Novel Structural Parameters for Acyclic Planning Using Tree Embeddings

Christer Bäckström[1], **Peter Jonsson**[1], **Sebastian Ordyniak**[2]
[1] Department of Computer and Information Science, Linköping University, Sweden
[2] Algorithms group, University of Sheffield, UK
christer.backstrom@liu.se, peter.jonsson@liu.se, sordyniak@gmail.com

Abstract

We introduce two novel structural parameters for acyclic planning (planning restricted to instances with acyclic causal graphs): *up-depth* and *down-depth*. We show that cost-optimal acyclic planning restricted to instances with bounded domain size and bounded up- or down-depth can be solved in polynomial time. For example, many of the tractable subclasses based on polytrees are covered by our result. We analyze the parameterized complexity of planning with bounded up- and down-depth: in a certain sense, down-depth has better computational properties than up-depth. Finally, we show that computing up- and down-depth are fixed-parameter tractable problems, just as many other structural parameters that are used in computer science. We view our results as a natural step towards understanding the complexity of acyclic planning with bounded treewidth and other parameters.

1 Introduction

Causal graphs are widely used in planning for analyzing planning instances. In particular, instances with acyclic causal graphs (which we refer to as *acyclic planning*) have attracted a great deal of attention. One should note that satisficing acyclic planning is PSPACE-complete [Jonsson *et al.*, 2014], i.e. it is as hard as unrestricted propositional planning. Many tractability results for acyclic planning have appeared in the literature—examples include [Williams and Nayak, 1997; Jonsson and Bäckström, 1998; Brafman and Domshlak, 2003; Giménez and Jonsson, 2008]—but they have also been utilized in decomposing planning instances into action hierarchies [Knoblock, 1994; Bacchus and Yang, 1994] and to compute domain-independent heuristics [Helmert, 2006; Katz and Domshlak, 2010]. In recent years, *polytree* causal graphs have become popular for defining tractable subclasses [Katz and Keyder, 2012; Katz and Domshlak, 2010; Aghighi *et al.*, 2015; Ståhlberg, 2017]. A directed graph is a polytree if its underlying undirected graph is a connected tree. The work on polytree causal graphs has lead to interesting work concerning for instance *structural pattern heuristics* [Katz and Domshlak, 2010]. The usefulness of such heuristics increases with the availability of large causal graph fragments such that cost-optimal planning can be solved in polynomial time. Graph fragments such as *forks* and *inverted forks* were used quite early in this research and have gradually evolved into more expressive types of graphs.

Due to the highly restricted structure of polytree, it is natural to try to find new tractability results that are not based on polytrees. To this end, we suggest two structural parameters *up-depth* and *down-depth* in this paper. Structural parameters based on graph representations are very common in computer science and are becoming increasingly used in artificial intelligence: examples include SAT and constraint satisfaction [Paulusma *et al.*, 2016; Ganian *et al.*, 2017; Gaspers and Szeider, 2013], knowledge representation [Gottlob *et al.*, 2010], and answer set programming [Bliem *et al.*, 2017; 2016]. An acyclic graph D has up-depth k if it can be embedded into a transitive up-tree of height k, and down-depth is defined analogously using transitive down-trees. One may note that forks have up-depth 1 and inverted forks have down-depth 1. One should also note that every directed graph has a well-defined up-depth and down-depth.

We demonstrate that our new structural parameters can be used for obtaining new tractability results: cost-optimal acyclic planning restricted to instances with bounded domain size and bounded up- or down-depth can be solved in polynomial time. Naturally, this can be viewed as an alternative proof of the fact that cost-optimal planning with (inverted) forks and bounded domains can be solved in polynomial time. The important difference between this result and polytree based classes is that our algorithm is applicable to *every* acyclic graph. Another related result is presented by Brafman and Domshlak [2006]. They prove that acyclic planning is tractable when both the treewidth of the causal graph and the local depth are bounded. The local depth is a bound on the number of variable changes in an optimal solution, and this parameter may be prohibitively difficult to estimate.

From a parameterized complexity view, we show that cost-optimal planning with bounded down-depth and bounded domain is fixed-parameter tractable while planning with bounded up-depth is W[1]-hard. This implies that planning with bounded up-depth is computationally more expensive under standard complexity assumptions. In fact, we show that our algorithm for bounded down-depth runs in linear time with respect to the number of variables. We also show that computing the up- and down-depth is a fixed-parameter tractable problem, allowing us to compute the parameters quite efficiently, at least when they are small.

The paper is structured as follows: Section 2 contains basic definitions and Section 3 introduces our structural parameters and shows how to compute the parameters. Section 4 and Section 5 contain the algorithms for planning with bounded up- and down-width, respectively, together with the results concerning parameterized complexity.

2 Preliminaries

We refer to the handbook by Diestel (2012) for standard graph terminology. Let T be an undirected rooted tree and $v \in V(T)$. We denote by $U_T(v)$ and $L_T(v)$ the set of all ancestors respectively descendants of v in T and by $U_T[v]$ and $L_T[v]$ the sets $U_T(v) \cup \{v\}$ and $L_T(v) \cup \{v\}$, respectively. We denote by T_v the subtree of T rooted at v. The *depth* of v in T, denoted by $w_T(v)$, is the number of vertices on the unique path from v to the root of T and the *height* of v in T is equal to the maximum number of vertices from v to any leaf of T_v. The *depth* and *height* of T, denoted by w_T respectively h_T, are the maximum depth respectively height of any vertex in T. Note that the above definitions can be naturally extended to rooted forests instead of trees, where a rooted forest is a forest consisting of a disjoint union of rooted trees. For a directed graph G we denote by $\widetilde{G}$ its underlying undirected graph.

2.1 Sequences

Let s be a sequence. We denote by $s[i]$ the i-th element of s for every i with $1 \leq i \leq |s|$. We denote by $\mathsf{cp}(s)$ the *compressed sequence* of s, i.e., the sub-sequence of s obtained after removing every element that is equal to its successor in s. We call a sequence s *compact* if $\mathsf{cp}(s) = s$. We say that a sequence s is an *extension* of a sequence s' if $\mathsf{cp}(s) = s'$.

Let X be a finite set and let S be a set of sequences over X. We denote by $\mathsf{LCSS}(S)$ the length of a longest sequence s_L (over X) such that: (S1) every sequence $s \in S$ is a sub-sequence of s_L and (S2) no sub-sequence of s_L satisfies (S1). For a given integer l and set X we are interested in the maximum number, in the following denoted by $\mathsf{TLCSS}(l, |X|)$, such that there is a set S of sub-sequences over X each having length at most l such that $\mathsf{TLCSS}(l, |X|) = \mathsf{LCSS}(S)$.

Proposition 1. *Let X be a finite set and l a natural number. Then $\mathsf{TLCSS}(l, |X|) \leq |X|^l l$.*

2.2 Parameterized Complexity

In parameterized algorithmics [Downey and Fellows, 2013] the run-time of an algorithm is studied with respect to a parameter $k \in \mathbb{N}$ and input size n. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Problems that can be solved in this time are called *fixed-parameter tractable* (fpt). Moreover, we say that a problem is *linear-time fixed-parameter tractable* if it can be solved in time $f(k) \cdot n$. Aside from the complexity class FPT, we will also make use of the complexity class $\mathsf{W}[1]$ (proving hardness for $\mathsf{W}[1]$ is often used to show that a problem is unlikely to be in FPT).

2.3 Planning

Let $V = \{v_1, \dots, v_n\}$ be a finite set of *variables* over a finite *domain* D and let $D^+ = D \cup \{\mathbf{u}\}$, where $\mathbf{u}$ is a special "undefined" value not present in D. Then D^n is the set of *total states* and $(D^+)^n$ is the set of *partial states* over V and D. Clearly, $D^n \subseteq (D^+)^n$. The value of a variable v in a state $s \in (D^+)^n$ is denoted by $s\langle v\rangle$ and the projection of a state s to a set of variables $V' \subseteq V$ is denoted by $s\langle V'\rangle$. We say that a total state t over some set of variables V' is *compatible* with a partial state p over V', denoted by $t \sim p$ if for every $v \in V'$ either $p\langle v\rangle = \mathbf{u}$ or $p\langle v\rangle = t\langle v\rangle$. A *SAS$^+$ instance* is a tuple $\mathbb{P} = \langle V, D, A, I, G, \Delta\rangle$ where V is a set of variables, D is a domain, A is a set of *actions*, $I \in D^n$ is the *initial state*, $G \in (D^+)^n$ is the (partial) *goal state*, and $\Delta : A \rightarrow \mathbb{Q}_{\geq 0}$ is a cost function for the actions. Each action $a \in A$ has a *precondition* $\mathsf{pre}(a) \in (D^+)^n$ and an *effect* $\mathsf{eff}(a) \in (D^+)^n$. We will frequently use the convention that a variable has value $\mathbf{u}$ in a precondition/effect unless a value is explicitly specified. Let $a \in A$ and $s \in D^n$. Then a is *valid in* s if s is compatible with $\mathsf{pre}(a)$. Furthermore, the *result of* a *in* s is a state $t \in D^n$ defined such that for all $v \in V$, $t\langle v\rangle = \mathsf{eff}(a)\langle v\rangle$ if $\mathsf{eff}(a)\langle v\rangle \neq \mathbf{u}$ and $t\langle v\rangle = s\langle v\rangle$ otherwise. Let $s_0, s_\ell \in D^n$ and let $\omega = \langle a_1, \dots, a_\ell\rangle$ be a sequence of actions (of length ℓ). Then ω is a *plan from s_0 to s_ℓ* if either (i) $\omega = \langle\rangle$ and $\ell = 0$, or (ii) there are states $s_1, \dots, s_{\ell-1} \in D^n$ such that for all $1 \leq i \leq \ell$, a_i is valid in s_{i-1} and s_i is the result of a_i in s_{i-1}. A state $s \in D^n$ is a *goal state* if for all $v \in V$, either $G\langle v\rangle = s\langle v\rangle$ or $G\langle v\rangle = \mathbf{u}$. An action sequence ω is a *plan for* $\mathbb{P}$ if ω is a plan from I to some goal state. We say a plan ω is minimal for $\mathbb{P}$ if no proper sub-sequence of ω is a plan for $\mathbb{P}$. The *cost* of a plan ω, denoted by $\Delta(\omega)$, is equal to the sum of the cost of all actions in ω, i.e., $\Delta(\omega) = \sum_{i=1}^{|\omega|} \Delta(\omega[i])$. We say that ω is *cost-optimal* for $\mathbb{P}$ if there is no plan ω' for $\mathbb{P}$ with $\Delta(\omega') < \Delta(\omega)$. We will consider the following computational problems for planning. Given a SAS$^+$ instance $\mathbb{P} = \langle V, D, A, I, G, \Delta\rangle$. PLANNING is the problem of deciding whether $\mathbb{P}$ has a plan and COST OPTIMAL PLANNING is the problem of deciding whether $\mathbb{P}$ has a plan of cost at most c for some given number $c \in \mathbb{Q}_{\geq 0}$.

Let V' be a subset of V. For an action $a \in A$ we denote by $a\langle V'\rangle$ the action a projected to the variables in V', i.e., $a\langle V'\rangle$ has precondition $\mathsf{pre}(a)\langle V'\rangle$ and effect $\mathsf{eff}(a)\langle V'\rangle$ and we denote by $A\langle V'\rangle$ the set $\{\, a\langle V'\rangle \mid a \in A \,\}$. The planning instance induced by V', denoted by $\mathbb{P}[V']$, is the instance $\langle V', D, A\langle V'\rangle, I\langle V'\rangle, G\langle V'\rangle, \Delta\rangle$. Since there is a one-to-one correspondence between A and $A\langle V'\rangle$, we will often not distinguish between them; for instance we will say that a sequence of actions in A is a plan for $\mathbb{P}[V']$, meaning that the corresponding actions in $A\langle V'\rangle$ are a plan for $\mathbb{P}[V']$. We denote by $A_{V'}$ the set of actions $\{\, a \in A \mid \mathsf{eff}(a)\langle V'\rangle \neq \mathbf{u} \,\}$. For a sequence ω of actions we denote by $\omega\langle V'\rangle$ the restriction of ω to actions in $A_{V'}$, i.e., $\omega\langle V'\rangle$ is obtained from ω after removing all actions not in $A_{V'}$. We often use v as a shortcut for $\{v\}$. For a sequence ρ of (partial) states, we denote by $\mathsf{proj}(\rho, V')$ the sequence $\langle \rho[1]\langle V'\rangle, \dots, \rho[|\rho|]\langle V'\rangle\rangle$. We say that a sequence ρ of total states over V' is *compatible* with a sequence ρ' of partial states over V', denoted by $\rho \sim \rho'$ if $|\rho| = |\rho'|$ and $\rho[i] \sim \rho'[i]$ for every i with $1 \leq i \leq |\rho|$.

For a sequence of actions ω, we denote by $\mathsf{PR}_{\mathbf{u}}(\omega, V')$ the sequence of preconditions of the actions in ω on the variables in V', i.e., the sequence $\mathsf{proj}(\langle \mathsf{pre}(a_1), \dots, \mathsf{pre}(a_l)\rangle, V')$ and we denote by $\mathsf{PR}(\omega, V')$ the sequence obtained from $\mathsf{PR}_{\mathbf{u}}(\omega, V')$ after removing all states that are completely undefined. For a plan ω for $\mathbb{P}$, we denote by $\mathsf{ST}_{\mathbb{P}}(\omega)$ the sequence of all states achieved during the execution of ω on $\mathbb{P}$, including the initial and the goal state. Moreover, for two subsets V_1 and V_2 of V, we denote by $\mathsf{ST}_{\mathbb{P}}(\omega, V_1)$ the sub-sequence of $\mathsf{ST}_{\mathbb{P}}(\omega)$ retaining only the states that were achieved before the execution of an action in A_{V_1}, i.e., the sequence $\mathsf{ST}_{\mathbb{P}}(\omega)[\alpha(1)], \dots, \mathsf{ST}_{\mathbb{P}}(\omega)[\alpha(|\omega'|)]$ (where $\alpha = \mathsf{subF}\langle \omega, \omega' \rangle$ and $\omega' := \omega\langle V_1\rangle$), and we denote by $\mathsf{ST}_{\mathbb{P}}(\omega, V_1, V_2)$ the sequence $\mathsf{proj}(\mathsf{ST}_{\mathbb{P}}(\omega, V_1), V_2)$. We omit the subscript if the planning instance is clear from the context.

The *causal graph* of $\mathbb{P}$, denoted by $\mathcal{G}_C(\mathbb{P})$, has vertex set V and contains a directed arc from v to u if there is an action $a \in A$ such that $\mathsf{eff}(a)\langle u \rangle \neq \mathbf{u}$ and either $\mathsf{pre}(a)\langle v \rangle \neq \mathbf{u}$ or $\mathsf{eff}(a)\langle v \rangle \neq \mathbf{u}$. Informally, the causal graph models dependencies between planning variables given by the actions. We set $a_{\max} := \max_{v \in V} |A_v|$, i.e., $a_{\max}$ is the maximum number of actions effecting any variable.

3 Introducing Up-depth and Down-depth

In this section we will introduce our novel decompositional parameters for planning. Our parameters, which we call up-depth and down-depth can be seen as natural directed versions of the well-known decompositional parameter treedepth [Nesetril and de Mendez, 2012]. The *treedepth* of an undirected graph H, denoted by $\mathrm{td}(H)$, is the smallest natural number k such that there is an undirected rooted forest F with vertex set $V(H)$ of height at most k for which H is a subgraph of $C(F)$, where $C(F)$ is called the *closure* of F and is the undirected graph with vertex set $V(F)$ having an edge between u and v if and only if u is an ancestor of v in F. Informally a graph has treedepth at most k if it can be embedded in the closure of a forest of height k.

Our parameters *up-depth* and *down-depth* are now defined very similar to treedepth; the main difference is that we are using upward-closures and downward-closures instead of (undirected) closures. That is the *upward-closure/downward-closure* of an undirected rooted forest F, denoted by $C^{\uparrow}(F)$/$C^{\downarrow}(F)$, is the directed graph with vertex set $V(F)$ having an edge from u to v if v is an ancestor/descendant of u in F. The *up-depth/down-depth* of a directed acyclic graph G is then the smallest natural number k such that there is an undirected rooted forest F with vertex set $V(D)$ and height at most k such that D is a subgraph of $C^{\uparrow}(F)$/$C^{\downarrow}(F)$. Note that our notions of up-depth and down-depth require that the graph is acyclic since the upward and downward closure of a forest is always acyclic. On the other hand every acyclic graph G has up-depth and down-depth at most $|V(G)|$ since any acyclic graph is a subgraph of the upward and downward closure of an undirected path, whose vertices are ordered against/according to the topological ordering of G. The following theorem shows the relationship between our new parameters and treedepth.

Theorem 2. *Let G be a directed acyclic graph. Then* $\mathrm{td}(\widetilde{G}) \leq \min\{\mathsf{dep}^{\uparrow}(G), \mathsf{dep}^{\downarrow}(G)\}$.

Proof. The theorem follows immediately from the observation that if a directed acyclic graph G is a subgraph of either $C^{\downarrow}(F)$ or $C^{\downarrow}(F)$ of the rooted undirected forest F, then $\widetilde{G}$ is also a subgraph of $C(F)$. $\square$

In order to employ the new parameters for our algorithms for planning, we first need to be able to compute them. Namely we want to be able to solve the following problem.

UP-DEPTH	
Input:	A directed acyclic graph G and a natural number k.
Parameter:	k
Question:	Decide whether $\mathsf{dep}^{\uparrow}(G) \leq k$ and if so output a undirected forest witnessing this.

The corresponding problem for computing the down-depth, which we call the DOWN-DEPTH problem, is analogously defined by replacing $\mathsf{dep}^{\uparrow}(G)$ with $\mathsf{dep}^{\downarrow}(G)$ in the above definition.

The following theorem shows that computing up-depth/down-depth is NP-complete, however, as we will see below (as it is also the case for their undirected counterpart treedepth) both parameters can be efficiently computed if the input graph has small depth, which will be sufficient for our algorithms for COST OPTIMAL PLANNING.

Theorem 3. DOWN-DEPTH *and* UP-DEPTH *are* NP *-complete.*

Proof Sketch. We reduce from the corresponding problem for treedepth, which is well known to be NP -complete [Bodlaender *et al.*, 1998]. Since the proof is almost identical for DOWN-DEPTH and UP-DEPTH, we will only give it for DOWN-DEPTH. Hence let H be a connected undirected graph and let G be the directed graph obtained from H after replacing every edge $\{u, w\} \in E(H)$ with a new vertex v_e and the two arcs (u, v_e) and (w, v_e). One can now show that $\mathrm{td}(H) + 1 = \mathsf{dep}^{\downarrow}(G)$, which concludes the proof of the theorem. $\square$

We now provide our main theorem of this section showing that both up-depth and down-depth are linear-time fixed-parameter tractable. The main ingredient for the proof is an interesting characterization of up-depth/down-depth in terms of a partition of the vertex set. Using this characterization, we can then show that the UP-DEPTH and DOWN-DEPTH problems can be reduced to the model checking problem of Monadic Second Order Logic, which due to a well-known theorem by Courcelle (1990) is fixed-parameter tractable parameterized by the length of the input formula and the treedepth of the input structure (in our case a directed graph). Together with Theorem 2 this then implies the result.

Theorem 4. DOWN-DEPTH *and* UP-DEPTH *are linear-time fixed-parameter tractable.*

4 Planning on Up-Trees

In this section we present our algorithm for Planning on up-trees, which shows that COST OPTIMAL PLANNING can be solved in polynomial-time for planning instances with bounded domain $|D|$ and bounded up-depth. In the following let $\mathbb{P} = \langle V, D, A, I, G, \Delta\rangle$ be a SAS$^+$ instance, G the (acyclic) causal graph of $\mathbb{P}$, and F an undirected forest with vertex set $V(G)$ such that G is contained in the upward-closure of F. For a plan ω for $\mathbb{P}$ and a variable $v \in V$, we denote by $\delta_\omega(v)$ the number of times the value of v changes during the execution of ω on $\mathbb{P}$. One of the crucial observations for our algorithm is that $\delta_\omega(v)$ can be bounded solely in terms of $w_F(v)$ and $|D|$ as shown in the following lemma. Let $f^\uparrow(x) : \mathbb{N} \rightarrow \mathbb{N}$ be the function defined recursively by setting $f^\uparrow(1) = |D| - 1$ and $f^\uparrow(x+1) = (|D|-1)((\sum_{y=1}^{x} f^\uparrow(y)) + 1)$. A straightforward computation shows that $f^\uparrow(x) \leq (x)!(|D|-1)^x$.

Lemma 5. *Let ω be a minimal plan for $\mathbb{P}$ and $v \in V$. Then $\delta_\omega(v) \leq f^\uparrow(w_F(v)) \leq (w_F(v)!)(|D|-1)^{w_F(v)})$.*

Proof. The proof is via induction on $w_F(v)$. For the base case it holds that v is a root of F and as such does not appear in a precondition of any action effecting a variable other than v. Consequently we obtain that $\delta_\omega(v) \leq |D| - 1$, as required.

Towards showing the induction step, let $v \in V$ and because of the induction hypothesis it holds that $\delta_\omega(w) \leq f^\uparrow(w_F(w))$ for every w with $w_F(w) < w_F(v)$. The main property of up-depth allowing us to bound the number of changes for v in ω is the fact that because G is contained in the upward-closure of F, it holds that the only actions in ω having a precondition on v (other than actions effecting v) are actions that effect one of the ancestors of v in F. Since there are at most $w_F(v) - 1$ ancestors of v in F and the number of actions in ω effecting any ancestor w of v is by the induction hypothesis at most $f^\uparrow(w_F(w))$, we obtain that there are at most $\sum_{d=1}^{w_F(v)-1} f^\uparrow(d)$ actions in ω that have a precondition on v. Since every such precondition requires at most $|D| - 1$ changes to the value of v and at most $|D| - 1$ changes to the value of v are required by ω to get v to its goal state, we obtain $\delta(v) \leq (|D|-1)((\sum_{x=1}^{w_F(v)-1} f^\uparrow(x)) + 1) = f^\uparrow(w_F(v))$. □

Theorem 6. COST OPTIMAL PLANNING *can be solved in time $\mathcal{O}(|a_{\max} w_F|^{\ell(w_F)} \ell(w_F)(w_F - 1 + |D|^2)|V|)$, where $\ell(i) = \sum_{j=1}^{<i} f^\uparrow(j) \leq f^\uparrow(i)$. Hence* COST OPTIMAL PLANNING *can be solved in polynomial-time if $w_F + |D|$ is constant.*

Proof. We will show the theorem by providing a dynamic programming algorithm for COST OPTIMAL PLANNING that proceeds bottom-up on the forest F. The algorithm computes a set $\mathcal{R}(v)$ of records for every vertex $v \in V(F)$, where each record (ω, c) in $\mathcal{R}(v)$ consists of a sequence of actions in $A_{U_F(v)}$ and a number $c \in \mathbb{Q}_{\geq 0}$. The semantics of a record is as follows. $(\omega, c) \in \mathcal{R}(v)$ if and only if:

(R1) for every $v \in U_F(v)$, it holds that $|\omega\langle v\rangle| \leq f^\uparrow(w_F(v))$,

(R2) ω is a plan for $\mathbb{P}[U_F(v)]$, and

(R3) c is the minimum cost of $\omega^+\langle L_F[v]\rangle$, where ω^+ is a cost-optimal plan for $\mathbb{P}[U_F(v) \cup L_F[v]]$ such that $\omega^+\langle U_F(v)\rangle = \omega$.

Note that $|\omega| \leq \ell(w_F(v))$ for any sequence ω satisfying (R1).

Before we show how to construct the set of records for every node of F, let us briefly comment on how we obtain a solution for $\mathbb{P}$ once we have computed the set of records for every node. Since a solution for F can be obtained by putting together solutions for each connected component of F (since the planning instances induced by different components of F are independent), we can assume that F is a tree with only one root r. Because of the semantics of the records, we have that $\mathcal{R}(r)$ contains at most one record and if it contains a record then the record is of the form $(\langle\rangle, c)$, where c is the minimum cost of a plan for $\mathbb{P}$. Moreover if $\mathcal{R}(r) = \emptyset$ then we know that $\mathbb{P}$ does not have a plan.

We now show how to compute the set of all records for every vertex v of F, via a bottom-up dynamic programming algorithm starting at the leaves of F. To achieve this it is sufficient to show how the set of all records can be computed: (A) for a leaf of F and (B) for an internal vertex v of F provided that the sets of all records for all children of v in F have already been computed. Towards (A), let l be a leaf of F. Because of the semantics of a record, we can compute the set $\mathcal{R}(l)$ of records for l by enumerating all sequences ω of actions in $A_{U_F(l)}$ that satisfy (R1), checking for each of them whether they satisfy (R2) and if so computing the number c as defined in (R3). Checking whether ω satisfies (R2) can be achieved in time $\mathcal{O}(|\omega||U_F(l)|)$ by simply executing ω on $\mathbb{P}[U_F(l)]$. The following claim shows that the number c satisfying (R3) can also be computed efficiently.

Claim 1. Let ω be a plan for $\mathbb{P}[U_F(l)]$, then the minimum cost c of $\omega^+\langle l\rangle$, where ω^+ is a cost-optimal plan for $\mathbb{P}[U_F[l]]$ such that $\omega^+\langle U_F(l)\rangle = \omega$ can be computed in time $\mathcal{O}(|\omega||D|^2)$.

Hence the total time required to check whether ω satisfies (R2) and to compute the number c satisfying (R3) is at most $\mathcal{O}(\ell(w_F(l))(w_F - 1 + |D|^2))$. Moreover since there are at most $|a_{\max} w_F(l)|^{\ell(w_F(l))}$ such sequences satisfying (R1), we obtain $\mathcal{O}(|a_{\max} w_F(l)|^{\ell(w_F(l))} \ell(w_F(l))(w_F - 1 + |D|^2))$ as the total running time required to compute $\mathcal{R}(l)$.

Towards (B), let v be an internal node with children $v_1, \dots, v_r$ and assume that the sets of records for $v_1, \dots, v_r$ have already been computed. The following claim, characterizes the existence of a record $(\omega, c) \in \mathcal{R}(v)$.

Claim 2. Let ω be a sequence of actions in $A_{U_F(v)}$ satisfying (R1) and (R2). Then $(\omega, c) \in \mathcal{R}(v)$ if and only if c is the minimum number such that for every i with $1 \leq i \leq r$ there is a record $(\omega_v, c_i) \in \mathcal{R}(v_i)$, where ω_v is any sequence of actions in $A_{U_F[v]}$ (satisfying (R1) and (R2)) such that $\omega = \omega_v\langle U_F(v)\rangle$, and $c = (\sum_{i=1}^{l} c_i) + \Delta(\omega_v\langle v\rangle)$.

With the help of the above claim we now show how to compute the set of records for an internal node v. The computation of the set of records for v is achieved in two steps.

First we compute the set $\mathcal{R}$ that contains every pair (ω, c) such that for every i with $1 \leq i \leq r$ there is a c_i with $(\omega, c_i) \in \mathcal{R}(v_i)$ and moreover $c = \sum_{i=1}^{r} c_i$. The computation of $\mathcal{R}$ can be done by "joining" the sets of records

$\mathcal{R}(v_1), \dots, \mathcal{R}(v_r)$ one by one in time that is linear in the maximum number of records in any $\mathcal{R}(v_i)$, the maximum length of any sequence ω with $(\omega, c) \in \mathcal{R}$, and the number r of children of v, i.e., the computation of $\mathcal{R}$ can be achieved in time $\mathcal{O}(|a_{\max} w_F(v_i)|^{\ell(w_F(v_i))}\ell(w_F(v_i))r)$.

In the second step we aggregate the records in $\mathcal{R}$, which represent sequences of actions ω effecting variables in $U_F(v) \cup \{v\}$, onto sequences of actions that effect only the variables in $U_F(v)$. Note that the correctness of the computation of $\mathcal{R}(v)$ follows immediately from Claim 2. Moreover $\mathcal{R}(v)$ can be computed from $\mathcal{R}$ using a simple run of all $(\omega, c) \in \mathcal{R}$ in time $\mathcal{O}(|a_{\max} w_F(v_i)|^{\ell(w_F(v_i))}\ell(w_F(v_i))r)$. The total running-time of the algorithm is then the sum of the running-times over all nodes of F, which is at most $\mathcal{O}(|a_{\max} w_F|^{\ell(w_F)}\ell(w_F)(w_F - 1 + |D|^2)|V|)$. □

The above algorithm shows that COST OPTIMAL PLANNING is solvable in polynomial-time if both the domain D and the depth w_F of F are bounded by a constant. It is natural to ask whether the algorithm can be improved to a fixed-parameter algorithm parameterized by D and w_F. The following theorem shows that this is unlikely by showing that already PLANNING is W[1]-hard parameterized by w_F even if $|D| = 2$.

Theorem 7. PLANNING *is* W*[1]-hard parameterized by* w_F *even for planning instances with binary domain.*

5 Planning on Down-Trees

In this section we show that COST OPTIMAL PLANNING is fixed-parameter tractable parameterized by domain size and down-depth. In the following let $\mathbb{P} = \langle V, D, A, I, G, \Delta \rangle$ be a SAS$^+$ instance, G the (acyclic) causal graph of $\mathbb{P}$, and F be an undirected forest with vertex set $V(G)$ such that G is contained in the downward-closure of F. The crucial observation behind our algorithm is the following lemma, which shows that we can assume that the length of the sequence of states of the ancestors of a variable v in F that are required by actions effecting a variable v or its descendants in a plan for $\mathbb{P}$ can be bounded in terms of only $|D|$ and $h_F(v)$. This allows us later to remember only a limited amount of information about all sub-plans for the descendants of v that are contained in a plan for $\mathbb{P}$; this is because if the sequence of such states has bounded length then so does the sequence of required preconditions of the actions effecting v or descendants of v. Let $f^{\downarrow}(x) : \mathbb{N} \to \mathbb{N}$ be the function defined recursively by setting $f^{\downarrow}(1) = |D| - 1$ and $f^{\downarrow}(x) = (\mathsf{TLCSS}(f^{\downarrow}(x-1), |D|^{w_F - x}) + 1)|D|$.

Lemma 8. *Let* ω *be a plan for* $\mathbb{P}$*. Then there is a plan* ω' *for* $\mathbb{P}$ *whose cost is at most the cost of* ω *such that for every* $v \in V$ *it holds that:*

(P1) $|\mathsf{cp}(\mathsf{ST}(\omega', L_F[v], U_F[v]))| \leq f^{\downarrow}(h_F(v))$ *and*

(P2) $\mathsf{cp}(\mathsf{ST}(\omega', L_F[v], U_F[v]))$ *is a sub-sequence of* $\mathsf{cp}(\mathsf{ST}(\omega, L_F[v], U_F[v]))$.

Proof Sketch. We will transform ω into ω' using a bottom-up procedure starting at the leaves of F. Namely, we will start by showing how the transformation procedure is applied to the leaves of F and then under the assumption that the transformation has already been applied for all children of an internal node v of F, we will apply the transformation to v.

Hence let ω be the given plan and v be a leaf of F. Since v does not appear in a precondition of an action that does effect a variable other than v, it follows that there is no reason to set v to a value that it had before. Hence if $|\omega\langle v \rangle| > |D| - 1$ then $\omega\langle v \rangle$ contains unnecessary actions that can be removed from ω to obtain ω'. But then $|\omega'\langle v \rangle| \leq |D| - 1$, which implies in particular that $|\mathsf{cp}(\mathsf{ST}(\omega', L_F[v], U_F[v]))| \leq |\mathsf{ST}(\omega', L_F[v], U_F[v])| = |\omega'\langle v \rangle| \leq |D| - 1 = f^{\downarrow}(1)$. Hence ω' satisfies (P1) for the node v and it also satisfies (P2) since we only removed actions from ω.

Towards showing the transformation of ω into ω' for an internal node of F, let v be an internal node of F with children $v_1, \dots, v_r$ and let ω' be the plan obtained from ω after we have already applied the transformation for all descendants of v in F. Then $|\mathsf{cp}(\mathsf{ST}(\omega', L_F[u], U_F[u]))| \leq f^{\downarrow}(h_F(u))$ for every $u \in L_F(v)$. We now show how to transform ω' such that it satisfies (P1) and (P2) also for the node v.

Let X be the set consisting of all total states of the variables in $U_F[v]$. Then $s := \mathsf{ST}(\omega, L_F(v), U_F[v])$ and $s_i := \mathsf{cp}(\mathsf{ST}(\omega', L_F[v_i], U_F(v_i)))$ for every i with $1 \leq i \leq r$ are sequences over X. Let S be the set $\{s_1, \dots, s_r\}$. Then because ω' already satisfies (P1) for all children of v, we have that $|s_i| \leq f^{\downarrow}(h_F(v) - 1)$. Also s satisfies (S1) for the set S of sequences (by definition). Let s' be a sub-sequence of s satisfying (S1) and (S2) for S. Then because of Proposition 1, s' has length at most $\mathsf{TLCSS}(f^{\downarrow}(h_F(v) - 1), |X|)$. We show now that by reordering the actions effecting variables in $L_F(v)$, we can transform ω' into a plan ω'' such that $\mathsf{cp}(\mathsf{ST}(\omega'', L_F(v), U_F[v])) = s'$. Informally, we obtain ω'' from ω' by aligning the action sequences $\omega\langle L_F[v_i] \rangle$ with the first occurrence of s_i in s'. By doing so we ensure that $\omega''\langle L_F[v_i] \rangle = \omega'\langle L_F[v_i] \rangle$ and all preconditions of the actions in $\omega''\langle L_F[v_i] \rangle$ on $U_F[v]$ are still satisfied, which implies that ω'' is still a plan for $\mathbb{P}$. Then $\mathsf{cp}(\mathsf{ST}(\omega'', L_F(v), U_F[v])) = s'$ and hence ω'' satisfies (P2) and moreover $|\mathsf{cp}(\mathsf{ST}(\omega'', L_F(v), U_F[v]))| = |s'| \leq \mathsf{TLCSS}(f^{\downarrow}(h_F(v) - 1), |X|)$. It hence only remains to deal with the actions in A_v, which can be done by showing that at most $(|D|-1)(|s'|+1)$ of those need to appear in ω, which implies $|\mathsf{cp}(\mathsf{ST}(\omega, L_F[v], U_F[v]))| \leq |s'| + (|s'|+1)(|D|-1) \leq (|s'| + 1)|D| \leq f^{\downarrow}(h_F(v))$. □

Since the actions effecting a variable v can only have preconditions on the ancestors of v in F, it holds that $a_{\max}$ can be bounded only in terms of $|D|$ and d_F, which implies that our algorithm for COST OPTIMAL PLANNING presented below runs in linear-time w.r.t. the number of variables of $\mathbb{P}$.

Theorem 9. COST OPTIMAL PLANNING *can be solved in time* $\mathcal{O}((|D|^{w_F} + 1)^{2f^{\downarrow}(h_F)} f^{\downarrow}(w_F)|V|a_{\max} w_F)$ *and is hence linear-time fixed-parameter tractable parameterized by* w_F *and* $|D|$.

Proof. Similar to the algorithm given in Theorem 6 we will provide a dynamic programming algorithm for COST OPTIMAL PLANNING that proceeds bottom-up on the forest F. That is we will compute a set $\mathcal{R}(v)$ of records for each node of F, where a record is a pair (ρ, c) consisting of a compact sequence ρ of total states on the variables in $U_F[v]$ of length at most $f^{\downarrow}(h_F(v))$ and $c \in \mathbb{Q}_{\geq 0}$. For a variable $v \in V$ and a sequence ρ of total states over some set of variables containing v, we denote by $\mathsf{cp}(\rho, v)$ the sequence obtained from ρ after removing every element whose position is in $\{ i \mid \rho[i]\langle v\rangle = \rho[i+1]\langle v\rangle \}$, i.e., we compress ρ w.r.t. changes of v.

For a vertex $v \in V(F)$ and a compact sequence ρ of total states on the variables in $U_F[v]$, we say that a sequence ω of actions in $A_{L_F[v]}$ is *compatible* with ρ if:

(C1) ω is a plan for $\mathbb{P}[L_F[v]]$.

(C2) For every i with $1 \leq i \leq |\omega_v|$, where $\omega_v := \omega\langle v\rangle$, we have that $\mathsf{pre}(\omega_v[i])\langle U_F[v]\rangle \sim \mathsf{cp}(\rho, v)[i]$ and $\mathsf{eff}(\omega_v[i])\langle v\rangle = \mathsf{cp}(\rho, v)[i+1]\langle v\rangle$. Moreover, $|\omega_v| = |\mathsf{cp}(\rho, v)| - 1$.

(C3) There is an extension ρ' of some sub-sequence of ρ such that $\rho' \sim \mathsf{PR}(\omega_R, U_F[v])$, where $\omega_R := \omega\langle L_F(v)\rangle$.

Intuitively (C2) ensures that the sequences of changes to variable v resulting from ω is modeled exactly by $\mathsf{proj}(\rho, v)$ and moreover the preconditions of the actions in ω used to achieve these changes are compatible with the sequence of states of the variables in $U_F[v]$ given by ρ. (C3) ensures that the sequence of preconditions on the variables in $U_F[v]$ that are required by the actions in $\omega_R := \omega\langle L_F(v)\rangle$ are contained in ρ.

The semantics of a record is as follows. $(\rho, c) \in \mathcal{R}(v)$ if and only if ρ is compact, $|\rho| \leq f^{\downarrow}(h_F(v))$ and c is the minimum cost of any sequence ω of actions (from $A_{L_F[v]}$) that is compatible with ρ; if no such plan exists then $(\rho, c) \notin \mathcal{R}(v)$. Intuitively, the set $\mathcal{R}(v)$ of records contains the information about all cost-optimal plans ω for $\mathbb{P}[L_F[v]]$ whose sequence of required preconditions on the variables in $U_F[v]$, i.e., the sequence $\mathsf{PR}(\omega, U_F[v])$, can be satisfied by a given sequence ρ of total states on the variables in $U_F[v]$ of length at most $f^{\downarrow}(h_F(v))$. It then follows from Lemma 8 that it is sufficient to only consider sequences ρ of this length since it shows the existence of a cost-optimal plan having Property (P1).

It is now easy too see how to obtain a solution from the sets of records (see also the proof of Theorem 6). We will hence skip this part and come directly to the computation of the records for (A) a leaf node and (B) an internal node of F.

Towards (A), let l be a leaf of F. Then the following claim, which is a special case of Claim 5 below, characterizes the existence of a record (ρ, c) in $\mathcal{R}(l)$.

Claim 3. Let ρ be a compact sequence of total states on the variables in $U_F[l]$ of length at most $f^{\downarrow}(1) = |D| - 1$. Then $(\rho, c) \in \mathcal{R}(v)$ if and only if c is the cost of a cost-optimal plan ω_v for $\mathbb{P}[v]$ that satisfies (C2) w.r.t. ρ.

See Claim 5 for a proof of the above claim. Because of the above claim, we can compute the set $\mathcal{R}(l)$ of records for l by enumerating all compact sequences ρ of total states of the variables in $U_F[v]$ of length at most $|D| - 1$ and for each such sequence ρ we compute the cost of a cost-optimal plan ω_v for $\mathbb{P}[v]$ that satisfies (C2) w.r.t. ρ. The following claim shows that this can be achieved in time $\mathcal{O}((|D| - 1)a_{\max}w_F)$ (the claim even shows a more general statement that we will use for the computation of the records for inner nodes later on).

Claim 4. Let v be a node of F and ρ be a (compact) sequence of total states on the variables in $U_F[v]$. Then a cost-optimal plan for $\mathbb{P}[v]$ that satisfies (C2) w.r.t. ρ can be computed in time $\mathcal{O}(|\rho| a_{\max} w_F(v))$.

Since there are at most $(|D|^{w_F} + 1)^{|D|-1}$ compact sequences of total states of the variables in $U_F[v]$ of length at most $|D| - 1$ and using the above claim we need time at most $\mathcal{O}((|D| - 1)a_{\max}w_F)$ for every such sequence, we obtain $\mathcal{O}((|D|^{w_F} + 1)^{|D|-1}(|D| - 1)a_{\max}w_F)$ as the total time required to compute $\mathcal{R}(l)$.

Towards (B), let v be an internal node with children $v_1, \ldots, v_r$. The following claim, which is a generalization of Claim 3, characterizes the existence of a record $(\rho, c) \in \mathcal{R}(v)$.

Claim 5. Let ρ be a compact sequence of total states on the variables in $U_F[v]$ of length at most $f^{\downarrow}(h_F(v))$. Then $(\rho, c) \in \mathcal{R}(v)$ if and only if c is the minimum number satisfying $c \geq c_v + \sum_{i=1}^{r} c_i$, where:

(1) c_v is the minimum cost of a plan ω_v for $\mathbb{P}[v]$ that satisfies (C2) w.r.t. ρ and

(2) for every i with $1 \leq i \leq r$, c_i is the minimum number such that there is a record $(\rho_i, c_i) \in \mathcal{R}(v_i)$ such that $\mathsf{cp}(\mathsf{proj}(\rho_i, U_F[v]))$ is a sub-sequence of ρ.

Because of the above claim, we can compute the set $\mathcal{R}(v)$ of records by enumerating all compact sequences ρ of total states of the variables in $U_F[v]$ of length at most $f^{\downarrow}(h_F(v))$ and for each such sequence ρ computing c_v defined by (1) and $c_1, \ldots, c_r$ defined by (2). Hence let ρ be such a compact sequence of total states for the variables in $U_F[v]$ of length at most $f^{\downarrow}(h_F(v))$. Because of Claim 4, c_v can be computed in time $\mathcal{O}(f^{\downarrow}(h_F(v))a_{\max}w_F(v))$. Moreover, we can compute c_i by finding the minimum c' such that there is a record $(\rho', c') \in \mathcal{R}(v_i)$ and $\mathsf{cp}(\mathsf{proj}(\rho', U_F[v]))$ is a sub-sequence of ρ. A straightforward analysis then shows that the total running of the algorithm is at most $\mathcal{O}((|D|^{w_F} + 1)^{2f^{\downarrow}(h_F)} f^{\downarrow}(w_F)|V|a_{\max}w_F)$. □

6 Conclusion

We view this paper as an initial step towards a better understanding of acyclic planning based on exploiting structural parameters on the causal graph. There are a few algorithms based on bounding the treewidth reported in the literature [Brafman and Domshlak, 2006; Bäckström, 2014; Domshlak and Nazarenko, 2013]. The most relevant in our context is the algorithm by Brafman and Domshlak. A shortcoming with their algorithm is that the number of variable changes must be somehow estimated and this is typically a non-trivial task. In the case of up-trees and down-trees, we do have bounds on the number of variable changes (Lemma 5 and Lemma 8). A natural step would thus be to obtain bounds for the number of variable changes for causal graphs with bounded treewidth. How to do this is, unfortunately, highly unclear. Another way forward may be to generalize our results to other structural parameters such as treedepth. This is probably easier than trying to approach treewidth directly.

References

[Aghighi *et al.*, 2015] Meysam Aghighi, Peter Jonsson, and Simon Ståhlberg. Tractable cost-optimal planning over restricted polytree causal graphs. In *Proc. Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-2015)*, pages 3225–3231, 2015.

[Bacchus and Yang, 1994] Fahiem Bacchus and Qiang Yang. Downward refinement and the efficiency of hierarchical problem solving. *Artif. Intell.*, 71(1):43–100, 1994.

[Bäckström, 2014] Christer Bäckström. Parameterising the complexity of planning by the number of paths in the domain-transition graphs. In *21st European Conference on Artificial Intelligence (ECAI 2014)*, pages 33–38, 2014.

[Bliem *et al.*, 2016] Bernhard Bliem, Sebastian Ordyniak, and Stefan Woltran. Clique-width and directed width measures for answer-set programming. In *Proc. 22nd European Conference on Artificial Intelligence (ECAI 2016)*, pages 1105–1113, 2016.

[Bliem *et al.*, 2017] Bernhard Bliem, Marius Moldovan, Michael Morak, and Stefan Woltran. The impact of treewidth on ASP grounding and solving. In *Proc. 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 852–858, 2017.

[Bodlaender *et al.*, 1998] Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of graphs. *SIAM J. Discrete Math.*, 11(1):168–181, 1998.

[Brafman and Domshlak, 2003] R. Brafman and C. Domshlak. Structure and complexity in planning with unary operators. *J. Artif. Intell. Res. (JAIR)*, 18:315–349, 2003.

[Brafman and Domshlak, 2006] Ronen I. Brafman and Carmel Domshlak. Factored planning: How, when, and when not. In *Proc. AAAI 2006*, pages 809–814. AAAI Press, 2006.

[Courcelle, 1990] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[Diestel, 2012] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

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

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Ganian *et al.*, 2017] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In *Proc. 34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, pages 36:1–36:17, 2017.

[Gaspers and Szeider, 2013] Serge Gaspers and Stefan Szeider. Strong backdoors to bounded treewidth SAT. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 489–498. IEEE Computer Society, 2013.

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

[Gottlob *et al.*, 2010] Georg Gottlob, Reinhard Pichler, and Fang Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artif. Intell.*, 174(1):105–132, 2010.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.*, 26:191–246, 2006.

[Jonsson and Bäckström, 1998] Peter Jonsson and Christer Bäckström. Tractable plan existence does not imply tractable plan generation. *Annals of Mathematics and Artificial Intelligence*, 22(3–4):281–296, 1998.

[Jonsson *et al.*, 2014] Anders Jonsson, Peter Jonsson, and Tomas Lööw. Limitations of acyclic causal graphs for planning. *Artif. Intell.*, 210:36–55, 2014.

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

[Katz and Keyder, 2012] Michael Katz and Emil Keyder. Structural patterns beyond forks: Extending the complexity boundaries of classical planning. In *Proc. Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-2012)*, 2012.

[Knoblock, 1994] Craig A. Knoblock. Automatically generating abstractions for planning. *Artif. Intell.*, 68(2):243–302, 1994.

[Nesetril and de Mendez, 2012] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[Paulusma *et al.*, 2016] Daniël Paulusma, Friedrich Slivovsky, and Stefan Szeider. Model counting for CNF formulas of bounded modular treewidth. *Algorithmica*, 76(1):168–194, 2016.

[Ståhlberg, 2017] Simon Ståhlberg. *Methods for Detecting Unsolvable Planning Instances using Variable Projection*. PhD thesis, Linköping University, Sweden, 2017.

[Williams and Nayak, 1997] B. C. Williams and P. Pandurang Nayak. A reactive planner for a model-based executive. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 1178–1185, 1997.