommy Färnqvist, 'Constraint optimization problems and bounded tree-width revisited', in *Proc. 9th Int'l Conf. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimzation Problems (CPAIOR 2012), Nantes, France.*, pp. 163–179, (2012).

[19] Jörg Flum and Martin Grohe, *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*, Springer, Berlin, 2006.

[20] Omer Giménez and Anders Jonsson, 'The complexity of planning problems with simple causal graphs', *J. Artif. Intell. Res.*, **31**, 319–351, (2008).

[21] Omer Giménez and Anders Jonsson, 'Planning over chain causal graphs for variables with domains of size 5 is NP-hard', *J. Artif. Intell. Res.*, **34**, 675–706, (2009).

[22] Malte Helmert, 'A planning heuristic based on causal graph analysis', in *Proc. 14th Int'l Conf. Automated Planning and Scheduling (ICAPS 2004), Whistler, BC, Canada*, pp. 161–170, (2004).

[23] Jörg Hoffmann, 'Where 'ignoring delete lists' works: Local search topology in planning benchmarks', *J. Artif. Intell. Res.*, **24**, 685–758, (2005).

[24] Jörg Hoffmann, 'Analyzing search topology without running any search: On the connection between causal graphs and h^+', *J. Artif. Intell. Res.*, **41**, 155–229, (2011).

[25] Michael Katz and Carmel Domshlak, 'New islands of tractability of cost-optimal planning', *J. Artif. Intell. Res.*, **32**, 203–288, (2008).

[26] Michael Katz and Carmel Domshlak, 'Implicit abstraction heuristics', *J. Artif. Intell. Res.*, **39**, 51–126, (2010).

[27] Michael Katz and Emil Keyder, 'Structural patterns beyond forks: Extending the complexity boundaries of classical planning', in *Proc. 26th AAAI Conf. Artif. Intell. (AAAI 2012), Toronto, ON, Canada*, pp. 1779–1785, (2012).

[28] Martin Kronegger, Andreas Pfandler, and Reinhard Pichler, 'Parameterized complexity of optimal planning: A detailed map', in *Proc. 23rd Int'l Joint Conf. Artif. Intell. (IJCAI 2013), Beijing, China*, pp. 954–961, (2013).