Article:
Kronegger, M., Ordyniak, S. and Pfandler, A. (2019) Backdoors to planning. Artificial Intelligence, 269. pp. 49-75. ISSN: 0004-3702

https://doi.org/10.1016/j.artint.2018.10.002

Backdoors to Planning[☆]

Martin Kronegger[a,*], Sebastian Ordyniak[b,*], Andreas Pfandler[b,c,*]

[a] *Johannes Kepler University Linz, Austria*
[b] *TU Wien, Austria*
[c] *University of Siegen, Germany*

Abstract

Backdoors measure the distance to tractable fragments and have become an important tool to find fixed-parameter tractable (fpt) algorithms for hard problems in AI and beyond. Despite their success, backdoors have not been used for planning, a central problem in AI that has a high computational complexity. In this work, we introduce two notions of backdoors building upon the causal graph. We analyze the complexity of finding a small backdoor (detection) and using the backdoor to solve the problem (evaluation) in the light of planning with (un)bounded plan length/domain of the variables. For each setting we present either an fpt-result or rule out the existence thereof by showing parameterized intractability. For several interesting cases we achieve the most desirable outcome: detection and evaluation are fpt. In addition, we explore the power of polynomial preprocessing for all fpt-results, i.e., we investigate whether polynomial kernels exist. We show that for the detection problems, polynomial kernels exist whereas we rule out the existence of polynomial kernels for the evaluation problems.

Keywords: Planning, Backdoors, Causal graph, Fixed-parameter tractable algorithms, (Parameterized) complexity

1. Introduction

Planning is one of the central formalisms in AI. Unfortunately, the expressive power of planning comes at the cost of high computational complexity. In general, already propositional STRIPS planning is PSPACE-complete for unbounded plan length. In order to cope with this high complexity, several fragments of planning have been considered where planning becomes tractable. For propositional STRIPS a comprehensive complexity analysis was performed by Bylander [14]. Later, Bäckström and Nebel [8] presented a similar analysis for the SAS$^+$ formalism, where the state variables range over multi-valued domains.

A more fine-grained understanding of the hardness and tractability of a problem can be obtained from the viewpoint of parameterized complexity theory [21]. Here one is interested in identifying one or multiple features of the instance – the so-called parameters – which capture the combinatorial explosion. More formally, the time needed to solve an instance of the problem is not only measured in terms of the input size n, but also depends on the parameter (or combination of parameters) k. The class of efficiently solvable problems is FPT (*fixed-parameter tractable*), i.e., the problem can be solved by an fpt-algorithm in time $f(k) \cdot n^{\mathcal{O}(1)}$, where $f(k)$ is a computable function depending only on the parameter k but not on n. The exponential time complexity is thus confined to the parameter, i.e., the function $f(k)$. Therefore, an fpt-algorithm can be considered efficient as long as the parameter values of an instance are sufficiently low. Furthermore notice

[☆] The paper is a thoroughly revised and largely extended version of [55] containing novel and constructive algorithms for backdoor detection, kernelization lower bounds for backdoor evaluation, examples for the applicability of our approach to problems in AI and beyond, as well as enhanced versions of our tractability results for planning.

[*] Corresponding authors
URL: martin.kronegger@jku.at (Martin Kronegger), sordyniak@gmail.com (Sebastian Ordyniak), pfandler@dbai.tuwien.ac.at (Andreas Pfandler)

that an fpt-result immediately yields tractability of the problem if parameter k is bounded by a constant. Techniques from parameterized complexity have been successfully used to tackle hard problems, e.g., in Knowledge Representation & Reasoning (for a survey we refer to Gottlob and Szeider [44]). But parameterized complexity is not limited to this field. In the last years these techniques have lead to notable progress in many areas such as QBF [2, 25], ILP [33], SAT and CSP [34, 36], Computational Social Choice [9, 13, 26], and Belief change [64]. For planning, a parameterized complexity analysis was initiated by Downey et al. [23]. In recent works [1, 3, 4, 7, 55–57], several fpt-results have been obtained for SAS^+ and propositional STRIPS. Despite this success the desire for additional fpt-algorithms prevails.

A powerful tool to obtain fpt-algorithms are so-called *backdoors* (for a survey see Gaspers and Szeider [39]). Backdoors were originally introduced by Crama et al. [16] and Williams et al. [69] to explain the behavior of SAT and CSP solvers on practical instances. The basic idea is that the size of a backdoor set measures the distance to a tractable fragment of the problem. In the past, the backdoor approach has been used to obtain fpt-results for SAT [39] and CSP [34, 36–38, 59, 61], QBFs [25, 68], ASP [27], and Argumentation [24]. Furthermore, several experiments regarding the size of the backdoors of SAT and ASP instances were performed [28, 45, 53]. Interestingly, some of the instances were obtained through SAT- and ASP-encodings of planning instances. Also quite recently backdoors have also been used to construct parameterized reductions to SAT for problems harder than NP such as ASP [29] and Abduction [63]. However, backdoors for planning have not been considered yet.

In our approach to backdoors for planning we build upon the so-called causal graph. The causal graph models the dependencies between variables in a planning instance and based on the structure of the causal graph various tractable fragments of planning have been identified [6, 10, 15, 20, 40–42, 48–50, 52]. In this work we consider backdoor sets into the polynomial-time tractable fragment of planning instances whose causal graph consists merely of small components, which has been introduced in [15]. Informally, in this way our backdoor sets capture the distance to planning instances that are the disjoint union of small independent planning instances. In comparison to the classical backdoors for SAT our backdoors for planning exhibit quite a few notable and interesting differences. Whereas solving SAT given a backdoor is easy this task becomes more involved for planning, which, in fact, turned out to be the main challenge for our approach. Intuitively, one reason for this complication is that for planning is about finding a sequence of states rather than a single assignment to the variables as is the case for SAT. Also in contrast to classical backdoors for SAT that are defined by syntactical restrictions of the formula, e.g., Horn formulas, we focus on a tractable fragment defined via structural restrictions of the instance and consider two different distance measures towards this tractable fragment.

As a by-product of our analysis of the parameterized complexity of backdoors to planning we obtain a methodology that is applicable to AI problems far beyond planning. We show how the underlying graph structure can be used to define sensible notions of backdoors that can also handle problems dealing with states (or similar concepts). Furthermore, we show how prominent techniques from kernelization can be combined with the former approach for an additional gain in efficiency. There is hope that this approach can be used in many other AI settings to develop efficient algorithms that scale well with the distance of a given instance to the tractable fragment.

Main contributions

In this work, we introduce two natural notions of backdoors for planning, which are based on the underlying *causal graph* of the planning instance. For both of the new notions of backdoors, we perform a comprehensive parameterized complexity analysis (see Table 1 for an overview). In more detail, we analyze the complexity of the two phases of the backdoor approach: (i) finding a small backdoor (*detection phase*), and (ii) using the additional information given in the backdoor to solve the planning instance (*evaluation phase*). In the evaluation phase, we additionally consider a bound on the plan length and/or on the domain of the variables as additional restrictions. For each of the above settings we present either an fpt-result, or show parameterized intractability. Furthermore, we strengthen all fpt-results by an investigation of the power of polynomial preprocessing, more precisely, of the existence of polynomial kernels. It turns out that polynomial kernels exist for the detection problems, whereas we rule out the existence of polynomial kernels for the evaluation problems (under the usual complexity theoretic assumptions). This gives us a *complete* picture of the parameterized complexity of the backdoor approach with respect to the considered notions of backdoors.

Among our fpt-results is the first fpt-algorithm for planning with unbounded plan length that neither limits the number of variables nor the number of actions in the planning instance.

Another contribution of this work is to present and illustrate the methodology of the backdoor approach in the planning setting, where it is less immediate how backdoors can be used. We believe that this methodology pursued in this work can be generalized to other hard problems in AI and beyond, so that other areas can benefit from the backdoor approach.

2. Preliminaries

We assume the reader to be familiar with the basics of graph theory [18], complexity theory [62], and planning [66]. We use the following notation. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For two sequences s_1 and s_2 of elements from an arbitrary set, we denote by $s_1 s_2$ the concatenation of s_1 and s_2. We will use standard notation from graph theory that can, e.g., be found in [18]. Namely, for an undirected graph $G = (V, E)$ and a subset V' of its vertices, we denote by $G \setminus V'$ the graph obtained from G after removing all vertices (together with all edges incident to a vertex in V') from G. The set of vertices of a graph G is denoted by $V(G)$. Moreover, we denote by $G[V']$ the graph $G \setminus (V \setminus V')$ and by $G \setminus E'$ the graph obtained from G after removing all edges in $E' \subseteq E$.

Parameterized Complexity. Parameterized algorithmics (cf. Downey and Fellows [21, 22], Flum and Grohe [31], Niedermeier [58], Cygan et al. [17]) is a promising approach to obtain efficient algorithms for fragments of intractable problems. In a parameterized complexity analysis the runtime 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. The most favorable 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. We call such an algorithm fixed-parameter tractable (fpt).

Formally, a *parameterized problem* is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is the input alphabet. Problem reductions now also have to take the parameter into account. Let L_1 and L_2 be parameterized problems, with $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$. A *parameterized reduction* (or fpt-reduction) from L_1 to L_2 is a mapping $P : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ such that (i) $(x, k) \in L_1$ iff $P(x, k) \in L_2$; (ii) the mapping can be computed by an fpt-algorithm w.r.t. parameter k; (iii) there is a computable function g such that $k' \leq g(k)$, where $(x', k') = P(x, k)$.

Next, we will define the classes capturing fixed-parameter intractability needed in this work. For further details we refer to the literature on parameterized complexity theory.

The class W[1] contains all problems that are fpt-reducible to Independent Set when parameterized by the size of the solution, i.e., the size of the independent set [22]. The class paraNP [30] is defined as the class of problems that are solvable by a non-deterministic Turing-machine in fpt-time. In our paraNP-hardness proofs, we will make use of the following characterization of paraNP-hardness given by Flum and Grohe [31], Theorem 2.14: any parameterized problem that remains NP-hard when the parameter is set to some constant is paraNP-hard. The following relations between the parameterized complexity classes hold: FPT $\subseteq$ W[1] $\subseteq$ paraNP. Showing W[1]-hardness for a problem rules out the existence of a fixed-parameter algorithm under the usual complexity theoretic assumption FPT $\neq$ W[1].

Closely related to the search for fpt-algorithms is the search for efficient preprocessing techniques. The goal here is to find an equivalent instance (the so-called *kernel*) in polynomial time whose size can be bounded by a function of the parameter. A *kernelization* algorithm transforms in polynomial time a problem instance (x, k) of a parameterized problem L into an instance (x', k') of L such that (i) $(x, k) \in L$ iff $(x', k') \in L$, (ii) $k' \leq f(k)$, and (iii) the size of x' can be bounded above by $g(k)$, for functions f and g depending only on k. It is easy to show that a parameterized problem is in FPT if and only if there is kernelization algorithm. A *polynomial kernel* is a kernel, whose size can be bounded by a polynomial in the parameter.

A *polynomial parameter transformation* (PPT) from a parameterized problem P to a parameterized problem Q is a parameterized reduction from P to Q that maps instances $\langle \mathbb{I}, k \rangle$ of P to instances $\langle \mathbb{I}', k' \rangle$ of Q with the additional property that

1. $\langle \mathbb{I}', k' \rangle$ can be computed in time that is polynomial in $|\mathbb{I}| + k$, and
2. k' is bounded by some polynomial p of k.

Proposition 1 ([7, Proposition 1])**.** *Let P and Q be two parameterized problems such that there is a PPT-reduction from P to Q. Then, if Q has a polynomial kernel also P has a polynomial kernel.*

Planning. Let $V = \{v_1, \ldots, v_n\}$ be a finite set of *variables* over a finite *domain* D. Implicitly define $D^+ = D \cup \{\mathbf{u}\}$, 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* for n variables 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 $s[v]$. A *SAS$^+$ instance* is a tuple $\mathbb{P} = \langle V, D, A, I, G \rangle$ 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* and $G \in (D^+)^n$ is the (partial) *goal state*. 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 let $s \in D^n$. Then a is *valid in* s if for all $v \in V$, either $\mathsf{pre}(a)[v] = s[v]$ or $\mathsf{pre}(a)[v] = \mathbf{u}$. Furthermore, the *result of a in s*, denoted by $\mathrm{res}(s, a)$, is the state $t \in D^n$ defined such that for all $v \in V$, $t[v] = \mathsf{eff}(a)[v]$ if $\mathsf{eff}(a)[v] \neq \mathbf{u}$ and $t[v] = s[v]$ otherwise.

Let $s_0, s_\ell \in D^n$ and let $\omega = \langle a_1, \ldots, a_\ell \rangle$ 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 $s_1, \ldots, s_{\ell-1} \in D^n$ 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 $G[v] = s[v]$ or $G[v] = \mathbf{u}$. An action sequence ω is a *plan for a SAS$^+$ instance* $\mathbb{P}$ if ω is a plan from I to some goal state. We will study the following problems:

SAS$^+$ Planning	
Instance:	A SAS$^+$ instance $\mathbb{P}$.
Question:	Does $\mathbb{P}$ have a plan?

Bounded SAS$^+$ Planning	
Instance:	A SAS$^+$ instance $\mathbb{P}$ and a positive integer k.
Parameter:	k
Question:	Does $\mathbb{P}$ have a plan of length at most k?

Notice that the propositional version of the well-known Strips planning language is a special case of SAS$^+$.

Let $\mathbb{P} = \langle V, D, A, I, G \rangle$ be an SAS$^+$ instance, $V' \subseteq V$, and $A' \subseteq A$. We sometimes use $V(\mathbb{P})$, $A(\mathbb{P})$, $D(\mathbb{P})$, $I(\mathbb{P})$, $G(\mathbb{P})$ to refer to V, A, D, I, and G, respectively. We denote by $\mathbb{P}[V']$ the SAS$^+$ instance $\langle V', D_r, A_r, I_r, G_r \rangle$, where D_r is the restriction of D to the domains of the variables in V', A_r are the actions in A whose preconditions and effects are restricted to the variables in V', and I_r and G_r are the restriction of I and G to the variables in V'. We write $\mathbb{P} \setminus V'$ for the instance $\mathbb{P}[V \setminus V']$. Similarly, we denote by $\mathbb{P}[A']$ the SAS$^+$ instance $\langle V, D, A', I, G \rangle$ and by $\mathbb{P} \setminus A'$ the SAS$^+$ instance $\mathbb{P}[A \setminus A']$. We denote by $\mathbb{P}(V')$ the SAS$^+$ instance obtained from $\mathbb{P}[V']$ after deleting all actions that have at least one precondition or effect on $V(\mathbb{P}) \setminus V'$. If ω is a sequence of actions in A and $A' \subseteq A$, we denote by $\omega[A']$, the sequence obtained from ω after removing all occurrences of actions in $A \setminus A'$.

Finally, we consider the following simple variant of cost-optimal planning. A SAS^+_C instance $\mathbb{P}$ is a tuple $\langle V, D, A, \gamma, I, G \rangle$, where $\langle V, D, A, I, G \rangle$ is a SAS$^+$ instance and $\gamma : A \to \mathbb{N}$ is a *cost-function* assigning a cost in terms of a natural number to every action in A. All notions and definitions for SAS$^+$ planning also carry over to SAS^+_C planning in the natural way. Additionally, we define the *cost* of a sequence $\omega = \langle a_1, \ldots, a_l \rangle$ of actions, denoted by $\gamma(\omega)$, as the sum of the costs of all action occurrences in ω, i.e., $\gamma(\omega) = \sum_{i=1}^{l} \gamma(a_i)$.

3. Using the Causal Graph for Backdoors

In this work we will introduce two new types of backdoors. Before we start with the presentation of the details, we give a high-level introduction to the backdoor approach.

Setting	Domain	Detection (un)bounded plan len.	Evaluation bounded plan length	Evaluation unbounded plan length
variable-deletion	(un)bounded	in FPT (Thm. 5) and pk (Thm. 6)	W[1]-hard (Thm. 7)	W[1]-hard (Thm. 7)
action-deletion	bounded	in FPT (Thm. 9) and pk (Thm. 10)	in FPT (Cor. 14) and npk (Thm. 17)	in FPT (Cor. 12) and npk (Thm. 17)
	unbounded			paraNP-hard (Thm. 15)

Table 1: Complexity map of backdoors to planning. We use *pk* to denote that the problem admits a polynomial kernel and *npk* to denote that the problem does not admit a polynomial kernel (unless coNP $\subseteq$ NP/poly).

The backdoor approach can be separated into two phases. In the first phase (*detection*) one searches for a set, i.e., the backdoor, whose size measures the distance of the given instance to a tractable base class. In the second phase (*evaluation*) one makes use of the information of the backdoor to solve the problem. Usually, the size of the backdoor is considered as parameter in the detection and evaluation problem. Hence, if both problems are fpt, these problems and in consequence also the planning problem can be solved efficiently as long as the backdoor is of moderate size.

For instance, for the SAT problem, one searches for a small set of variables of size k such that the given formula can be reduced to 2^k formulas that belong to the desired tractable base class (e.g., Horn or Krom). For planning, however, it is not immediately clear how to break a hard instance into multiple easy instances by using a backdoor set since the states have to be taken into account.

The basic idea of this work is to use the underlying structure of the planning instance, namely the *causal graph*, instead of the planning instance itself to define the backdoor. Therefore, we first need to recapitulate the concept of the causal graph [6, 11, 15, 54].

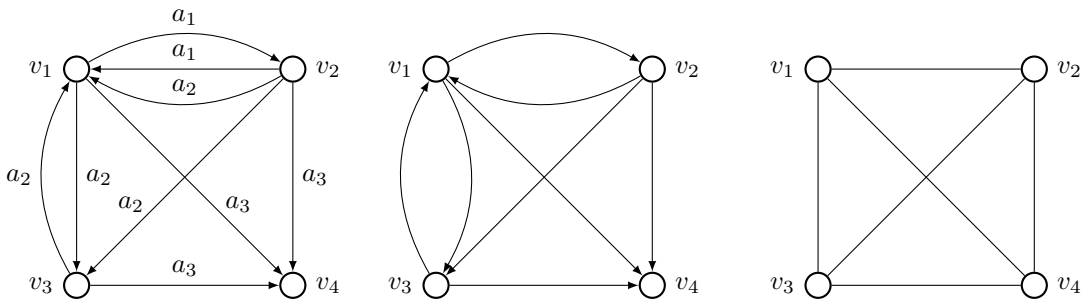
The *labeled causal graph*, denoted by $\mathcal{G}_{\text{L-CAUSAL}}(\mathbb{P})$ of a SAS$^+$ instance $\mathbb{P} = \langle V, D, A, I, G\rangle$ is the edge-labeled directed graph with vertex set V and edge-labeling function $\lambda : E(\mathcal{G}_{\text{L-CAUSAL}}(\mathbb{P})) \to A$ that for every action $a \in A$ and every pair of distinct variables v and v' such that either v appears in the precondition of a and v' appears in the effect of a or v and v' appear together in the effect of a has an arc from v to v' with label a. Moreover, we also define the *causal graph* of an SAS$^+$ instance $\mathbb{P}$, denoted by $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})$, as the directed graph obtained from the labeled causal graph after omitting the labels on the edges and deleting multiple arcs between the same pair of vertices. If C is a set of vertices of a subgraph of $\mathcal{G}_{\text{L-CAUSAL}}(\mathbb{P})$ (or $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})$) we denote by $\mathbb{P}[C]$ and $\mathbb{P}(C)$ the SAS$^+$ instance $\mathbb{P}[V']$ and $\mathbb{P}(V')$, respectively, where $V' \subseteq V$ are all variables that correspond to vertices of C. For a directed graph $D = (V, A)$ we denote by $\overline{D}$ its underlying undirected graph, i.e., the graph with vertex set V and edge set $\{\,\{u, v\} \mid (u, v) \in A\,\}$. We denote by $\mathsf{cc\text{-}size}(H)$ the size of the largest connected component of H if H is an undirected graph and if H is a directed graph then $\mathsf{cc\text{-}size}(H)$ denotes the size of its largest weakly connected component. Note that for a directed graph H it holds that $\mathsf{cc\text{-}size}(H) = \mathsf{cc\text{-}size}(\overline{H})$. If it is clear from the context, we will write "components" instead of "weakly connected components of the causal graph $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})$".

Consider the following example. Let $\mathbb{P}_{\text{example}} = \langle V, D, A, I, G\rangle$ be a planning instance which is given by the variables $V = \{v_1, v_2, v_3, v_4\}$, the domain $D = \{0, 1\}$, the actions $A = \{a_1, a_2, a_3\}$ where $\mathsf{pre}(a_1)[v_1] = 0, \mathsf{eff}(a_1)[v_1] = 1, \mathsf{eff}(a_1)[v_2] = 1, \mathsf{pre}(a_2)[v_2] = 1, \mathsf{eff}(a_2)[v_1] = 0, \mathsf{eff}(a_2)[v_3] = 1, \mathsf{pre}(a_3)[v_1] = \mathsf{pre}(a_3)[v_2] = \mathsf{pre}(a_3)[v_3] = 1, \mathsf{eff}(a_3)[v_4] = 1$, the initial state $I = 0^4$, and the goal G where $G[v_4] = 1$, and $G[v_1] = G[v_2] = G[v_3] = \mathbf{u}$. The different versions of the causal graph as defined above are shown in Figure 1 for the planning instance $\mathbb{P}_{\text{example}}$.

In order to be able to define the backdoors, we need to find a suitable, tractable base class. From the literature it is known that SAS$^+$ PLANNING is solvable in polynomial time if the maximum component size of the underlying causal graph can be bounded by a constant c, i.e., $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})) \leq c$.

Proposition 2 (Chen and Giménez [15])**.** *Let c be a constant. Then* SAS$^+$ PLANNING *can be solved in polynomial time for instances where $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})) \leq c$.*

For the setting with bounded plan length it is easy to obtain a similar result. It suffices to construct the state transition graph (of size $\mathcal{O}(|D|^c)$) for each component and compute the shortest path to the partial



(a) $\mathcal{G}_{\text{L-CAUSAL}}(\mathbb{P}_{\text{example}})$ (b) $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}_{\text{example}})$ (c) $\overline{\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}_{\text{example}})}$

Figure 1: Three different versions of the causal graph of the planning instance $\mathbb{P}_{\text{example}}$ as defined Section 3.

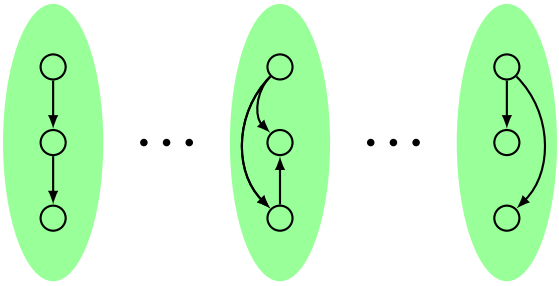


Figure 2: A planning instance $\mathbb{P}$ with $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})) \leq 3$.

goal. If these shortest paths exist, an arbitrary combination of the shortest plans for each component gives a solution.

Proposition 3. *Let c be a constant. Then c-*BOUNDED SAS$^+$ PLANNING *can be solved in polynomial time for instances where* $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P})) \leq c$.

For both settings, notice that each planning instance where $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}))$ is bounded by a constant models (multiple) independent planning instances of constant size. An example of such a situation is depicted in Figure 2. This property, however, is very fragile: a single variable or a single action can be responsible for the causal graph being connected. In this work, we will deal with such situations by allowing a set of variables or actions to be responsible for the causal graph being connected, which yields more robust algorithms.

The backdoors introduced in this work measure the distance of the planning instance $\mathbb{P}$ to a tractable planning instance $\mathbb{P}'$ that has bounded component size. We will consider two natural ways to decompose the components of the causal graph. The backdoor set S either contains the *variables* or the *actions* that have to be removed from the instance to reduce the size of the weakly connected components. Notice that from the viewpoint of classical complexity theory the evaluation problem remains as hard as the original planning problem, because one is free to add all variables or all actions to the backdoor set.

In this work, each backdoor type is considered in the light of four different settings of planning. We consider the SAS$^+$ PLANNING and the BOUNDED SAS$^+$ PLANNING problem in case of an bounded or unbounded domain of the variables. For each case we will present either an fpt-algorithm or show hardness for the classes $\mathsf{W}[1]$ or paraNP. The results of this work are summarized in Table 1.

3.1. Variable-Deletion-Backdoors

In this section we consider planning instances that have a small number of variables whose removal results in a causal graph with components of bounded size. We show that even though the detection of these planning instances is fixed-parameter tractable (Theorem 5) this does not hold for the evaluation problem even for planning instances with bounded domain (Theorem 7).

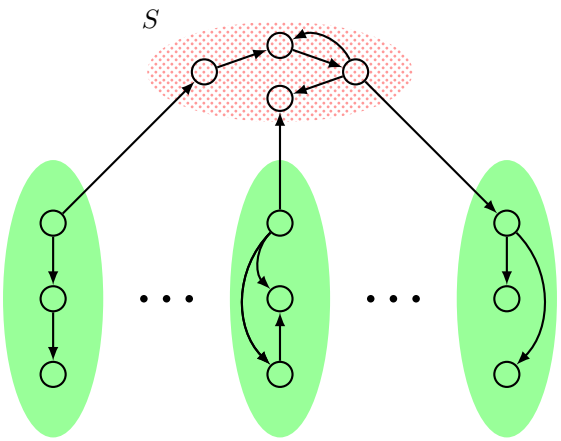


Figure 3: The causal graph of a planning instance where the variables in S need to be removed such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}\setminus S)) \leq 3$.

We start by defining the detection problem for the setting where variables are allowed to be removed (*variable-deletion-backdoors*).

c-CAUSAL DETECTION[VARIABLES]	
Instance:	A SAS$^+$ instance $\mathbb{P}$ and a positive integer k.
Parameter:	k
Question:	Is there a set S of at most k variables of $\mathbb{P}$ such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq c$?

Intuitively, a variable-deletion-backdoor captures the distance (in terms of variables that need to be removed) to an instance where all variables can be partitioned into small sets that are relevant for independent subtasks. Observe that the property of having small components in the causal graph is very fragile: Even if the variables form small independent components, adding a single variable that occurs in all actions in the precondition creates a single, big component in the causal graph. Variable-deletion-backdoors can be seen as a more robust notion. In the example above a backdoor of size one (containing the newly introduced variable) is sufficient. Based on Figure 2, Figure 3 shows the causal graph of a planning instance where the variables in the set S need to be removed such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq 3$.

Next, we build upon this backdoor and extend the problems SAS$^+$ PLANNING and BOUNDED SAS$^+$ PLANNING to make use of a previously computed backdoor S.

c-CAUSAL EVALUATION[VARIABLES]	
Instance:	A SAS$^+$ instance $\mathbb{P}$ and a set S of variables of $\mathbb{P}$ such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}\setminus S)) \leq c$.
Parameter:	$\lvert S\rvert$
Question:	Does $\mathbb{P}$ have a plan?

c-BOUNDED CAUSAL EVALUATION[VARIABLES]	
Instance:	A SAS$^+$ instance $\mathbb{P}$, a set S of variables of $\mathbb{P}$ such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq c$ and an integer k.
Parameter:	$\lvert S\rvert + k$
Question:	Does $\mathbb{P}$ have a plan of length at most k?

We start the complexity analysis by a discussion of the complexity of the detection problem. Observe that at its core the detection problem is a problem on the causal graph of the planning instance. Namely, the detection problem can be equivalently stated as the following graph problem.

c-Vertex Deletion to Small Components	
Instance:	An undirected graph G and an integer k.
Parameter:	k
Question:	Is there a set $R \subseteq V(G)$ of at most k vertices such that $\mathsf{cc\text{-}size}(G \setminus R) \leq c$?

To see that both problems are equivalent consider the following polynomial-time reductions in both directions. Given an instance $I = \langle \mathbb{P}, k \rangle$ of c-Causal Detection[Variables], then $\langle \mathcal{G}_{\text{Causal}}(\mathbb{P}), k \rangle$ is an instance of c-Vertex Deletion to Small Components that is equivalent to I. Moreover, if $I = \langle G, k \rangle$ is an instance of c-Vertex Deletion to Small Components, then $\langle \mathbb{P}, k \rangle$, where $\mathbb{P}$ is any planning instance such that $\overline{\mathcal{G}_{\text{Causal}}(\mathbb{P})}$ is isomorphic with G is equivalent to I. Note that such an instance $\mathbb{P}$ can for instance be obtained as follows:

- for every vertex $v \in V(G)$, $\mathbb{P}$ has one variable v with arbitrary domain,
- for every edge $\{u, v\} \in E(G)$, $\mathbb{P}$ has one action that has an arbitrary effect on u and v, and
- the domain, the initial state as well as the goal state of $\mathbb{P}$ are defined arbitrarily.

To justify a parameterized complexity analysis, we first show NP-hardness using a reduction from the following well-known NP-hard problem [35].

Vertex Cover	
Instance:	An undirected graph G and an integer k.
Parameter:	k
Question:	Does G have a *vertex cover* of size at most k, i.e., is there a set S of at most k vertices of G such that for all edges $\{x, y\} \in E(G)$ either $x \in S$ or $y \in S$?

Theorem 4. *For every $c \geq 1$, c-Causal Detection[Variables] is NP-complete even for planning instances with bounded domain.*

Proof. Due to the equivalence of c-Causal Detection[Variables] and c-Vertex Deletion to Small Components discussed above it is sufficient to show that c-Vertex Deletion to Small Components is NP-complete. Because any solution to c-Vertex Deletion to Small Components (which is clearly of polynomial size) can be verified in polynomial-time, it holds that c-Vertex Deletion to Small Components is in NP. Towards showing NP-hardness of c-Vertex Deletion to Small Components we give a polynomial-time reduction from the Vertex Cover problem. Given an instance $\langle G, k \rangle$ of Vertex Cover, we construct the instance $\langle G', k' \rangle$ of c-Vertex Deletion to Small Components as follows. The graph G' is obtained from G by replacing every vertex $v \in V(G)$ with a complete graph, in the following denoted by C_v, on c new vertices and making every vertex in C_v adjacent to every vertex in C_u for every $\{u, v\} \in E(G)$. Moreover, we set k' to be equal to ck. Note that the reduction can be computed in polynomial-time and it remains to show the equivalence between the two instances.

Towards showing the forward direction, assume that $S \subseteq V(G)$ is a vertex cover of G of size at most k. We claim that $R = \bigcup_{v \in S} V(C_v)$ is a solution for $\langle G', k' \rangle$. Clearly, R has size at most $c|S| \leq ck \leq k'$ as required. Moreover, because S is a vertex cover of G, it follows that every component of $G' \setminus R$ contains vertices of at most one complete graph C_v for any vertex $v \in V(G)$. Hence every component of $G' \setminus R$ has size at most c, as required.

Towards showing the reverse direction, assume that $R \subseteq V(G')$ is a solution for $\langle G', k' \rangle$. We first show that w.l.o.g. we can assume that the following property holds:

(P1) For every vertex $v \in V(G)$ it holds that either $V(C_v) \cap R = \emptyset$ or $V(C_v) \subseteq R$.

Suppose that this is not the case, i.e., there is a vertex $v \in V(G)$ with $V(C_v) \cap R \neq \emptyset$ but not $V(C_v) \subseteq R$. Because C_v forms a complete subgraph of G', it follows that all vertices in $C_v \setminus R$ are contained in the same component, say C, of $G' \setminus R$. If $C = C_v \setminus R$, then we can simply remove all vertices in $V(C_v) \cap R$ from R

and thereby obtain a new solution R' such that either $V(C_v) \cap R' = \emptyset$ or $V(C_v) \subseteq R'$, which brings us one step closer towards satisfying Property (P1). Hence assume that is not the case, i.e., there are two vertices $v, u \in V(G)$ such that C has a non-empty intersection with $V(C_v)$ and $V(C_u)$. Because R is a solution we have that $|(V(C_v) \cup V(C_u)) \setminus R| \leq |V(C)| \leq c$ and consequently $|(V(C_v) \cup V(C_u)) \cap R| \geq c$. It follows that we can obtain a new solution R' from R by replacing $|V(C_v \setminus R)|$ vertices in $R \cap V(C_u)$ with the vertices in $C_v \setminus R$. But then again either $V(C_v) \cap R' = \emptyset$ or $V(C_v) \subseteq R'$ holds and by iterating this procedure until it is no longer applicable, we eventually obtain a solution R'' satisfying Property (P1).

Now let R be a solution satisfying Property (P1). We claim that the set S containing all vertices v of G such that $V(C_v) \cap R \neq \emptyset$ is a vertex cover of G of size at most k. Because of Property (P1), it holds that $|S| = |R|/c \leq k'/c = k$ as required. Moreover, because R is a solution every component of $G' \setminus R$ contains at most c vertices, which due to Property (P1) implies that any such component is equal to C_v for some $v \in V(G)$, which in turn implies that S is a vertex cover of G. □

In the next theorem we will show that the detection problem is fixed-parameter tractable.

Theorem 5. *For any $c \geq 1$, c-Causal Detection[Variables] can be solved in time $\mathcal{O}((c+1)^k|E(G)|)$ and is hence fixed-parameter tractable.*

Proof. Because of the equivalence of c-Causal Detection[Variables] and c-Vertex Deletion to Small Components it is sufficient to show the result for c-Vertex Deletion to Small Components. We will show the theorem by providing a depth-bounded search tree algorithm, which given an instance $I = \langle G, k \rangle$ of c-Vertex Deletion to Small Components either determines that the instance is a No-instance, or outputs a solution $R \subseteq V(G)$ of minimum size for I. The algorithm is based on the following two observations:

- O1 If G is not connected then a (minimum) solution for I can be obtained as the disjoint union of (minimum) solutions for every component of G.
- O2 If G is connected and C is any set of $c+1$ vertices of G such that $G[C]$ is connected, then any solution for I has to contain at least one vertex from C.

Using the above observations the algorithm first checks whether G is connected. If G is not connected the algorithm calls itself recursively on the instance (C, k) for each component C of G. If one of these recursive calls returns No or if the size of the union of the (minimum) solutions returned for each component exceeds k, the algorithm returns that I is a No-instance. Note that for this step of the algorithm it is crucial that the algorithm returns a minimum solution (instead of returning an arbitrary solution of size at most k) for any Yes-instance (G, k). Otherwise the algorithm returns the union of the (minimum) solutions returned for each component of G.

If G is connected and $|V(G)| \leq c$, the algorithm returns the empty set as a solution. Otherwise, i.e., if G is connected but $|V(G)| > c$, the algorithm first computes a set C of $c+1$ vertices of G such that $G[C]$ is connected. This can for instance be done by a Depth-First Search that starts at any vertex of G and stops as soon as $c+1$ vertices have been visited. The algorithm then branches on the vertices in C, i.e., for every $v \in C$ the algorithm recursively computes a solution for the instance $(G \setminus \{v\}, k-1)$. It then returns the solution of minimum size returned by any of those recursive calls, or No if none of those calls returns a solution. This completes the description of the algorithm. The correctness of the algorithm follows immediately from the above observations. Moreover the running time of the algorithm is easily seen to be dominated by the maximum time required for the case that G is connected at each step of the algorithm. In this case the running time can be obtained as the product of the number of recursive calls times the time spent on each of those. Because at each recursive call the parameter k is decreased by one and the number of branching choices is at most $c+1$, we obtain that there are at most $(c+1)^k$ recursive calls. Furthermore, the time spend at each recursive call is dominated by the time required to check whether G is connected, which is linear in the number of edges of G. Putting everything together, we obtain $\mathcal{O}((c+1)^k|E(G)|)$ as the total time required by the algorithm, which completes the proof of the theorem. □

In the next theorem, we show that the backdoor detection problem additionally allows for a polynomial kernel, which allows for efficient preprocessing rules to be applied.

Theorem 6. *For any $c \geq 1$, c-*Causal Detection[Variables] *admits a polynomial kernel of size at most $\mathcal{O}((k+c)^2 kc)$.*

Proof. We will show that c-Vertex Deletion to Small Components has a kernel with at most $(k+k(k+c)c)$ vertices and at most $(k+c)(k+k(k+c)c)$ edges. Using the provided reduction from c-Vertex Deletion to Small Components to c-Causal Detection[Variables] this implies that c-Causal Detection[Variables] has a kernel with at most $(k+k(k+c)c)$ variables and at most $(k+c)(k+k(k+c)c)$ actions, which implies the result of the theorem. Let $\langle G, k\rangle$ be an instance of c-Vertex Deletion to Small Components. W.l.o.g. we can assume that G is connected, otherwise a deletion set for G can be obtained as the union of deletion sets for every component of G. Furthermore, we can assume that the degree of every vertex of G is at most $k+c$. Assume that this is not the case, i.e., there is a vertex v of G with more than $k+c$ neighbors. Then v has to be contained in the deletion set, because otherwise the deletion set would have to contain more than k of its neighbors. Consequently, the instance $\langle G \setminus \{v\}, k-1\rangle$ is equivalent to the instance $\langle G, k\rangle$.

We now show that if $\langle G, k\rangle$ is a Yes-instance, then G can have at most $k+k(k+c)c$ vertices. Since we can safely return an arbitrary trivial No-instance of small size in case G contains more than $k+k(k+c)c$ vertices, it follows that the problem admits a kernel with at most $k+k(k+c)c$ vertices. Assume that $\langle G, k\rangle$ is a Yes-instance and let R be a deletion set of size at most k witnessing this. Because every vertex in R has at most $k+c$ neighbors and G is connected, we obtain that $G \setminus R$ has at most $k(k+c)$ components and hence G has at most $k+k(k+c)c$ vertices, as required. Moreover, because the maximum degree of G is at most $k+c$, we obtain that G has at most $(k+c)(k+k(k+c)c)$ edges. □

In the next result we show that although finding a small backdoor is fpt, the evaluation remains hard for this type of backdoor. This is shown by a reduction from the Partitioned Clique problem, which is known to be W[1]-complete [65].

Partitioned Clique	
Instance:	An integer k, a k-partite graph $G = (V, E)$ with partition $\{V_1, \ldots, V_k\}$ of V into sets of equal size.
Parameter:	k
Question:	Does G have a k-clique, i.e., a set $C \subseteq V$ of k vertices such that $\forall u, v \in C$, with $u \neq v$ there is an edge $\{u, v\} \in E$, and $\forall i \in [k]$ it holds that $\lvert C \cap V_i\rvert = 1$?

Theorem 7. *c-*Causal Evaluation[Variables] *and c-*Bounded Causal Evaluation[Variables] *are* W*[1]-hard even for planning instances with bounded domain.*

Proof. We reduce from Partitioned Clique. Let $G' = (V, E)$ be a k-partite graph where $V = \{v_1, \ldots, v_n\}$ is partitioned into $V_1, \ldots, V_k$. We construct an instance $\langle \mathbb{P}, S, k'\rangle$ of c-Bounded Causal Evaluation[Variables] in the following way.

Let $k' \coloneqq k + \binom{k}{2}$ and $k'' \coloneqq \binom{k}{2} + k'$. The variables V' are the union of three kinds of variables:

(i) the variables corresponding to the vertices $V = \{v_1, \ldots, v_n\}$ of the graph G' (note that by slight abuse of notation we also use the variables $\{v_1, \ldots, v_n\}$ for $\langle \mathbb{P}, S, k'\rangle$),

(ii) pair-variables V_p of the form $p_{i,j}$ for $1 \leq i < j \leq k$, and

(iii) counter-variables $V_c = \{c_1, \ldots, c_{k'}\}$.

The actions A are the union of two types of actions:

- First, for each $v \in V$ and $l \in [k]$ we introduce an action a_v^l with $\mathsf{pre}(a_v^l)[c_l] = 0$ and $\mathsf{eff}(a_v^l)[c_l] = \mathsf{eff}(a_v^l)[v] = 1$. This type of actions allows to select vertices forming a clique in G'.

- Second, for each l and each edge $\{v_i, v_j\} \in E$ such that $k+1 \leq l \leq k'$, $1 \leq i < j \leq n$, $v_i \in V_{i'}, v_j \in V_{j'}$, and $i', j' \in [k]$ (with $i' \leq j'$), we introduce an action $a^l_{i,j}$ with $\mathsf{pre}(a^l_{i,j})[c_l] = 0$, $\mathsf{pre}(a^l_{i,j})[v_i] = \mathsf{pre}(a^l_{i,j})[v_j] = 1$, and $\mathsf{eff}(a^l_{i,j})[c_l] = \mathsf{eff}(a^l_{i,j})[p_{i',j'}] = 1$.

  The intuition of this type of actions is as follows. These actions allow to set a pair-variable $p_{i',j'} \in V_p$ to 1 whenever there is an edge between $v_i \in V_{i'}$ and $v_j \in V_{j'}$ in G' and the variables v_i and v_j have been selected previously using actions of the first type.

The initial state is defined as $I = 0^{n+k''}$. In the goal G we set the variables in V_p to 1 and all others to undefined ($\mathbf{u}$). Now, let $\mathbb{P} \coloneqq \langle V', \{0,1\}, A, I, G\rangle$ and $S \coloneqq V_p \cup V_c$.

It is easy to verify that deleting the variables in S yields a causal graph where the maximum size of the components is one. Notice that since $|S| = k''$, $|S|$ is bounded by a function of k.

The instance $\mathcal{I}$ of c-Causal Evaluation[Variables] is given by $\mathcal{I} \coloneqq \langle \mathbb{P}', V_p\rangle$ where $\mathbb{P}' = \mathbb{P} \setminus V_c$, i.e., $\mathcal{I}$ is constructed by the above-mentioned reduction without using the counter-variables.

It is now straightforward to check the equivalence of the following three statements:

- $\langle \mathbb{P}', V_p\rangle$ is a Yes-instance (of c-Causal Evaluation[Variables])
- $\langle \mathbb{P}, S, k'\rangle$ is a Yes-instance (of c-Bounded Causal Evaluation[Variables])
- $\langle k, G', \{V_1, \ldots, V_k\}\rangle$ is a Yes-instance (of Partitioned Clique).

This concludes the W[1]-hardness proof for c-Causal Evaluation[Variables] and c-Bounded Causal Evaluation[Variables]. □

Note that the previous construction could be simplified for c-Bounded Causal Evaluation[Variables] by removing the counter variables.

The hardness results for c-Causal Evaluation[Variables] and c-Bounded Causal Evaluation[Variables] indicate that we should continue and consider further types of backdoors in order to obtain the desirable case where both detection and evaluation are fpt.

3.2. Action-Deletion-Backdoors

In this section we show our two main positive results, namely, that SAS$^+$ Planning for planning instances with bounded domain variables is fixed-parameter tractable parameterized by the number of actions one needs to delete to obtain a causal graph with constant size components. The same holds true for Bounded SAS$^+$ Planning even for planning instances with an unbounded domain. To obtain these results we first show that the detection problem for action-deletion-backdoors is fixed-parameter tractable (Theorem 9). We then show that this also holds true for the evaluation problem for planning instances of bounded domain and for instances of bounded planning in Corollary 12 and Corollary 14, respectively.

For the action-deletion-backdoors, i.e., the setting where actions are removed to obtain components of size at most c, the problems c-Causal Detection[Actions], c-Causal Evaluation[Actions], and c-Bounded Causal Evaluation[Actions] are defined analogously to the respective problems for variable-deletion-backdoors. Notice that S now denotes a set of actions.

c-Causal Detection[Actions]	
Instance:	A SAS$^+$ instance $\mathbb{P}$ and a positive integer k.
Parameter:	k
Question:	Is there a set S of at most k actions of $\mathbb{P}$ such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{Causal}}(\mathbb{P} \setminus S)) \leq c$.

c-Causal Evaluation[Actions]	
Instance:	A SAS$^+$ instance $\mathbb{P}$ and a set S of actions of $\mathbb{P}$ such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{Causal}}(\mathbb{P} \setminus S)) \leq c$.
Parameter:	$\lvert S\rvert$
Question:	Does $\mathbb{P}$ have a plan?

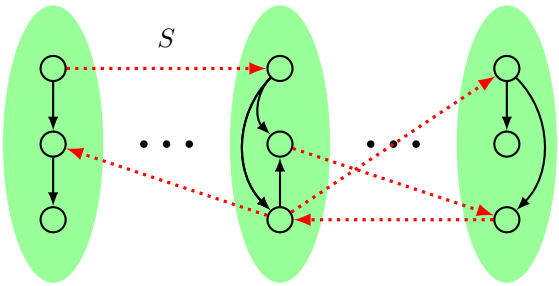


Figure 4: The causal graph of a planning instance where the actions in the set S (red dotted arcs), need to be removed such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq 3$. The set S contains all actions that introduce arcs between the highlighted components.

c-BOUNDED CAUSAL EVALUATION[ACTIONS]	
Instance:	A SAS$^+$ instance $\mathbb{P}$, a set S of variables of $\mathbb{P}$ such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq c$ and an integer k.
Parameter:	$\lvert S\rvert + k$
Question:	Does $\mathbb{P}$ have a plan of length at most k?

The property of having small component size in the causal graph is also fragile with respect to adding actions: For instance, consider the case where a "reset" action is added to a planning instance with small component size. The "reset" action sets all variables to their value in the initial state. As a result the causal graph contains only one big component. However, observe that the size of an action-deletion-backdoor for this instance is one. Based on Figure 2, Figure 4 shows the causal graph of a planning instance where the actions in the set S need to be removed such that $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq 3$.

To illustrate the applicability of action-backdoors, we will show how to model a wide variety of well-known NP-complete problems in a planning instance having a small action-deletion-backdoor set (into components of size two). Our first and simplest such problem is the well-known SHORTEST COMMON SUPERSEQUENCE problem [47, 65] (SCS), which serves as a illustrative example of the types of problems that can be naturally modeled in terms of planning instances with small action-deletion-backdoor sets. As mentioned by Jiang and Li [47] the SCS problem and its variants have applications in a wide variety of areas in AI such as automated manufacturing, bioinformatics, and syntactical pattern recognition.

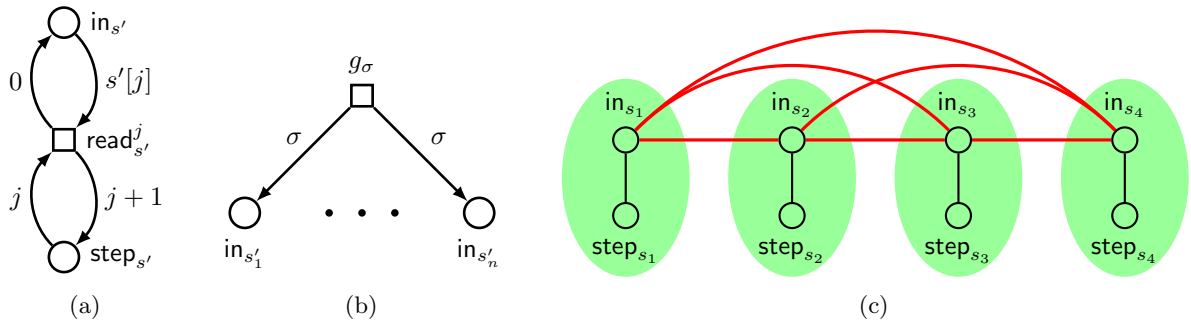
SHORTEST COMMON SUPERSEQUENCE (SCS)	
Instance:	A set S of sequences over alphabet Σ and an integer ℓ.
Question:	Is there a sequence s of length at most ℓ (over Σ) such that s' is a subsequence of s for every $s' \in S$?

Note that a sequence s' is a *subsequence* of s if s' can be obtained from s by removing elements. Given an instance $\mathcal{S} = \langle S, \Sigma, \ell \rangle$ of SCS, we will now construct a planning instance $\mathbb{P}(\mathcal{S})$ with an action-deletion-backdoor set of size at most $|\Sigma|$ into components of size two such that $\mathbb{P}(\mathcal{S})$ has a plan of length at most $\ell + \sum_{s' \in S} |s'|$ if and only if $\mathcal{S}$ has a solution. For every $s' \in S$, $\mathbb{P}(\mathcal{S})$ has two variables:

- $\mathsf{in}_{s'}$ with domain $\Sigma \cup \{0\}$ and
- $\mathsf{step}_{s'}$ with domain $\{0, \ldots, |s'| + 1\}$.

$\mathbb{P}(\mathcal{S})$ contains one "global action" g_σ for every $\sigma \in \Sigma$ such that $\mathsf{eff}(g_\sigma)[\mathsf{in}_{s'}] = \sigma$ for every $s' \in S$. Moreover, for every $s' \in S$ and j with $1 \leq j \leq |s'|$, $\mathbb{P}(\mathcal{S})$ has an action $\mathsf{read}^j_{s'}$ with:

- $\mathsf{pre}(\mathsf{read}^j_{s'})[\mathsf{step}_{s'}] = j$,
- $\mathsf{pre}(\mathsf{read}^j_{s'})[\mathsf{in}_{s'}] = s'[j]$,



(a) (b) (c)

Figure 5: (a) shows an action $\mathsf{read}^j_{s'}$ of $\mathbb{P}(\mathcal{S})$ with $s' \in S$ and $1 \leq j \leq |s'|$ where $\mathcal{S} = \langle S, \Sigma, \ell \rangle$ is an instance of SCS; (b) shows the "global action" g_σ of $\mathbb{P}(\mathcal{S})$ where $\mathcal{S} = \langle S, \Sigma, \ell \rangle$ is an instance of SCS; (c) shows the causal graph $\overline{\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}(\mathcal{S}))}$ where $\mathcal{S} = \langle S, \Sigma, \ell \rangle$ is an instance of SCS with $S = \{s_1, \ldots, s_4\}$

- $\mathsf{eff}(\mathsf{read}^j_{s'})[\mathsf{step}_{s'}] = j + 1$, and
- $\mathsf{eff}(\mathsf{read}^j_{s'})[\mathsf{in}_{s'}] = 0$.

The actions $\mathsf{read}^j_{s'}$ and g_σ as well as the causal graph $\overline{\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}(\mathcal{S}))}$ of an instance $\mathcal{S} = \langle S, \Sigma, \ell \rangle$ of SCS with $S = \{s_1, \ldots, s_4\}$ are depicted in Figure 5.

Initially all variables are set to 0 and for every $s' \in S$ the goal state requires the value $|s'| + 1$ from variable $\mathsf{step}_{s'}$. It is now easy to see that $\mathcal{S}$ has a solution if and only if $\mathbb{P}(\mathcal{S})$ has a plan of length at most $\ell + \sum_{s' \in S} |s'|$. Note that $\mathbb{P}(\mathcal{S})$ is a very simplistic example in the sense that it uses only a small subset of the features available to planning instances with small action-deletion-backdoors. Namely in $\mathbb{P}(\mathcal{S})$ the global actions do not have preconditions and the components only check very simple properties of the sequence of global actions. Using this additional power one can easily imagine that such planning instances can model more sophisticated variants of SCS and related problems, for instance, a straightforward adaptation would allow to restrict the sequence of global actions to arbitrary regular expressions.

Another very natural setting in which action-deletion-backdoors can be applied is the modeling of automated manufacturing processes, which has also already been mentioned above, more specifically modeling robotic assembly lines [32]. Here a large number of robots (or machines) takes part in a joint manufacturing process. Such processes are then usually synchronized by a central control unit. In this setting a single robot can be modeled by a component of small size, whereas the central control unit synchronizes the manufacturing process through the global actions, i.e., the actions contained in the action-deletion-backdoor set.

We start the analysis by a discussion of the complexity of the detection problem. Similar to the variable case the detection problem for actions backdoors is a problem on the causal graph of the planning instance. Namely, the detection problem can be equivalently stated as a problem on a undirected edge-labeled graph as follows. Let G be an undirected edge-labeled graph with edge-labeling function $\lambda : E(G) \to L$ for some set of labels L. For a set of labels $L' \subseteq L$, we denote by $G[L']$, the subgraph of G whose vertices are all vertices of G that are incident with an edge with label $l \in L'$ and whose edges are all edges of G with a label $l \in L'$. We say that G is *action labeled* if for every $l \in L$ it holds that the graph $G[\{l\}]$ has a *split partition into A and B*, i.e., the vertex set of $G[\{l\}]$ can be partitioned into two sets A and B such that A is an independent set in $G[\{l\}]$, B is a complete graph in $G[\{l\}]$, and $G[\{l\}]$ contains an edge between every pair (a, b) of vertices with $a \in A$ and $b \in B$.

c-Label Deletion to Small Components	
Instance:	An action labeled graph G with edge-labeling function $\lambda : E(G) \to L$ for a set of labels L and an integer k.
Parameter:	k
Question:	Is there a set $R \subseteq L$ of at most k labels such that $\mathsf{cc\text{-}size}(G \setminus \lambda^{-1}(R)) \leq c$?

To see that the problems c-Causal Detection[Actions] and c-Label Deletion to Small Components are equivalent consider the following two polynomial-time reductions. Given an instance $I = (\mathbb{P}, k)$ of c-Causal Detection[Actions], then $(\mathcal{G}_{\text{L-Causal}}(\mathbb{P}), k)$ is an instance of c-Label Deletion to Small Components that is equivalent to I. Note that $\mathcal{G}_{\text{L-Causal}}(\mathbb{P})$ is an action labeled graph, because every label l of $\mathcal{G}_{\text{L-Causal}}(\mathbb{P})$ corresponds to an action a of $\mathbb{P}$ and moreover the required split partition into A and B of $\mathcal{G}_{\text{L-Causal}}(\mathbb{P})[\{l\}]$ is given by defining B as the set of all variables occurring in an effect of a and A as the set of all variables that occur in a precondition of a and are not in B. Moreover, if $I = (G, \lambda, k)$ is an instance of c-Label Deletion to Small Components, then $(\mathbb{P}, k)$, where $\mathbb{P}$ is any planning instance such that $\overline{\mathcal{G}_{\text{L-Causal}}(\mathbb{P})}$ is isomorphic with G (including the edge-labels), is equivalent to I. Note that such an instance $\mathbb{P}$ can be obtained as follows:

- for every vertex $v \in V(G)$, $\mathbb{P}$ has one variable v with arbitrary domain,
- for every label $l \in L$ such that $G[\{l\}]$ has a split partition into A and B, $\mathbb{P}$ has one action that has an arbitrary precondition on every variable in A and an arbitrary effect on every variable in B, and
- the domain, the initial state as well as the goal state of $\mathbb{P}$ are defined arbitrarily.

To justify a parameterized complexity analysis, we start as before by showing NP-hardness. To this end, we use a reduction from the following well-known NP-complete problem [35].

3-Dimensional Matching	
Instance:	Three disjoint sets X, Y, Z of the same cardinality, a set of triples $T \subseteq (X \times Y \times Z)$, and a $k \in \mathbb{N}$.
Question:	Is there a set $T' \subseteq T$ with $\lvert T' \rvert \geq k$ such that no distinct $t, t' \in T'$ agree on at least one coordinate?

Theorem 8. *For every $c \geq 3$, c-Causal Detection[Actions] is NP-complete even for planning instances with bounded domain.*

Proof. Because of the equivalence of c-Causal Detection[Actions] and c-Label Deletion to Small Components it is sufficient to show that c-Label Deletion to Small Components is NP-complete. Because any solution to c-Label Deletion to Small Components can be verified in polynomial-time, it holds that c-Label Deletion to Small Components is in NP. Towards showing NP-hardness of c-Label Deletion to Small Components we give a polynomial-time reduction from the 3-Dimensional Matching problem. Let $\langle (X, Y, Z), T, k \rangle$ be an instance of 3-Dimensional Matching with $\lvert X \cup Y \cup Z \rvert = n$ and $\lvert T \rvert = t$ and let $c \geq 3$. We construct an instance (G, λ, k) of c-Label Deletion to Small Components as follows. The vertex set of G contains:

- one vertex v_x for every $x \in X \cup Y \cup Z$ and
- one vertex v_t^i for every $t \in T$ and every i with $3 < i \leq c$.

Moreover, for every triple $t \in T$ with $t = (x, y, z)$, G has edges between any pair of vertices in $\{v_x, v_y, v_z, v_t^4, \ldots, v_t^c\}$ with label l_t. Notice that the v_t^i variables are used to pad the size of the components to c if we reduce to c-Label Deletion to Small Components for $c > 3$. This completes the description of the reduction. It remains to show that the instance $\langle (X, Y, Z), T, k \rangle$ is a Yes-instance of 3-Dimensional Matching if and only if the instance $\langle G, \lambda, \lvert T \rvert - k \rangle$ is a Yes-instance of c-Label Deletion to Small Components.

Suppose that $\langle (X, Y, Z), T, k \rangle$ is a Yes-instance of 3-Dimensional Matching and let T' be a set of at least k triples witnessing this. It is straightforward to verify that the set $R = \{ l_t \mid t \in T \setminus T' \}$ satisfies $\mathsf{cc\text{-}size}(G \setminus \lambda^{-1}(R)) \leq c$ and hence R is a solution for $\langle G, \lambda, \lvert T \rvert - k \rangle$.

For the reverse direction suppose that $\langle G, \lambda, \lvert T \rvert - k \rangle$ is a Yes-instance of c-Label Deletion to Small Components and let R be a set of at most $\lvert T \rvert - k$ labels witnessing this. We claim that the set $T' = T \setminus \{ t \mid l_t \in R \}$ satisfies $t \cap t' = \emptyset$ for every two distinct t and t' in T'. Suppose for a contradiction that there are two distinct t and t' in T' with $t \cap t' \neq \emptyset$. It follows that the graph $G \setminus \lambda^{-1}(R)$ contains a

component that contains all vertices in $t \cup t'$ plus all the $2(c-3)$ vertices in $\{ v_p^i \mid p \in \{t, t'\} \text{ and } 3 < i \leq c \}$. Since we can assume w.l.o.g. that $t \neq t'$ and thus $|t \cup t'| \geq 4$ we obtain that this component contains at least $4 + 2(c-3)$ vertices, contradicting our assumption that all components of the graph $G \setminus \lambda^{-1}(R)$ have cardinality at most c. □

In the next theorem we will show that the detection problem is fixed-parameter tractable.

Theorem 9. *For any $c \geq 1$, c-Causal Detection[Actions] can be solved in time $\mathcal{O}(c^k|E(G)|)$ and is hence fixed-parameter tractable.*

Proof. Because of the equivalence of c-Causal Detection[Actions] and c-Label Deletion to Small Components it is sufficient to show the result for c-Label Deletion to Small Components. We will show the theorem by providing a depth-bounded search tree algorithm, which given an instance $I = \langle G, \lambda, k \rangle$ of c-Label Deletion to Small Components either determines that the instance is a No-instance or outputs a solution $R \subseteq L$ of minimum size for I. The algorithm is similar to the algorithm presented in the proof of Theorem 5 and is based on the following observations.

- O1 If G is not connected then a (minimum) solution for I can be obtained as the disjoint union of (minimum) solutions for every component of G.
- O2 If G is connected and C is any set of at most c labels of G such that $G[C]$ contains a component of size at least $c+1$, then any solution for I has to contain at least one label from C.

Using the above observations the algorithm first checks whether G is connected. If G is not connected the algorithm calls itself recursively on the instance (C, k) for each component C of G. If one of the recursive calls returns No or if the size of the union of the (minimum) solutions returned for each component exceeds k, the algorithm returns that I is a No-instance. Note that for this step of the algorithm it is crucial that the algorithm returns a minimum solution (instead of returning an arbitrary solution of size at most k) for any Yes-instance (G, λ, k). Otherwise the algorithm returns the union of the (minimum) solutions returned for each component of G.

If G is connected and $|V(G)| \leq c$, the algorithm returns the empty set as a solution. Otherwise, i.e., if G is connected but $|V(G)| > c$, the algorithm first computes a set C of at most c labels of G such that $G[C]$ contains a component of size at least $c+1$. This can for instance be achieved as follows. First a Depth-First Search that starts at any vertex of G and stops as soon as $c+1$ vertices have been visited is used to compute a set O of $c+1$ vertices that are connected in G. The desired set C of at most c labels is then obtained from O as the set of labels used by the edges of any spanning tree of $G[O]$. The algorithm then branches on the labels in C, i.e., for every $l \in C$ the algorithm recursively computes a solution for the instance $(G[L \setminus \{l\}], k-1)$. It then returns the solution of minimum size returned by any of those recursive calls, or No if none of those calls returns a solution. This completes the description of the algorithm. The correctness of the algorithm follows immediately from the above observations. Moreover the running time of the algorithm is easily seen to be dominated by the maximum time required for the case that G is connected at each step of the algorithm. In this case the running time can be obtained as the product of the number of recursive calls and the time spent on each of those. Because at each recursive call the parameter k is decreased by one and the number of recursive calls is at most c, we obtain that there are at most c^k recursive calls. Furthermore, the time at each recursive call is dominated by the time required to check whether G is connected, which is linear in the number of edges of G. Putting everything together, we obtain $\mathcal{O}(c^k|E(G)|)$ as the total time required by the algorithm, which completes the proof. □

Like c-Causal Detection[Variables], also Causal Detection[Actions] admits a polynomial kernel.

Theorem 10. *For any $c \geq 1$, c-Causal Detection[Actions] admits a polynomial kernel of size at most $\mathcal{O}((k + c^2k)^2)$.*

Proof. In view of the equivalence of c-Causal Detection[Actions] and c-Label Deletion to Small Components, it is sufficient to show the existence of a kernel for c-Label Deletion to Small Components. Thus let (G, λ, k) be an instance of c-Label Deletion to Small Components with label set L.

W.l.o.g. we can assume that G is connected, otherwise a deletion set for G can be obtained as the union of deletion sets for every component of G. Consider a label $l \in L$. W.l.o.g. we can assume that G contains at least one edge $e \in E(G)$ with $\lambda(e) = l$, otherwise we could simply delete l from L without changing the instance. Moreover, because G is action labeled, it follows that $G[\{l\}]$ is connected. We claim that we can assume that $G[\{l\}]$ contains at most c vertices. This is because otherwise l would have to be contained in the deletion set and we could reduce the instance to the equivalent but smaller instance $\langle G \setminus \{\lambda^{-1}(l)\}, k-1\rangle$.

We now show that if $\langle G, k\rangle$ is a YES-instance, then G can have at most c^2k vertices. Since we can safely return an arbitrary trivial NO-instance of small size in case (G, k) contains more than c^2k vertices, it follows that the problem admits a kernel with at most c^2k vertices. Assume that $\langle G, \lambda, k\rangle$ is a YES-instance and let R be a deletion set of size at most k witnessing this. Because for every label $l \in R$ it holds that $G[\{l\}]$ has at most c vertices and because G is connected, we obtain that $G \setminus R$ has at most ck components, which in turn implies that G has at most $k + c^2k$ vertices. It follows that G has at most $(k + c^2k)^2/2$ edges, which completes the proof of the theorem. □

After these promising results for the detection problem, we turn to the evaluation problem. In case neither the plan length nor the size of the domain of the variables is bounded, evaluation remains hard even for action-deletion-backdoors. However, if either the domain or the plan length is bounded we can show for each case an fpt-result for the evaluation problem.

We start by showing that c-CAUSAL EVALUATION[ACTIONS] is fixed-parameter tractable for planning instances with bounded domain. In fact we show a stronger result, namely, that even computing a shortest plan is fixed-parameter tractable and moreover the result still holds for instances of unbounded domain as long as the size of the domain is considered as an additional parameter.

Theorem 11. *Let $\langle \mathbb{P}, S\rangle$ be an instance of c-CAUSAL EVALUATION[ACTIONS] and let k be an integer. Then the problem of deciding whether $\mathbb{P}$ has a plan of length at most k is fixed-parameter tractable parameterized by $|S|$ and $|D|$.*

Proof. Let $\langle \mathbb{P}, S\rangle$ with $\mathbb{P} = \langle V, D, A, I, G\rangle$ be an instance of c-CAUSAL EVALUATION[ACTIONS], k an integer, and let $d = |D|$.

We say that two SAS$^+$ instances $\mathbb{P}_1 = \langle V_1, D_1, A_1, I_1, G_1\rangle$ and $\mathbb{P}_2 = \langle V_2, D_2, A_2, I_2, G_2\rangle$ are *isomorphic* if there is a bijection α from $V_1 \cup D_1$ to $V_2 \cup D_2$ such that:

(I1) $\alpha(v) \in V_2$ for every $v \in V_1$ and $\alpha(d') \in D_2$ for every $d' \in D_1$,

(I2) for every $v \in V_1$ it holds that $\alpha(I_1[v]) = I_2[\alpha(v)]$ and $\alpha(G_1[v]) = G_2[\alpha(v)]$.

(I3) there is a bijection β from A_1 to A_2 such that for every action $a \in A_1$ it holds that $\alpha(\mathsf{pre}(a)[v]) = \mathsf{pre}(\beta(a))[\alpha(v)]$ and $\alpha(\mathsf{eff}(a)[v]) = \mathsf{eff}(\beta(a))[\alpha(v)]$ for every variable $v \in V_1$.

We call such a bijection α an *isomorphism* between $\mathbb{P}_1$ and $\mathbb{P}_2$. Let C_1 and C_2 be two weakly connected components of the graph $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$. We say that C_1 and C_2 are *globally equivalent*, denoted by $C_1 \equiv C_2$, if there is an isomorphism α between $\mathbb{P}(C_1)$ and $\mathbb{P}(C_2)$ such that:

(GE) for every action $s \in S$ it holds that $\alpha(\mathsf{pre}(s)[v]) = \mathsf{pre}(s)[\alpha(v)]$ and $\alpha(\mathsf{eff}(s)[v]) = \mathsf{eff}(s)[\alpha(v)]$ for every variable v of $V(\mathbb{P}(C_1))$.

Claim 1. The number of equivalence classes of the components of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$, with respect to $\equiv$, is at most $c_G = c \cdot ((d+1)^{2c})^{|S|+1} \cdot 2^{(d+1)^{2c}}$.

The claim follows from the following observations for every component C of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$:

- C has at most c variables,
- there are at most d^c potential initial states for $\mathbb{P}(C)$,
- there are at most $(d+1)^c$ potential goal states for $\mathbb{P}(C)$; note that together with the previous observation it follows that there are at most $d^c(d+1)^c \leq (d+1)^{2c}$ potential combinations of initial and goal state for $\mathbb{P}(C)$,

- there are at most $(d+1)^{2c}$ potential combinations of preconditions and effects for any action in S on the variables of C,
- there are at most $((d+1)^{2c})^{|S|}$ potential ways in which the actions in S can interact with the instance $\mathbb{P}(C)$,
- there are at most $(d+1)^{2c}$ possible distinct actions in $\mathbb{P}(C)$,
- there are at most $2^{(d+1)^{2c}}$ distinct sets of actions for $\mathbb{P}(C)$.

Informally, if two components C_1 and C_2 are globally equivalent then the same types of sub-plans can be used for C_1 and C_2 within a plan for $\mathbb{P}$. More formally the *sub-plan* of a plan ω for $\mathbb{P}$ for a component C of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$, denoted by $\omega[C]$, is the sequence $\omega[A(\mathbb{P}(C))]$. The following observation formalizes this intuition.

Observation 1. Let ω be a plan for $\mathbb{P}$ and let C_1 and C_2 be two globally equivalent components of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P}\setminus S)$. Then the sequence ω' obtained from ω after removing $\omega[C_2]$ from ω and replacing every occurrence of an action a of $\mathbb{P}(C_1)$ in ω with the sequence $(a, \beta(a))$ is also a plan for $\mathbb{P}$. Here β is a bijection from $A(\mathbb{P}(C_1))$ to $A(\mathbb{P}(C_2))$ that satisfies (I3) and exists because C_1 and C_2 are globally equivalent.

The following claim now follows from a repeated application of the above observation.

Claim 2. Let ω be a plan for $\mathbb{P}$. Then there is a plan ω' for $\mathbb{P}$ with $|\omega'| \leq |\omega|$ such that $|\omega'[C_1]| = |\omega'[C_2]|$ for every two globally equivalent components C_1 and C_2 of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$.

In the following let $\mathcal{P}$ be the unique partition of the components of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$ into equivalence classes w.r.t. $\equiv$ and for every $P \in \mathcal{P}$ let $C(P)$ be an arbitrary component in P. Let $\mathbb{P}'$ be the SAS$^+$ instance $\mathbb{P}[\bigcup_{P\in\mathcal{P}} C(P)]$ and let $\mathbb{P}_R$ be the SAS^+_C instance $\mathbb{P}'$, whose cost-function γ_R is defined by setting $\gamma_R(a) = 1$ if $a \in S$ and $\gamma_R(a) = |P|$ if $a \in A(\mathbb{P}(C(P)))$ for some $P \in \mathcal{P}$.

Claim 3. $\mathbb{P}$ has a plan of length at most k if and only if $\mathbb{P}_R$ has a plan of cost at most k. Moreover, a plan for $\mathbb{P}$ can be obtained from a plan for $\mathbb{P}_R$ in polynomial-time (w.r.t. the size of $\mathbb{P}$).

Towards showing the forward direction, let ω be a plan for $\mathbb{P}$ of length at most k. Because of Claim 2 we can assume that $|\omega[p_1]| = |\omega[p_2]|$ for every $p_1, p_2 \in P$ and every $P \in \mathcal{P}$. Then $\omega_R = \omega[S \cup \bigcup_{P\in\mathcal{P}} A(\mathbb{P}_R(C(P)))]$ is a plan for $\mathbb{P}_R$. Moreover

$$\begin{aligned}
\gamma_R(\omega_R) &= \gamma_R(\omega_R[S]) + \sum_{P\in\mathcal{P}} \gamma_R(\omega_R[C(P)]) \\
&= |\omega[S]| + \sum_{P\in\mathcal{P}} |\omega[C(P)]| \cdot |P| \\
&= |\omega[S]| + \sum_{p\in P \wedge P\in\mathcal{P}} |\omega[p]| \\
&= |\omega|
\end{aligned}$$

as required.

For the reverse direction, let ω_R be a plan for $\mathbb{P}_R$ and for every $P \in \mathcal{P}$ and every $p \in P \setminus \{C(P)\}$ let β_p be a bijection from $A(\mathbb{P}(C(P)))$ to $A(\mathbb{P}(p))$ that satisfies (I3); which exists because p and $C(P)$ are globally equivalent. Moreover let ω be the sequence of actions obtained from ω_R after doing the following for every $P \in \mathcal{P}$ with $\{p_1, \ldots, p_r\} = P \setminus C(P)$: replace every occurrence of an action a of $\mathbb{P}(C(P))$ in ω_R with the sequence $\langle a, \beta_{p_1}(a), \ldots, \beta_{p_r}(a)\rangle$. Then using the same ideas underlying Observation 1, we obtain that ω is a

plan for $\mathbb{P}$. Moreover

$$\begin{aligned}
|\omega| &= |\omega[S]| + \sum_{p \in P \wedge P \in \mathcal{P}} |\omega[p]| \\
&= |\omega[S]| + \sum_{P \in \mathcal{P}} |\omega[C(P)]||P| \\
&= \gamma_R(\omega_R[S]) + \sum_{P \in \mathcal{P}} \gamma_R(\omega_R[C(P)]) \\
&= \gamma_R(\omega_R)
\end{aligned}$$

as required.

Because of Claim 3, it is sufficient to decide whether $\mathbb{P}_R$ has a plan of cost at most k, which can be achieved using the following three steps.

1) Compute the partition $\mathcal{P}$ into global component types, i.e., equivalence classes w.r.t. $\equiv$, of the components in $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$.

2) Construct the SAS^+_C instance $\mathbb{P}_R$ from $\mathbb{P}$.

3) Compute a cost-optimal plan for $\mathbb{P}_R$ and return YES if and only if the cost of the plan does not exceed k.

The correctness of the algorithm follows immediately from Claim 3.

The running time of the algorithm is obtained as follows. Computing the global type of each component (Step 1) can be achieved by checking for each pair C_1 and C_2 of components of $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$ whether they are equivalent w.r.t. $\equiv$. This can be done by going over all of the at most $c!d!$ many potential isomorphisms between C_1 and C_2 and checking for each whether it constitutes an isomorphism that additionally satisfies (GE), which in turn can be done in time $\mathcal{O}((d+1)^{4c}c + |S|c)$ by testing whether an isomorphism α satisfies (I1)–(I3) and (GE) as follows:

(Checking I1) Checking whether α satisfies (I1) can be done in time $\mathcal{O}(c + d)$,

(Checking I2) Checking whether α satisfies (I2) can be done in time $\mathcal{O}(c)$,

(Checking I3) Checking whether α satisfies (I3) can be done in time $\mathcal{O}((d+1)^{4c}c)$ as follows. First check whether $|A(\mathbb{P}(C_1))| = |A(\mathbb{P}(C_2))|$, if not output NO. Otherwise, check for every action $a \in A(\mathbb{P}(C_1))$ whether there is an action $a' \in A(\mathbb{P}(C_2))$ such that $\alpha(\mathsf{pre}(a)[v]) = \mathsf{pre}(a')[\alpha(v)]$ and $\alpha(\mathsf{eff}(a)[v]) = \mathsf{eff}(a')[\alpha(v)]$ for every variable $v \in V_1$. If not output NO, otherwise output YES. Note that the above can be achieved in time $\mathcal{O}(|A(\mathbb{P}(C_1))| \cdot |A(\mathbb{P}(C_2))| \cdot |V(\mathbb{P}(C_1))|)$, which is at most $\mathcal{O}((d+1)^{4c}c)$ because (as observed in the proof of Claim 1), $\mathbb{P}(C_1)$ and $\mathbb{P}(C_2)$ each contain at most $(d+1)^{2c}$ distinct actions.

(Checking GE) Checking whether α satisfies (GE) can be done in time $\mathcal{O}(|S|c)$.

Hence the total running time to execute Step 1 of the algorithm is at most $\mathcal{O}(c!d!((d+1)^{4c}c + |S|c)|V|^2)$.

Step 2 can be executed in time $\mathcal{O}(|V|)$. Finally Step 3 can be achieved in at most $\mathcal{O}(d^{2(c \cdot c_G)})$, by using for instance Dijkstra's algorithm on the state-transition graph of $\mathbb{P}_R$, because $\mathbb{P}_R$ has at most $c \cdot c_G$ variables. Hence the total running time of the algorithm is $\mathcal{O}(c!d!((d+1)^{4c}c + |S|c)|V|^2 + d^{2(c \cdot c_G)})$, which shows that deciding whether $\mathbb{P}$ has a plan of length at most k is fixed-parameter tractable parameterized by $|S|$ and $|D|$. □

Observe that the algorithm given in the proof of Theorem 11 suggests an interesting preprocessing procedure for planning. Namely, suppose you are given an instance of planning with only a few global actions. Then by using an efficient procedure to identify globally equivalent components (even a heuristic procedure would be sufficient at this point), one can reduce the size of the instance by removing all but one component

from each equivalence class. The resulting, potentially much smaller, planning instance can then be solved by any suitable planner.

At a first glance one might be tempted to think that the main idea behind the proof of Theorem 11, i.e., the categorization of components into equivalence classes, can also be employed to solve instances of c-Bounded Causal Evaluation[Variables] with bounded domain. However, as it is shown in Theorem 7, this is not the case. The intuitive reason why this is not possible is that in the case of a variable backdoor set, there can still be an unbounded number of "global actions", i.e., actions that depend on (in this case have a precondition on variables of) arbitrary many components, which means that the number of component types can no longer be bounded in terms of the size of the backdoor set.

As a corollary of Theorem 11, we obtain:

Corollary 12. *c-Causal Evaluation[Actions] is fixed-parameter tractable for planning instances with bounded domain.*

It might be interesting to note the implications of Theorem 11 for our example instance $\mathbb{P}(\mathcal{S})$ and by extension for the SCS problem, as introduced earlier in this section, as well as related problems. Recall that for a given instance $\mathcal{S} = \langle S, \Sigma, \ell \rangle$ of SCS, the instance $\mathbb{P}(\mathcal{S})$ has a plan of length at most $\ell + \sum_{s' \in S} |s'|$ if and only if $\mathcal{S}$ has a solution. Because of Theorem 11 deciding whether $\mathbb{P}(\mathcal{S})$ has a plan of length at most $\ell + \sum_{s' \in S} |s'|$ is fixed-parameter tractable parameterized by $|S|$ and $|D|$. Since $|S| \leq |\Sigma|$ and $|D| \leq \max\{|\Sigma| + 1\} \cup \{\ell_S\}$, where $\ell_S = \max\{\, |s'| + 1 \mid s' \in S \,\}$, it now follows that SCS is fixed-parameter tractable parameterized by $|\Sigma|$ and the maximum length ℓ_S of any sequence in S. In other words Theorem 11 gives an efficient algorithm for instances of SCS (and related problems), where both the size of the alphabet as well as the maximum length of any sequence in S is small.

We will show next that the evaluation problem is also fixed-parameter tractable for bounded planning even without bounding the domain of the planning instances. In fact we show something much stronger, namely, that in order to obtain tractability it is sufficient to only bound the length of the sequence of global actions occurring in a plan; instead of the length of the whole plan.

Theorem 13. *Let $\langle \mathbb{P}, S \rangle$ be an instance of c-Causal Evaluation[Actions] and let q be an integer. Then there is an algorithm that decides whether $\mathbb{P}$ has a plan ω with $|\omega[S]| \leq q$ and if so outputs a shortest such plan in time $\mathcal{O}((|S|+1)^q |V| (q+2)^2 |D|^{3c})$.*

Proof. Let $\langle \mathbb{P}, S \rangle$ be an instance of c-Causal Evaluation[Actions] and let $C_1, \ldots, C_m$ be the components of $\mathcal{G}_{\text{Causal}}(\mathbb{P} \setminus S)$. For every i with $1 \leq i \leq m$ we denote by $\mathbb{P}_i$ the planning instance $\mathbb{P}(C_i)$. We say that two sequences ω and ω' of actions of $\mathbb{P}$ are *globally compatible* if $\omega[S] = \omega'[S]$.

Our algorithm is based on the following two observations:

- O1 The number of possible sequences of global actions (actions in S) occurring in any plan ω with $|\omega[S]| \leq q$ for $\mathbb{P}$ is at most $\sum_{0 \leq l \leq q} \binom{q}{l} |S|^l \leq (|S|+1)^q$.
- O2 For a fixed sequence $\omega_G = \langle a_1, \ldots, a_l \rangle$ of global actions with $0 \leq l \leq q$, one only needs to consider plans of the form:

$$\omega = \langle \underbrace{\ldots}_{\omega^1}, a_1, \underbrace{\ldots}_{\omega^2}, a_2, \ldots, a_l, \underbrace{\ldots}_{\omega^{l+1}} \rangle$$

  Because each of the sub-plans ω^i for i with $1 \leq i \leq l+1$ contains no global actions, it follows that these sub-plans can be computed independently for each sub-instance $\mathbb{P}_j$ for every j with $1 \leq j \leq m$.

Based on the above observations our algorithm for c-Bounded Causal Evaluation[Actions] works as follows. For each of the at most $(|S|+1)^q$ sequences $\omega_G = \langle a_1, \ldots, a_l \rangle$ of global actions from S the algorithm computes a shortest plan ω for $\mathbb{P}$ that is globally compatible with ω_G, i.e., $\omega[S] = \omega_G$. Then a shortest plan for $\mathbb{P}$ is obtained as a shortest plan obtained for any such sequence ω_G. Next we show how to compute a shortest plan for $\mathbb{P}$ that is globally compatible with a given sequence $\omega_G = \langle a_1, \ldots, a_l \rangle$ of global actions.

Claim 4. For any sequence $\omega_G = \langle a_1, \ldots, a_l \rangle$ of global actions, there is an algorithm that computes a shortest plan ω for $\mathbb{P}$ that is globally compatible with ω_G in time $\mathcal{O}(|V|(l+2)^2 d^{3c})$.

In line with Observation O2, we will compute a shortest plan ω_j that is globally compatible with ω_G for each instance $\mathbb{P}_j$ independently. Note that each such plan ω_j has the following form:

$$\omega_j = \omega_j^1\langle a_1\rangle\omega_j^2\langle a_2\rangle\cdots\langle a_l\rangle\omega_j^{l+1}$$

where for every i with $1 \leq i \leq l+1$, ω_j^i is a sequence of actions from $\mathbb{P}_j$. Moreover, after the plans $\omega_1, \ldots, \omega_m$ for the subinstances $\mathbb{P}_1, \ldots, \mathbb{P}_m$ have been computed, the desired shortest plan ω for $\mathbb{P}$ that is globally compatible with ω_G is obtained by setting:

$$\omega = \omega_1^1\cdots\omega_m^1\langle a_1\rangle\omega_1^2\cdots\omega_m^2\langle a_2\rangle\cdots\omega_1^l\cdots\omega_m^l\langle a_l\rangle\omega_1^{l+1}\cdots\omega_m^{l+1}$$

Consequently, it only remains to show how we can compute a shortest plan ω_j for $\mathbb{P}_j$ that is globally compatible with ω_G. Towards showing this first note that because the state-transition graph of $\mathbb{P}_j$ has at most d^c states, where d is the maximum domain size of any variable in $\mathbb{P}_j$, it is possible to compute a shortest plan between any pair of states of $\mathbb{P}_j$ using for instance Dijkstra's algorithm in time $\mathcal{O}(d^{3c})$. We now define an auxiliary directed graph H that will allow use to compute a shortest plan ω_j for $\mathbb{P}_j$ that is globally compatible with ω_G as follows. H is a directed graph with positive integer weights, given by the function $w : E(H) \to \mathbb{N}$, on its edges. Moreover, H has the following vertices:

- the vertex v_I, which will represent the initial state of $\mathbb{P}_j$,
- the vertex v_i^s, for every i with $1 \leq i \leq l$ and every state s of $\mathbb{P}_j$ that is compatible with the precondition of the action a_i,
- the vertex v_{l+1}^s, for every goal state s of $\mathbb{P}_j$.

Furthermore, H has an arc with weight w from the vertex v_I to a vertex v_1^s (for some state s of $\mathbb{P}_j$) if and only if w is the length of a shortest plan for $\mathbb{P}_j$ from the initial state to the state s. Finally, H has an arc with weight w from a vertex v_i^s to a vertex $v_{i+1}^{s'}$ (for some i with $1 \leq i \leq l$ and some states s and s' of $\mathbb{P}_j$) if and only if the action a_i is applicable in the state s and w is the length of a shortest plan for $\mathbb{P}_j$ from the state $\text{res}(s, a_i)$ to the state s'. Note that H has at most $1 + (l+1)d^c$ vertices and at most $(l+2)d^{2c}$ edges and using our observation above can be constructed in time $\mathcal{O}((l+2)d^{2c} + d^{3c})$.

We claim that H has a path from v_I to some vertex v_{l+1}^s with length w if and only if $\mathbb{P}_j$ has a plan of length $w + l$ that is globally compatible with ω_G. This then shows that finding a shortest plan for $\mathbb{P}_j$ that is globally compatible with ω_G is equivalent to finding a shortest path from v_I to some vertex v_{l+1}^s, which in turn can be achieved (e.g. by using Dijkstra's algorithm) in time $\mathcal{O}((l+2)^2 d^{3c})$. Towards showing that H has a path from v_I to some vertex v_{l+1}^s with length w if and only if $\mathbb{P}_j$ has a plan of length $w + l$ that is globally compatible with ω_G, let P be a path from v_I to some vertex $v_{l+1}^{s_{l+1}}$ with weight w in H. Because of the structure of H, we obtain that P has the form $(v_I, v_1^{s_1}, v_2^{s_2}, \ldots, v_{l+1}^{s_{l+1}})$. Let ω_j^0 be the plan for $\mathbb{P}_j$ of length $w(v_I, v_1^{s_1})$ from the initial state to the state s_1 in $\mathbb{P}_j$ and for every i with $1 \leq i \leq l$, let ω_j^i be the plan for $\mathbb{P}_j$ of length $w(v_i^{s_i}, v_{i+1}^{s_{i+1}})$ from the state $\text{res}(s_i, a_i)$ to the state s_{i+1}. Note that the definition of H ensures the existence of the plans ω_j^i and also ensures that for every i with $1 \leq i \leq l$, it holds that a_i is applicable in s_i. Consequently,

$$\omega_j = \omega_j^0\langle a_1\rangle\omega_j^1\langle a_2\rangle\cdots\langle a_l\rangle\omega_j^l$$

is a plan for $\mathbb{P}_j$ whose length is $w + l$, as required.

Towards showing the reverse direction let ω_j be a plan of $\mathbb{P}_j$ that is globally compatible with ω_G of length w. Because ω_j is globally compatible with ω_G it must have the form:

$$\omega_j = \omega_j^0\langle a_1\rangle\omega_j^1\langle a_2\rangle\cdots\langle a_l\rangle\omega_j^l$$

where for every i with $0 \leq i \leq l$, ω_j^i contains no global actions. Let s_i be the state of $\mathbb{P}_j$ that is reached after the execution of ω_j up to (not including) the occurrence of a_i and let s_{l+1} be the state reached after the execution of ω_j. Because of the definition of H, it follows that $(v_I, v_1^{s_1})$ is an arc in H with weight $|\omega_j^0|$ and

similarly for every i with $1 \leq i \leq l$, $(v_i^{s_i}, v_{i+1}^{s_{i+1}})$ is an arc in H with weight $|\omega_j^i|$. Hence H contains the path $(v_I, v_1^{s_1}, \ldots, v_{l+1}^{s_{l+1}})$ and its length is equal to $w - l$, as required. This concludes the proof of the claim.

The total running time of the algorithm is now the time required to enumerate all sequences of global actions ($\mathcal{O}((|S|+1)^q)$) times the time required to compute a shortest plan that is compatible with a sequence of global actions, which due to Claim 4 is at most $\mathcal{O}(|V|(q+2)^2 d^{3c})$. □

Theorem 13 shows fixed-parameter tractability for c-Causal Evaluation[Actions], when considering the number of global actions in a plan (q) as an additional parameter. Interestingly, a bound on q can also be obtained for the seemingly unrelated setting considered in Theorem 11, i.e., when considering the maximum domain value $|D|$ as an additional parameter. This is because the number of global actions used by a plan ω for $\mathbb{P}$ that is obtained from a plan ω_R for $\mathbb{P}_R$, as described in the proof of Theorem 11, is the same as the number of global actions in ω_R, which is at most $|D|^{c \cdot c_G}$, since $\mathbb{P}_R$ contains at most $c \cdot c_G$ variables. This shows that q is at most $|D|^{c \cdot c_G}$ and hence bounded only in terms of the parameters considered in Theorem 11. In other words, the underlying reason for tractability in both results is that the number of global actions in any plan is bounded in terms of the considered parameters. Note, however, that the algorithm given in Theorem 11 is by far more efficient, than the algorithm one would obtain from Theorem 13 by using $|D|^{c \cdot c_G}$ as an upper bound for q.

The discussed relation between the two tractability results is also highly relevant for the implications of the two results for our example instance $\mathbb{P}(\mathcal{S})$, where $\mathcal{S} = \langle S, \Sigma, \ell \rangle$. Namely, Theorem 13 shows that we can decide whether our example instance $\mathbb{P}(\mathcal{S})$ has a plan of length at most $\ell + \sum_{s' \in S} |s'|$ in time $\mathcal{O}((|\Sigma|+1)^\ell |S| (\ell+2)^2 d^6)$, where $d = \max\{|\Sigma|+1\} \cup \{\ell_S\}$ and $\ell_S = \max\{\, |s'|+1 \mid s' \in S \,\}$. In terms of parameterized complexity this implies that SCS is fixed-parameter tractable parameterized by $|\Sigma|$ and ℓ. On the other hand, Theorem 11 implies that SCS is fixed-parameter tractable parameterized by $|\Sigma|$ and ℓ_S. Since ℓ can always be assumed to be at least ℓ_S, this would at a first glance indicate that Theorem 11 is more general than Theorem 13 at least as far as its implications for SCS are concerned. However, since also $\ell \leq |\Sigma| \ell_S$ (because any sequence of length at most ℓ_S is contained in the supersequence that repeats all characters of the alphabet (Σ) ℓ_S times) both results are actually equivalent w.r.t. to their implications for the parameterized complexity of SCS. Moreover, when using $\ell \leq |\Sigma| \ell_S$, the algorithm given in Theorem 13 is actually much more efficient than its counterpart given in Theorem 11. This is, however, only due to the simplicity of SCS in comparison to the underlying planning problem and will not carry over to more complex variants of SCS or other application areas.

Because the total length of a plan is a trivial upper bound for any contained sequence of global actions, we obtain our tractability result for c-Bounded Causal Evaluation[Actions].

Corollary 14. *c-Bounded Causal Evaluation[Actions] can be solved in time $\mathcal{O}((|S|+1)^k |V| (k+2)^2 |D|^{3c})$ and is hence fixed-parameter tractable.*

Corollary 12 and 14 show in combination with Theorem 9 that the backdoor approach can indeed be used to obtain new fpt-algorithms for planning problems. This is because, under action-deletion-backdoors the detection problem as well as two important evaluation problems are efficiently solvable – as long as the size of the backdoor is moderate. The fpt-algorithm for planning works as follows. First we search for a small backdoor (detection phase), which is then used in the following step to solve the given planning instance (evaluation phase).

To complement the picture for the evaluation problem for action-deletion backdoors, we now show our hardness result, which shows that c-Causal Evaluation[Actions] is paraNP-hard in the general case.

Theorem 15. *9-Causal Evaluation[Actions] is paraNP-hard even if the global actions (the actions in the backdoor S) have no preconditions.*

Proof. We will start by giving an informal description of the proof. The proof is via a reduction from 3-SAT. Let Φ be a 3-CNF formula with variables $x_1, \ldots, x_n$. We construct an instance $\langle \mathbb{P}, S \rangle$ of 9-Causal Evaluation[Actions] where S contains 5 "global actions" $s_1, \ldots, s_5$ and $\mathcal{G}_{\text{Causal}}(\mathbb{P}) \setminus S$ contains one "local" component for each clause of Φ and one "global" component (each component contains at most 9

variables). The main idea is that the global component guesses an assignment α for the variables of Φ and forces a sequence of global actions that communicates α to the components of the clauses. Note that the communication between the global component and the local components of the clauses can only be achieved through global actions. Moreover, since we are only allowed to use a constant number of global actions, the only way to encode the assignment α is to employ long sequences of global actions. In particular, for every variable x_i, if $\alpha(x_i) = 0$, then the following sequence of global actions is forced by the global component:

$$\langle \underbrace{s_1, s_3, \ldots, s_1, s_3}_{(i-1)\text{-times } s_1,s_3}, s_4, s_3, \underbrace{s_1, s_3, \ldots, s_1, s_3}_{(n-i)\text{-times } s_1,s_3}, s_5 \rangle \tag{1}$$

Otherwise, i.e., if $\alpha(x_i) = 1$, then the following sequence of global actions is forced by the global component:

$$\langle \underbrace{s_2, s_3, \ldots, s_2, s_3}_{(i-1)\text{-times } s_2,s_3}, s_4, s_3, \underbrace{s_2, s_3, \ldots, s_2, s_3}_{(n-i)\text{-times } s_2,s_3}, s_5 \rangle \tag{2}$$

Observe that the above sequences uniquely identify the variable x_i together with its assignment $\alpha(x_i)$ for every variable x_i. The total sequence of global actions forced by the global component is then the concatenation of the above sequences for every variable x_i. The components of the clauses now ensure that every clause is satisfied by the assignment chosen by the global component. They do so by allowing only sequences of global actions that correspond to a satisfied literal of the respective clause to lead to their (local) goal state.

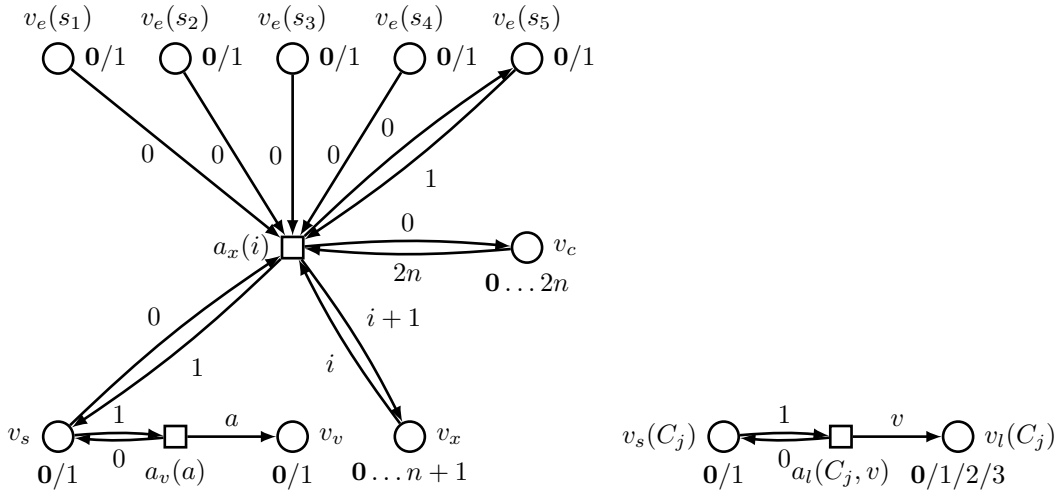


Figure 6: Left: The actions $a_v(a)$ and $a_x(i)$ for every $a \in \{1, 2\}$ and $i \in \{1, \ldots, n\}$. Right: The actions $a_l(C_j, v)$ for every $v \in \{1, 2, 3\}$. Here and in the following figures squares represent actions and circles represent variables. Every variable is also labeled with its domain and its value in the initial state is given in bold face. A labeled arc from a variable to an action indicates a precondition, whereas a labeled arc from an action to a variable indicates an effect.

We now present the reduction in detail and then show its correctness. Let Φ be a 3-CNF formula with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. To make the presentation easier to follow we will sometimes refer to the i-th literal of some clause C_j. This should be understood with respect to some arbitrary but fixed ordering of the literals of C_j. We construct in polynomial-time an instance $\langle \mathbb{P}, S \rangle$ with $\mathbb{P} = \langle V, D, A, I, G \rangle$ and $|S| = 5$ of 9-Causal Evaluation[Actions] such that Φ is satisfiable if and only if there is a plan for $\mathbb{P}$ as follows. We set $S = \{s_1, \ldots, s_5\}$, $D = \{0, \ldots, 2n\}$. We start by introducing the variables and actions belonging to the global component. The global component contains the following variables:

- A binary variable v_s, whose purpose is to ensure that the assignment of a variable x_i of the formula Φ can only be made before the start of the sequence of global actions, which communicates the assignment to the local components of the clauses.

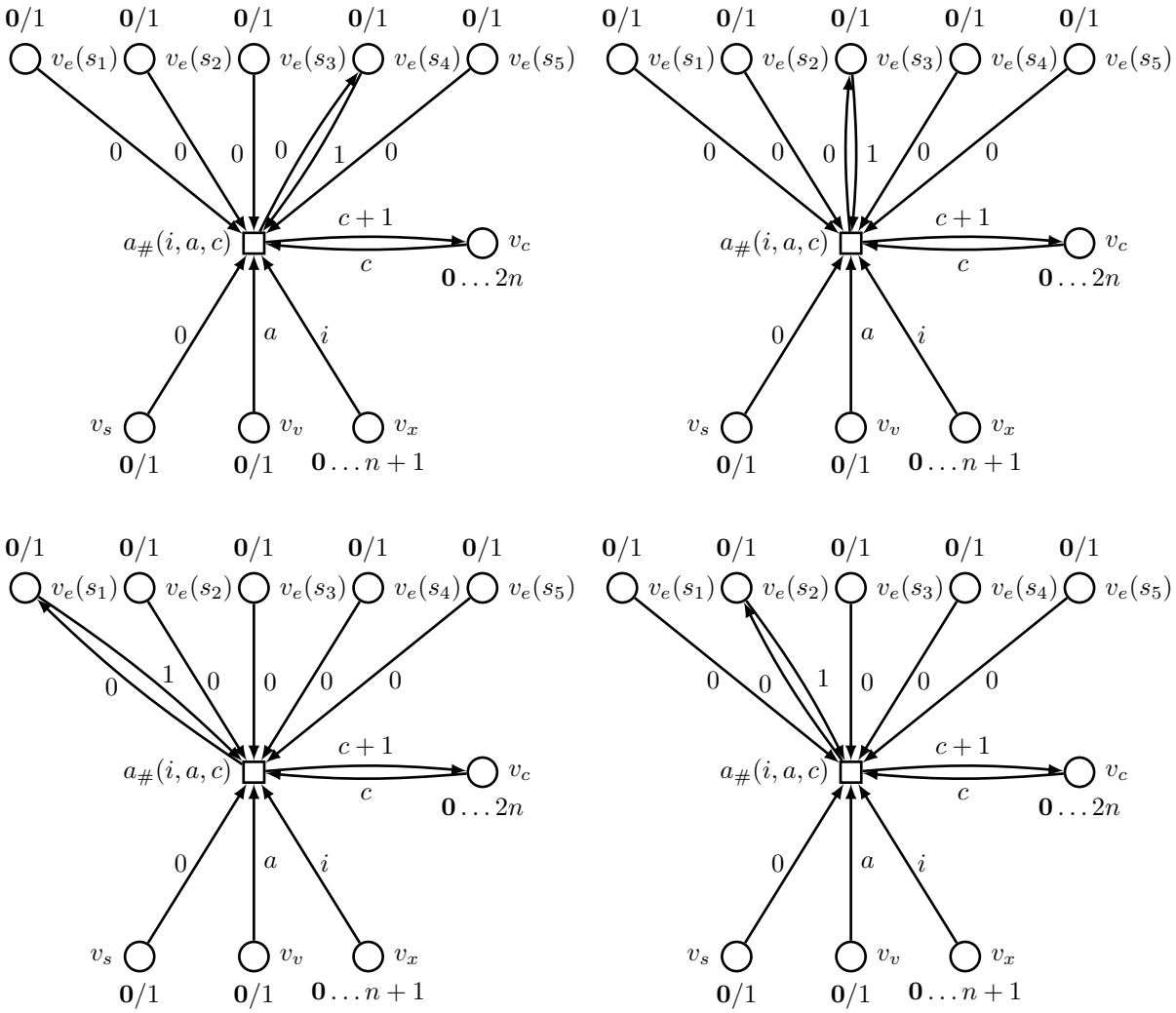


Figure 7: The action $a_\#(i,a,c)$ (from top left to bottom right): (a) $c = 2(i-1)$, (b) $c \neq 2(i-1)$ and c is odd, (c) $c \neq 2(i-1)$, c is even and $a = 0$, (d) $c \neq 2(i-1)$, c is even and $a = 1$.

- A variable v_x with values $\{0, \ldots, n+1\}$, which is used to indicate the variable x_i, which is currently processed. That is a value i between 1 and n for v_x means that the variable x_i is currently processed. The remaining values 0 and $n+1$ for v_x are used to indicate the start respectively end of the processing of the variables of Φ.
- A binary variable v_v, which holds the assignment of the current variable x_i.
- A variable v_c with values $\{0, 1, \ldots, 2n\}$, which is used (like a counter) to indicate the current position of the global sequence of actions for the current variable x_i.
- A binary variable $v_e(s_i)$ for every action $s_i \in S$. Informally, these variables are used to enforce the sequence of global actions representing the assignment guessed by the global component. This is achieved by (1) making the global actions the only actions that can set these variables to 1; in particular the variable $v_e(s_i)$ can only be set to 1 by the global action s_i and (2) by employing preconditions on these variables for actions in the global component that are responsible for increasing the value of the counter variable v_c.

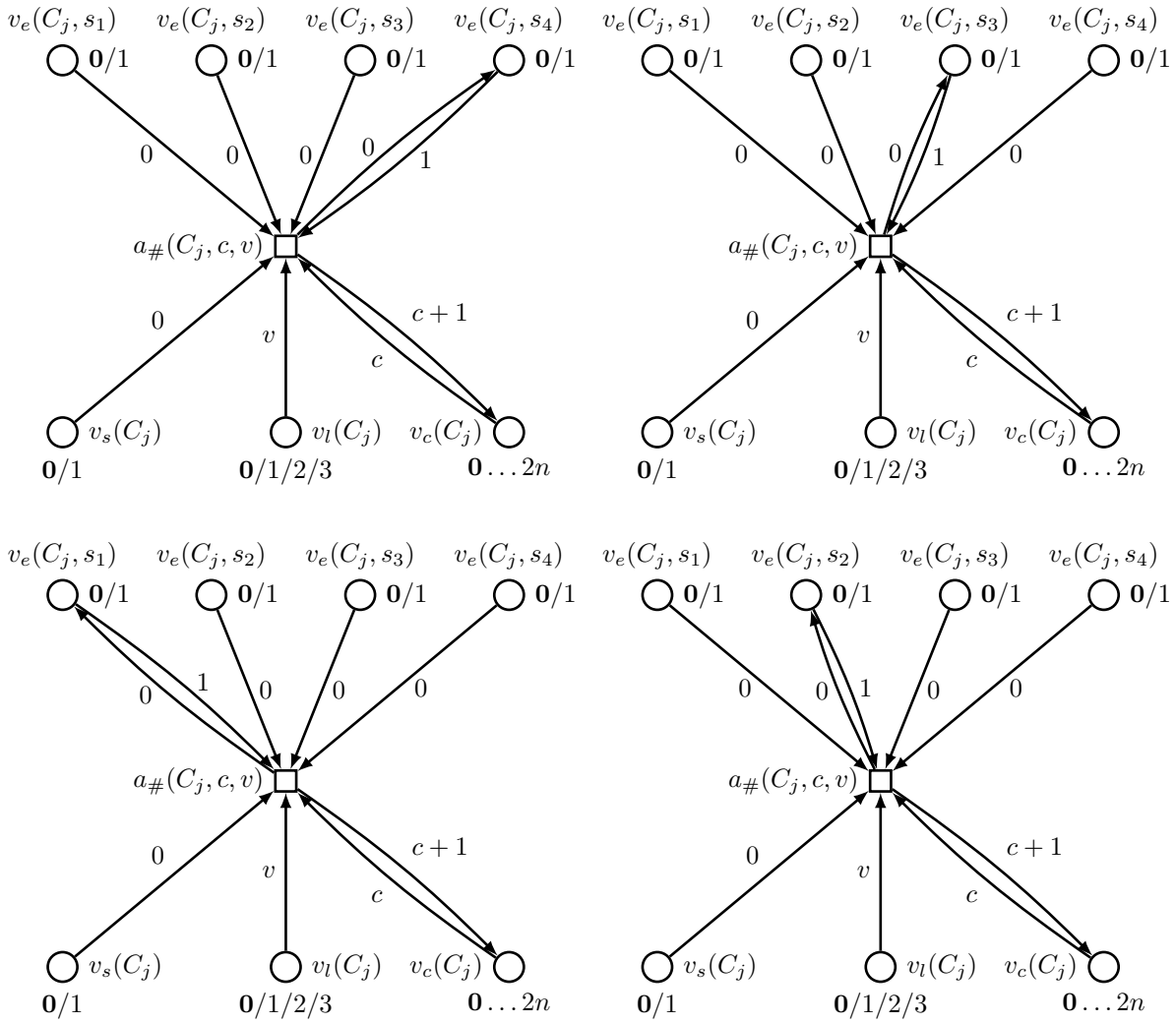


Figure 8: The action $a_\#(C_j, c, v)$ assuming that x_i is the v-th literal of C_j (from top left to bottom right): (a) $c = 2(i-1)$, (b) $c \neq 2(i-1)$ and c is odd, (c) $c \neq 2(i-1)$, c is even and x_i is positively in C_j, (d) $c \neq 2(i-1)$, c is even, and x_i is negatively in C_j.

We are now ready to define the actions within the global component.

- For every a with $a \in \{0, 1\}$, $\mathbb{P}$ contains an action $a_v(a)$ with $\mathsf{pre}(a_v(a))[v_s] = 1$, $\mathsf{eff}(a_v(a))[v_s] = 0$, and $\mathsf{eff}(a_v(a))[v_v] = a$. The actions $a_v(a)$ are illustrated in Figure 6.

  Informally, the purpose of these actions is to "guess" the assignment of the current variable x_i of Φ. Note that due to their precondition and effect on the variable v_s these actions can only be used once per variable x_i of Φ, which ensures that the assignment for the current variable can only be set once and remains constant during the execution of the global sequence of actions corresponding to the current variable.

- An action $a_x(0)$ with $\mathsf{pre}(a_x(0))[v_s] = 0$, $\mathsf{eff}(a_x(0))[v_s] = 1$, $\mathsf{pre}(a_x(0))[v_x] = 0$, $\mathsf{eff}(a_x(0))[v_x] = 1$, and $\mathsf{pre}(a_x(0))[v_e(s_r)] = 0$ for every r with $r \in \{1, 2, 3, 4, 5\}$.

  Informally, the action $a_x(0)$ is used to switch from the initial state to the first variable x_1. The purpose and idea behind $a_x(0)$ is very similar to the action $a_x(i)$ for $i > 0$ introduced below.

- For every i with $1 \leq i \leq n$, $\mathbb{P}$ contains an action $a_x(i)$ with $\mathsf{pre}(a_x(i))[v_s] = 0$, $\mathsf{eff}(a_x(i))[v_s] = 1$, $\mathsf{pre}(a_x(i))[v_x] = i$, $\mathsf{eff}(a_x(i))[v_x] = i + 1$, $\mathsf{pre}(a_x(i))[v_c] = 2n$, $\mathsf{eff}(a_x(i))[v_c] = 0$, $\mathsf{pre}(a_x(i))[v_e(s_5)] = 1$, $\mathsf{eff}(a_x(i))[v_e(s_5)] = 0$, and $\mathsf{pre}(a_x(i))[v_e(s_r)] = 0$ for every r with $r \in \{1, 2, 3, 4\}$. The actions $a_x(i)$ are illustrated in Figure 6.

  Informally, the actions $a_x(i)$ are used to switch from variable x_i to variable x_{i+1}. This is achieved by ensuring that:

  - the counter variable v_c has reached $2n$, i.e., the end of the sequence of global actions for x_i has been reached,
  - the counter variable v_c is set back to 0 in preparation of the global sequence of actions for x_{i+1},
  - the variable v_s is zero, which ensures that the assignment for x_i has been set previously,
  - the variable v_s is set to 1, which allows the actions $a_v(a)$ to set the assignment for x_{i+1},
  - the variable $v_e(s_5)$ is 1 (and the variables $v_e(s_1), \ldots, v_e(s_4)$ are zero), which ensures that the global action s_5 (and no other of the global actions) has been executed to terminate the sequence of global actions for x_i,
  - the variable $v_e(s_5)$ is reset to 0, which allows for a new sequence of global actions to start.

- For every i with $1 \leq i \leq n$, every $a \in \{0, 1\}$, and every c with $0 \leq c < 2n$, $\mathbb{P}$ contains an action $a_\#(i, a, c)$ with the following preconditions and effects. We set $\mathsf{pre}(a_\#(i, a, c))[v_s] = 0$, $\mathsf{pre}(a_\#(i, a, c))[v_v] = a$, $\mathsf{pre}(a_\#(i, a, c))[v_x] = i$, $\mathsf{pre}(a_\#(i, a, c))[v_c] = c$, $\mathsf{eff}(a_\#(i, a, c))[v_c] = c + 1$. Furthermore, we distinguish the following cases:

  - If $c = 2(i-1)$, we set $\mathsf{pre}(a_\#(i, a, c))[v_e(s_4)] = 1$, $\mathsf{eff}(a_\#(i, a, c))[v_e(s_4)] = 0$, and $\mathsf{pre}(a_\#(i, a, c))[v_e(s_r)] = 0$ for every r with $r \in \{1, 2, 3, 5\}$.
  - If $c \neq 2(i-1)$ and c is odd, we set $\mathsf{pre}(a_\#(i, a, c))[v_e(s_3)] = 1$, $\mathsf{eff}(a_\#(i, a, c))[v_e(s_3)] = 0$, and $\mathsf{pre}(a_\#(i, a, c))[v_e(s_r)] = 0$ for every r with $r \in \{1, 2, 4, 5\}$.
  - If $c \neq 2(i-1)$, c is even, and $a = 0$, we set $\mathsf{pre}(a_\#(i, a, c))[v_e(s_1)] = 1$, $\mathsf{eff}(a_\#(i, a, c))[v_e(s_1)] = 0$, and $\mathsf{pre}(a_\#(i, a, c))[v_e(s_r)] = 0$ for every r with $r \in \{2, 3, 4, 5\}$.
  - If $c \neq 2(i-1)$, c is even, and $a = 1$, we set $\mathsf{pre}(a_\#(i, a, c))[v_e(s_2)] = 1$, $\mathsf{eff}(a_\#(i, a, c))[v_e(s_2)] = 0$, and $\mathsf{pre}(a_\#(i, a, c))[v_e(s_r)] = 0$ for every r with $r \in \{1, 3, 4, 5\}$.

  The actions $a_\#(i, a, c)$ are illustrated in Figure 7.

  Informally, these actions are used to control the actions occurring in the sequence of global actions for the current variable x_i (where i is given by the current value of the variable v_x). Which sequence of global actions is enforced depends on the index i of the current variable x_i as well as the assignment of x_i given by the current value of the variable v_v. Note that each of these actions additionally checks that the variable v_s is set to 0, which ensures that the assignment for x_i has already been set (before the start of the sequence of global actions for x_i). Moreover, each such action uses the current value of the counter variable v_c (and increases the variable by one) to keep track of the current position in the sequence of global actions. Finally, each such actions enforces that a particular global action $s \in S$ has been executed before by checking that the variable $v_e(s)$ is 1 and prepares the execution of the next global action by resetting the value of $v_e(s)$ to 0. The choice of the global action that is enforced by such an action $a_\#(i, a, c)$ corresponds to the sequences given in Equations 1 and 2 and depends on the index i of the current variable x_i (given by the current value of the variable v_x), the assignment a of the current variable x_i (given by the current value of the variable v_v), and the current position c in the sequence of global actions for x_i (given by the current value of the variable v_c).

In summary the variables and actions of the global component ensure that every plan for $\mathbb{P}$ has to contain a sequence of global actions either corresponding to Equation 1 or to Equation 2 for every variable x_i of Φ.

We are now ready to describe the variables and actions of the local components. That is for every clause C_j, $\mathbb{P}$ contains the following variables:

- A binary variable $v_s(C_j)$ that ensures that the choice of the satisfying literal of C_j can only be made once.
- A variable $v_l(C_j)$ with values $\{0, 1, 2, 3\}$, which can only be set once in a plan and whose value indicates the choice of the literal of C_j that is satisfied.
- A variable $v_c(C_j)$ with values $\{0, 1, \ldots, 2n-1\}$, which is used like a counter to indicate the current position of the global sequence of actions.
- A binary variable $v_e(C_j, s_i)$ for every action $s_i \in S \setminus \{s_5\}$. Informally, these variables allow the local component to check whether a global action in $S \setminus \{s_5\}$ has recently been executed.

Moreover, $\mathbb{P}$ contains the following actions for every clause C_j:

- For every clause C_j and every v with $1 \leq v \leq 3$, $\mathbb{P}$ contains an action $a_l(C_j, v)$ with $\mathsf{pre}(a_l(C_j, v))[v_s(C_j)] = 1$, $\mathsf{eff}(a_l(C_j, v))[v_s(C_j)] = 0$, and $\mathsf{eff}(a_l(C_j, v))[v_l(C_j)] = v$. The actions $a_l(C_j, v)$ are illustrated in Figure 6.

  Informally, these actions allow the local component of C_j to chose, which literal is used to satisfy the clause. Because of their precondition and effect on $v_s(C_j)$ only one of these actions can be executed and moreover the action can only be executed once.

- For every clause C_j, every $0 \leq c < 2n$ and every v with $1 \leq v \leq 3$, $\mathbb{P}$ contains an action $a_\#(C_j, c, v)$. The preconditions and effects of $a_\#(C_j, c, v)$ are defined as follows. We set $\mathsf{pre}(a_\#(C_j, c, v))[v_s(C_j)] = 0$, $\mathsf{pre}(a_\#(C_j, c, v))[v_l(C_j)] = v$, $\mathsf{pre}(a_\#(C_j, c, v))[v_c(C_j)] = c$, $\mathsf{eff}(a_\#(C_j, c, v))[v_c(C_j)] = c + 1$. Furthermore, if the v-th literal of the clause C_j is a literal of the variable x_i for some i with $1 \leq i \leq n$, then

  - For every $c = 2(i-1)$, we set $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_4)] = 1$, $\mathsf{eff}(a_\#(C_j, c, v))[v_e(C_j, s_4)] = 0$, and $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_r)] = 0$ for every r with $r \in \{1, 2, 3\}$.
  - For every $c \neq 2(i-1)$ with c is odd, we set $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_3)] = 1$, $\mathsf{eff}(a_\#(C_j, c, v))[v_e(C_j, s_3)] = 0$, and $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_r)] = 0$ for every r with $r \in \{1, 2, 4\}$.
  - For every $c \neq 2(i-1)$ with c is even and x_i is positively in C_j, we set $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_1)] = 1$, $\mathsf{eff}(a_\#(C_j, c, v))[v_e(C_j, s_1)] = 0$, and $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_r)] = 0$ for every r with $r \in \{2, 3, 4\}$.
  - For every $c \neq 2(i-1)$ with c is even and x_i is negatively in C_j, we set $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_2)] = 1$, $\mathsf{eff}(a_\#(C_j, c, v))[v_e(C_j, s_2)] = 0$, and $\mathsf{pre}(a_\#(C_j, c, v))[v_e(C_j, s_r)] = 0$ for every r with $r \in \{1, 3, 4\}$.

  The actions $a_\#(C_j, c, v)$ are illustrated in Figure 8.

  Informally, these actions ensure that every plan for $\mathbb{P}$ has to contain a sequence of global actions that indicates the literal chosen to be the satisfying literal for C_j (which is determined by the value of the variable $v_l(C_j)$) is satisfied.

The effects of the actions $s_1, \ldots, s_5$ in S are defined as follows. For every $1 \leq i \leq 5$, we set $\mathsf{eff}(s_i)[v_e(s_i)] = 1$ and $\mathsf{eff}(s_i)[v_s] = 0$. For every i and j with $1 \leq i \leq 4$ and $1 \leq j \leq m$, we set $\mathsf{eff}(s_i)[v_e(C_j, s_i)] = 1$. Furthermore, for every i and j with $1 \leq i \leq 4$ and $1 \leq j \leq m$, we set $\mathsf{eff}(s_5)[v_e(C_j, s_i)] = 0$.

The initial state assigns 1 to the variables $v_s(C_j)$ for every j with $1 \leq j \leq m$ and 0 to every other variable. In the goal state we require $G[v_x] = n + 1$, $G[v_e(s_i)] = 0$ for every i with $1 \leq i \leq 5$, and for every j with $1 \leq j \leq m$ we require $v_c(C_j) = 2n$. This completes the construction of $\mathbb{P}$ and S. One can verify that:

- the instance $\langle \mathbb{P}, S \rangle$ can be constructed for any Φ in polynomial-time;
- the actions in S do not have preconditions;

- $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)) \leq 9$, i.e., the graph $\mathcal{G}_{\text{CAUSAL}}(\mathbb{P} \setminus S)$ has 1 component containing the 9 variables variables v_s, v_v, v_x, v_c, $v_e(s_1), \ldots, v_e(s_5)$ and m components containing the 7 variables $v_s(C_j)$, $v_l(C_j)$, $v_c(C_j)$, $v_e(C_j, s_1), \ldots, v_e(C_j, s_4)$.

It remains to show that the formula Φ is satisfiable if and only if the instance $\mathbb{P}$ has a plan.

Suppose that the formula Φ is satisfiable and let $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$ be a satisfying assignment of Φ. For a clause C_j with $1 \leq j \leq m$, let $\alpha(C_j)$ be the variable corresponding to a literal that is satisfied by α and let $\gamma(C_j)$ be the index of this literal, i.e., $\alpha(C_j)$ is the variable corresponding to the $\gamma(C_j)$-th literal of C_j. Then the concatenation of the following sequences is a plan for $\mathbb{P}$:

1. $\langle a_l(C_1, \gamma(C_1)), \ldots, a_l(C_m, \gamma(C_m)) \rangle$;
2. Let $\epsilon(i, c)$ be an arbitrary ordering of the set $\{\, a_c(C_j, c, \gamma(C_j)) \mid \alpha(C_j) = x_i \,\}$ for every i and c with $1 \leq i \leq n$ and $0 \leq c < 2n$ (i.e., the set of all actions required by clauses that are satisfied by x_i at the c-th step of the global sequence). Then for every i with $1 \leq i \leq n$, we append the sequence:

   $\langle a_x(i-1), a_v(\alpha(x_i))),$
   $s_{\alpha(x_i)+1}, a_\#(i, \alpha(x_i), 0), \epsilon(i, 0),\ s_3, a_\#(i, \alpha(x_i), 1), \epsilon(i, 1),$
   $\ldots$
   $s_{\alpha(x_i)+1}, a_\#(i, \alpha(x_i), 2(i-2)), \epsilon(i, 2(i-2)),\ s_3, a_\#(i, \alpha(x_i), 2(i-2)+1), \epsilon(i, 2(i-2)+1),$
   $s_4, a_\#(i, \alpha(x_i), 2(i-1)), \epsilon(i, 2(i-1)),\ s_3, a_\#(i, \alpha(x_i), 2(i-1)+1), \epsilon(i, 2(i-1)+1),$
   $s_{\alpha(x_i)+1}, a_\#(i, \alpha(x_i), 2i), \epsilon(i, 2i),\ s_3, a_\#(i, \alpha(x_i), 2i+1), \epsilon(i, 2i+1),$
   $\ldots$
   $s_{\alpha(x_i)+1}, a_\#(i, \alpha(x_i), 2(n-1)), \epsilon(i, 2(n-1)),\ s_3, a_\#(i, \alpha(x_i), 2(n-1)+1), \epsilon(i, 2(n-1)+1),$
   $s_5 \rangle$

   Note that the above sequence contains the required sequence of global actions given in Equation 1 and 2 for every variable x_i and additionally contains all required local actions.
3. $\langle a_x(n) \rangle$.

For the reverse direction suppose that ω is a plan for $\mathbb{P}$. W.l.o.g. we can assume that ω is inclusion minimal, i.e., deleting any action from ω results in a sequence of actions that is no longer a plan for $\mathbb{P}$. In particular, this implies that ω does not contain more than one consecutive occurrence of any action.

Claim 5. There is an assignment $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$ such that for every i with $1 \leq i \leq n$, the plan ω contains the following sequence as a (not necessarily consecutive) sub-sequence:

$$\underbrace{s_{\alpha(x_i)+1}, s_3, \ldots, s_{\alpha(x_i)+1}, s_3}_{(i-1)\text{-times}}, s_4, s_3, \underbrace{s_{\alpha(x_i)+1}, s_3, \ldots, s_{\alpha(x_i)+1}, s_3}_{(n-i)\text{-times}}, s_5$$

Furthermore, apart from the occurrences of the actions in S contained in one of the above sequences, the plan ω contains no other occurrences of the actions in S.

Because the initial state of v_x is 0 and the goal state of v_x is $n+1$, the plan ω has to contain a sequence of actions that set v_x from 0 to $n+1$. Now, by construction, the only sequence that can achieve this is $\langle a_x(0), \ldots, a_x(n) \rangle$ and furthermore ω must contain each of the actions of this sequence exactly once. Consequently, ω has to contain the sequence $\langle a_x(0), \ldots, a_x(n) \rangle$ as a (not necessarily consecutive) sub-sequence and ω contains no other occurrences of these actions.

Furthermore, for every i with $1 \leq i \leq n$, the action $a_x(i-1)$ sets the variable v_c to 0 and the action $a_x(i)$ expects the variable v_c to be $2n$. It follows that for every such i, the plan ω has to contain a sequence of actions in between the occurrences of $a_x(i-1)$ and $a_x(i)$ that increases v_c from 0 to $2n$. Again, by construction, the only sequences that can achieve this are of the form $\langle a_\#(i, a_0, 0), \ldots, a_\#(i, a_{2n-1}, 2n-1) \rangle$ where $a_r \in \{0, 1\}$ for every r with $0 \leq r < 2n$ and no other action of the form $a_\#(i', a', c')$, where $1 \leq i' \leq n$, $a \in \{0, 1\}$, and $0 \leq c' < 2n$, can be executed in between the occurrences of $a_x(i-1)$ and $a_x(i)$ in ω. We

continue by showing that $a_0 = a_1 = \cdots = a_{2n-1}$. Note that an action $a_\#(i, a, c)$ can only be executed if the variable v_v has the value a. Now the only actions that change v_v are the actions $a_v(a)$ where $a \in \{0, 1\}$. However, the action $a_v(a)$ can only be executed if the variable v_s is 1 and after $a_v(a)$ has been executed the variable v_s is 0 again. Moreover, the only actions that set v_s to 1 are the actions $a_x(p)$ where $0 \leq p \leq n$. Because each action $a_\#(i, a, c)$ can only be executed if v_s is 0 it follows that after the execution of $a_\#(i, a_0, 0)$ the variable v_v cannot be changed until the action $a_x(i)$ occurs. Consequently, $a_0 = a_1 = \cdots = a_{2n-1}$ and hence ω contains $\langle a_\#(i, a, 0), \ldots, a_\#(i, a, 2n-1)\rangle$ as a (not necessarily consecutive sub-sequence) in between the occurrences of $a_x(i-1)$ and $a_x(i)$ for some $a \in \{0, 1\}$ and no other action of the form $a_\#(i', a', c')$, where $1 \leq i' \leq n$, $a \in \{0, 1\}$, and $0 \leq c' < 2n$, can be executed in between the occurrences of $a_x(i-1)$ and $a_x(i)$ in ω. It now follows from the preconditions of the actions $a_\#(i, a, 0), \ldots, a_\#(i, a, 2n-1)$ that ω contains the sequence:

$$\langle \underbrace{s_{a+1}, s_3, \ldots, s_{a+1}, s_3}_{(i-1)\text{-times}}, s_4, s_3, \underbrace{s_{a+1}, s_3, \ldots, s_{a+1}, s_3}_{(n-i)\text{-times}} \rangle$$

as a (not necessarily consecutive) sub-sequence. This concludes the proof of the first part of the claim.

It follows from the proof of the first part that ω contains $\langle a_x(0), \ldots, a_x(n)\rangle$ as a (not necessarily consecutive) sub-sequence and furthermore that each of the actions in this sequence appears exactly once in ω. Moreover, we have obtained from the first part of the proof that for every $1 \leq i \leq n$, the plan ω contains $\langle a_\#(i, a, 0), \ldots, a_\#(i, a, 2n-1)\rangle$ as a (not necessarily consecutive sub-sequence) in between the occurrences of $a_x(i-1)$ and $a_x(i)$ for some $a \in \{0, 1\}$ and that no other action of the form $a_\#(i', a', c')$, where $1 \leq i' \leq n$, $a \in \{0, 1\}$, and $0 \leq c' < 2n$, can be executed in between the occurrences of $a_x(i-1)$ and $a_x(i)$ in ω. Because all actions of the form $a_\#(i', a', c')$, where $1 \leq i' \leq n$, $a \in \{0, 1\}$, and $0 \leq c' < 2n$ require v_x to be greater than 0 and less than $n+1$, it also follows that these actions do neither appear before the occurrence of $a_x(0)$ nor after the occurrence of $a_x(n)$ in ω. On the other hand it follows from the construction that in between two occurrences of distinct actions in S, the plan ω must contain an occurrence of an action in $\{\, a_x(i) \mid 1 \leq i \leq n \,\} \cup \{\, a_\#(i, a, c) \mid 1 \leq i \leq n$ and $a \in \{0, 1\}$ and $0 \leq c < 2n \,\}$. Because otherwise two distinct variables in $\{\, v_e(i) \mid 1 \leq i \leq 5 \,\}$ would be set to 1 and there is no sequence of actions that sets all of these variables to 0 again, as is required by the goal state. Moreover, because of the preconditions of $a_x(0)$ and the fact that the variables in $\{\, v_e(i) \mid 1 \leq i \leq 5 \,\}$ must be 0 in the goal state, it follows that ω cannot contain any additional occurrences of the actions in S. This concludes the proof of the claim.

It now remains to show that the assignment α implied by the above claim is a satisfying assignment for Φ. We need the following intermediate claim.

Claim 6. For every C_j with $1 \leq j \leq m$, there is a $v \in \{1, 2, 3\}$ such that the plan ω contains the following sequence as a (not necessarily consecutive) sub-sequence:

$$\underbrace{s_{b(C_j,v)}, s_3, \ldots, s_{b(C_j,v)}, s_3}_{(i-1)\text{-times}}, s_4, s_3, \underbrace{s_{b(C_j,v)}, s_3, \ldots, s_{b(C_j,v)}, s_3}_{(n-i)\text{-times}}, s_5$$

where $b(C_j, v) = 1$ if the v-th literal of C_j is negative and $b(C_j, v) = 2$ otherwise.

Because the initial state of $v_c(C_j)$ is 0 and its goal state is $2n$, the plan ω has to contain a sequence of actions that increase $v_c(C_j)$ from 0 to $2n$. By construction, this can be achieved only by sequences of the form $\langle a_\#(C_j, 0, v_0), \ldots, a_\#(C_j, 2n-1, v_{2n-1})\rangle$ and consequently ω has to contain such a sequence. We first show that $v_0 = v_1 = \cdots = v_{2n-1}$. Note that every action $a_\#(C_j, c, v)$ with $0 \leq c < 2n$ and $1 \leq v \leq 3$ requires that the variable $v_v(C_j)$ is v before it can be executed. Moreover, the only actions that can change the value of $v_v(C_j)$ are the actions $a_l(C_j, v)$ for every $1 \leq v \leq 3$. However, these actions can only be executed if the variable v_s is 1 and there is no action that sets v_s to 1. Because the actions in $\{\, a_\#(C_j, c, v) \mid 0 \leq c < 2n$ and $1 \leq v \leq 3 \,\}$ require v_s to be 0 it follows that the value of v_v can only be changed before the first occurrence of one of these actions. Consequently, $v_0 = v_1 = \cdots = v_{2n-1}$ and ω has to contain $\langle a_\#(C_j, 0, v), \ldots, a_\#(C_j, 2n-1, v)\rangle$ for some $v \in \{1, 2, 3\}$ as a (not necessarily consecutive) sub-sequence. The claim now follows because of the preconditions of these actions.

Putting the previous two claims together we obtain that for every C_j there is a literal that is satisfied by the assignment α. Hence, Φ is satisfiable, as required. □

4. Kernelization – Limits of Polynomial Preprocessing

In this section we explore the limits of polynomial preprocessing for c-Causal Evaluation[Actions] with bounded domain as well as for c-Bounded Causal Evaluation[Actions] – recall that these are the variants of the evaluation problem which were identified to be fixed-parameter tractable in the previous section. In what follows, we will show our kernelization lower bounds via PPT-reductions. To prove that neither c-Causal Evaluation[Actions] with bounded domain nor c-Bounded Causal Evaluation[Actions] admits a polynomial kernel (unless coNP $\subseteq$ NP/poly) we built upon the following auxiliary Lemma[1]. Here we will reduce from Small Universe Hitting Set, which does not admit a polynomial kernel (unless coNP $\subseteq$ NP/poly) [19, Theorem 5].

Small Universe Hitting Set	
Instance:	A finite set E, a collection S of subsets of E, and an integer k.
Parameter:	$\lvert E\rvert$
Question:	Does S have a *hitting set* of cardinality at most k, i.e., is there a set $H \subseteq E$ with $\lvert H\rvert \leq k$ and $H \cap s \neq \emptyset$ for every $s \in S$?

Lemma 16. SAS$^+$ Planning *planning with binary domain parameterized by the number of actions does not admit a polynomial kernel unless* coNP $\subseteq$ NP/poly.

Proof. We show this result by giving a PPT-reduction from Small Universe Hitting Set to SAS$^+$ Planning parameterized by the number of actions. Let $I = (E, S, k)$ be an instance of Small Universe Hitting Set. Note that $k \leq |E|$, otherwise I is a trivial Yes-instance. We construct an instance $\mathbb{P} = \langle V, D, A, I, G\rangle$ of SAS$^+$ Planning in polynomial-time such that I has a hitting set of size at most k if and only if $\mathbb{P}$ has a plan, $|A| \leq |E| + k \leq 2|E|$, and $D = \{0, 1\}$, which will show the lemma. The set V contains one binary variable v_s for every $s \in S$, one binary variable c_i for every i with $1 \leq i \leq k$, and one binary variable c. The initial state I requires all variables to be 0 and the goal state G is only defined on the variables in $\{v_s \mid s \in S\}$ requiring those variables to be 1. Finally, the set A contains:

- one action a_e for every $e \in E$, whose precondition requires the variable c to be 1 and whose effect sets c to 0 and for all $s \in S$ sets the variable v_s to 1 for which $e \in s$,
- one action a_i for every i with $1 \leq i \leq k$, whose precondition requires the variable c_i to be 0 and whose effect sets both c and c_i to 1.

This completes the description of $\mathbb{P}$. Note that $\mathbb{P}$ can be constructed in polynomial-time form I, $|A| \leq |E| + k \leq 2|E|$, and $D = \{0, 1\}$, as required. It remains to show that I has a hitting set if and only if $\mathbb{P}$ has a plan.

Towards showing the forward direction, let $H = \{h_1, \ldots, h_l\}$ be a hitting set for S of size at most k. It is straightforward to verify that $\langle a_1, a_{h_1}, \ldots, a_l, a_{h_l}\rangle$ is a plan for $\mathbb{P}$.

Towards showing the reverse direction, let ω be a plan for $\mathbb{P}$ and let H be the subset of E containing all elements for which there is an action a_e in ω. We claim that H is a hitting set for I of size at most k. Note that H is a hitting set for S because the actions in H are the only actions in ω that set the variables in $\{ v_s \mid s \in S \}$ to 1. Because any action a_e executed by ω sets the variable c to 0 but requires c to be 1 in order to be executed, we obtain that c has to be set to 1 before any action a_e can be executed in ω. Moreover, since the only actions in A that can set c to 1 are the actions in $\{ a_i \mid 1 \leq i \leq k \}$ and each of these actions a_i can be executed at most once in ω (because a_i sets the variable c_i to 1 but requires it to be 0), we obtain that H contains at most k elements. □

We are now ready to show kernelization lower bounds for all variants of the evaluation problem that we previously identified to allow for fixed-parameter tractability.

[1]We remark that this result has also been obtained by using a different reduction in the manuscript of the journal version of [57] (cf. `http://www.dbai.tuwien.ac.at/staff/pfandler/`).

Theorem 17. *Neither* c-Causal Evaluation[Actions] *with bounded domain nor* c-Bounded Causal Evaluation[Actions] *admits a polynomial kernel unless* coNP $\subseteq$ NP/poly *for any* $c \geq 1$.

Proof. Because of Lemma 16, it suffices to show that there is a PPT-reduction from SAS$^+$ Planning with binary domain to c-Causal Evaluation[Actions] with bounded domain. Let $\mathbb{P} = \langle V, D, A, I, G \rangle$ be an instance of SAS$^+$ Planning with binary domain. Because $\mathsf{cc\text{-}size}(\mathcal{G}_{\text{Causal}}(\mathbb{P} \setminus A)) \leq 1 \leq c$, we obtain that $(\mathbb{P}, A)$ is an instance of c-Causal Evaluation[Actions] satisfying the requirements of a polynomial parameter reduction. Towards showing the theorem for c-Bounded Causal Evaluation[Actions] let $\mathbb{P} = \langle V, D, A, I, G \rangle$ be the instance of SAS$^+$ Planning with binary domain constructed in the proof of Lemma 16 from an instance (E, S, k) of Small Universe Hitting Set. Recall from the proof of Lemma 16 that $\mathbb{P}$ has a plan if and only if $\mathbb{P}$ has a plan of length at most $2|E|$. It follows that $(\mathbb{P}, A, 2|E|)$ is an instance of c-Bounded Causal Evaluation[Actions] that has a solution if and only if (E, S, k) has a solution, showing that Small Universe Hitting Set has a PPT-reduction to c-Bounded Causal Evaluation[Actions]. □

5. Discussion

In this section we comment briefly on the potential impact to practical applications and the choice of the tractable base class. There has recently been a growing interest in finding tractable fragments of planning using the causal graph [1, 5, 10, 11, 15, 40, 50–52]). As pointed out in [52]:

> "Such results are not purely of theoretical interest, as the causal graph is used in a variety of practical applications from problem decomposition [11] to the derivation of non-admissible domain-independent heuristics for planning [46]."

We believe that this applies even more to (fixed-parameter) tractable extensions of these fragments obtained by the backdoor approach. In particular, the backdoor approach adds a novel dimension of flexibility to the definition of these tractable classes that allows one to trade efficiency for generality to best suit the particular application. This also allows to use the insights obtained for those tractable classes to solve arbitrary planning instances.

Regarding the choice of the tractable base class, we want to point out that even though the class of instances with bounded component size of the causal graph itself can be seen as rather trivial, the backdoor approach extends its applicability to arbitrary planning instances. Moreover, as shown by [15] this class of planning instances is in a sense as general as possible at least w.r.t. classes of planning instances that are obtained purely via restrictions on the causal graph. Namely, let $\mathcal{P}$ be the class of all planning instances, whose causal graph is a member of a given class $\mathcal{C}$ of directed graphs. Then (unless W[1] $\subseteq$ nu-FPT – recall that nu-FPT denotes the non-uniform version of FPT) $\mathcal{P}$ can be solved in polynomial time if and only if there is a constant c such that the maximum size of any (weakly connected) component of every graph in $\mathcal{C}$ is at most c. Clearly, their result does not consider classes that are obtained via other types of restrictions or combinations of such restrictions with restrictions on the causal graph and it hence remains interesting to consider further (polynomial time) tractable fragments of planning.

We note that our work is related to factored planning [12, 54, 60, 67] in the sense that both approaches are concerned with instances that can be naturally decomposed into almost independent subinstances; usually witnessed by a partition of the variables. However, in contrast to our approach, factored planning focuses on obtaining approximate (or heuristic) solutions. Recently factored planning has inspired the development of a planner that is tailored towards solving instances with a star-like topology [43], i.e., the causal graph of a star-like planning instance can be partitioned into one center component and several leaf components satisfying certain properties. Interestingly the center component (of a star-like topology) matches exactly our notion of a variable-deletion-backdoor set. Their planner, which has been shown to beat state-of-the-art planners on several planning instances, crucially exploits the observation that the leaf components are independent given a current plan for the center component. In this way we are hopeful that also our action-deletion-backdoor approach can inspire similar tailored-made planners.

6. Conclusion

In this work we have introduced the first two types of backdoor sets for planning. The distance to the tractable fragment was expressed by the number of variables or actions that need to be removed in order to obtain a causal graph of bounded maximum component size. For each backdoor type and each setting of considered SAS^+ planning formalisms (with bounded/unbounded plan length and bounded/unbounded domain of the variables) we have analyzed the (parameterized) complexity of the detection and evaluation problem. In three cases we have obtained the most desirable result where detection as well as evaluation are fixed-parameter-tractable, namely in all settings under action-deletion-backdoors with the exception of unbounded planning with unbounded domain of the variables. These results include the first fpt-algorithm for planning with unbounded plan length that neither limits the number of variables nor the number of actions in the planning instance. In the remaining cases, we have ruled out the existence of an fpt-algorithm by showing hardness for W[1] or paraNP. For the cases, where fixed-parameter-tractability was achieved, we have investigated the potential and limits of polynomial preprocessing, i.e., whether this setting also admits a polynomial kernel. Such a polynomial kernel would be a very desirable result from the algorithmic perspective. We were able to show that polynomial kernels exist for the detection problems, whereas we ruled out the existence of polynomial kernels for the evaluation problems (under the usual complexity theoretic assumptions).

We envisage the study of other underlying graph structures (such as the variable-action graph) to obtain further useful notions of backdoor sets. Furthermore, we want to explore additional supporting parameters that help to make the evaluation problem in the variable-deletion setting fixed-parameter tractable. We hope that the methodology used in this work will inspire other researchers as it is versatile and applicable to a variety of problems in AI and beyond.

Acknowledgments

Sebastian Ordyniak acknowledges support from the Employment of Newly Graduated Doctors of Science for Scientific Excellence (CZ.1.07/2.3.00/30.0009). Martin Kronegger and Andreas Pfandler were supported by the Austrian Science Fund (FWF): P25518-N23. In addition, Martin Kronegger was also supported by the Austrian Science Fund (FWF) under grants S11408-N23 and Y698. Andreas Pfandler was also supported by the German Research Foundation (DFG): ER 738/2-2.

References

[1] Aghighi, M., Jonsson, P., Ståhlberg, S., 2015. Tractable cost-optimal planning over restricted polytree causal graphs. In: Bonet, B., Koenig, S. (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. AAAI Press, pp. 3225–3231.

[2] Atserias, A., Oliva, S., 2014. Bounded-width QBF is pspace-complete. J. Comput. Syst. Sci. 80 (7), 1415–1429.

[3] Bäckström, C., 2014. Parameterising the complexity of planning by the number of paths in the domain-transition graphs. In: Proc. ECAI 2014. Vol. 263 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 33–38.

[4] Bäckström, C., 2015. Some fixed parameter tractability results for planning with non-acyclic domain-transition graphs. In: Proc. AAAI 2015. AAAI Press, pp. 3232–3238.

[5] Bäckström, C., 2015. Some fixed parameter tractability results for planning with non-acyclic domain-transition graphs. In: Bonet, B., Koenig, S. (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. AAAI Press, pp. 3232–3238.

[6] Bäckström, C., Jonsson, P., 2013. A refined view of causal graphs and component sizes: SP-closed graph classes and beyond. J. Artif. Intell. Res. 47, 575–611.

[7] Bäckström, C., Jonsson, P., Ordyniak, S., Szeider, S., 2015. A complete parameterized complexity analysis of bounded planning. J. Comput. Syst. Sci. 81 (7), 1311–1332.

[8] Bäckström, C., Nebel, B., 1995. Complexity results for SAS^+ planning. Comput. Intell. 11, 625–656.

[9] Bliem, B., Bredereck, R., Niedermeier, R., 2016. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In: Proc. IJCAI 2016. IJCAI/AAAI Press, pp. 102–108.

[10] Brafman, R. I., Domshlak, C., 2003. Structure and complexity in planning with unary operators. J. Artif. Intell. Res. 18, 315–349.

[11] Brafman, R. I., Domshlak, C., 2006. Factored planning: How, when, and when not. In: Proc. AAAI 2006. AAAI Press, pp. 809–814.

[12] Brafman, R. I., Domshlak, C., 2013. On the complexity of planning for agent teams and its implications for single agent planning. Artif. Intell. 198, 52–71.
[13] Bredereck, R., Faliszewski, P., Niedermeier, R., Skowron, P., Talmon, N., 2015. Elections with few candidates: Prices, weights, and covering problems. In: Proc. ADT 2015. Vol. 9346 of LNCS. Springer, pp. 414–431.
[14] Bylander, T., 1994. The computational complexity of propositional STRIPS planning. Artif. Intell. 69 (1–2), 165–204.
[15] Chen, H., Giménez, O., 2010. Causal graphs and structurally restricted planning. J. Comput. Syst. Sci. 76 (7), 579–592.
[16] Crama, Y., Ekin, O., Hammer, P. L., 1997. Variable and term removal from Boolean formulae. Discrete Applied Mathematics 75 (3), 217–230.
[17] Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S., 2015. Parameterized Algorithms. Springer.
[18] Diestel, R., 2000. Graph Theory, 2nd Edition. Vol. 173 of Graduate Texts in Mathematics. Springer, New York.
[19] Dom, M., Lokshtanov, D., Saurabh, S., 2009. Incompressibility through colors and ids. In: Proc. ICALP 2009. Vol. 5555 of LNCS. Springer, pp. 378–389.
[20] Domshlak, C., Dinitz, Y., 2001. Multi-entity off-line coordination: Structure and complexity. Tech. Rep. CS-01-04, Ben-Gurion University of Negev, Israel.
[21] Downey, R. G., Fellows, M. R., 1999. Parameterized Complexity. Springer.
[22] Downey, R. G., Fellows, M. R., 2013. Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer.
[23] Downey, R. G., Fellows, M. R., Stege, U., 1999. Parameterized complexity: A framework for systematically confronting computational intractability. In: Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future. Vol. 49 of DIMACS Series in Disc. Math. Theor. Comput. Sci. DIMACS, pp. 49–99.
[24] Dvorák, W., Ordyniak, S., Szeider, S., 2012. Augmenting tractable fragments of abstract argumentation. Artif. Intell. 186, 157–173.
[25] Eiben, E., Ganian, R., Ordyniak, S., 2016. Using decomposition-parameters for QBF: mind the prefix! In: Proc. AAAI 2016. AAAI Press, pp. 964–970.
[26] Erdélyi, G., Fellows, M. R., Rothe, J., Schend, L., 2015. Control complexity in bucklin and fallback voting: A theoretical analysis. J. Comput. Syst. Sci. 81 (4), 632–660.
[27] Fichte, J. K., Szeider, S., 2011. Backdoors to tractable answer-set programming. In: Proc. IJCAI 2011. pp. 863–868.
[28] Fichte, J. K., Szeider, S., 2011. Backdoors to tractable answer-set programming. CoRR abs/1104.2788.
[29] Fichte, J. K., Szeider, S., 2013. Backdoors to normality for disjunctive logic programs. In: Proc. AAAI 2013. AAAI Press, pp. 320–327.
[30] Flum, J., Grohe, M., 2003. Describing parameterized complexity classes. Inf. Comput. 187 (2), 291–319.
[31] Flum, J., Grohe, M., 2006. Parameterized Complexity Theory. Springer.
[32] Foulser, D. E., Li, M., Yang, Q., 1992. Theory and algorithms for plan merging. Artif. Intell. 57 (2-3), 143–181.
[33] Ganian, R., Ordyniak, S., 2016. The complexity landscape of decompositional parameters for ILP. In: Proc. AAAI 2016. AAAI Press, pp. 710–716.
[34] Ganian, R., Ramanujan, M. S., Szeider, S., 2016. Discovering archipelagos of tractability for constraint satisfaction and counting. In: Proc. ACM-SIAM, SODA 2016. SIAM, pp. 1670–1681.
[35] Garey, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.
[36] Gaspers, S., Misra, N., Ordyniak, S., Szeider, S., Zivny, S., 2017. Backdoors into heterogeneous classes of SAT and CSP. J. Comput. Syst. Sci. 85, 38–56.
[37] Gaspers, S., Ordyniak, S., Ramanujan, M. S., Saurabh, S., Szeider, S., 2016. Backdoors to q-horn. Algorithmica 74 (1), 540–557.
[38] Gaspers, S., Szeider, S., 2012. Backdoors to acyclic SAT. In: Proc. ICALP 2012. Vol. 7391 of LNCS. Springer, pp. 363–374.
[39] Gaspers, S., Szeider, S., 2012. Backdoors to satisfaction. In: Bodlaender, H. L., Downey, R., Fomin, F. V., Marx, D. (Eds.), The Multivariate Algorithmic Revolution and Beyond. Vol. 7370 of LNCS. Springer, pp. 287–317.
[40] Giménez, O., Jonsson, A., 2008. The complexity of planning problems with simple causal graphs. J. Artif. Intell. Res. 31, 319–351.
[41] Giménez, O., Jonsson, A., 2009. Planning over chain causal graphs for variables with domains of size 5 is NP-hard. J. Artif. Intell. Res. 34, 675–706.
[42] Giménez, O., Jonsson, A., 2012. The influence of k-dependence on the complexity of planning. Artif. Intell. 177-179, 25–45.
[43] Gnad, D., Hoffmann, J., 2018. Star-topology decoupled state space search. Artificial Intelligence 257, 24 – 60. URL http://www.sciencedirect.com/science/article/pii/S000437021730173X
[44] Gottlob, G., Szeider, S., 2008. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. Comput. J. 51 (3), 303–325.
[45] Gregory, P., Fox, M., Long, D., 2008. A new empirical study of weak backdoors. In: Proc. CP 2008. Vol. 5202 of LNCS. Springer, pp. 618–623.
[46] Helmert, M., 2004. A planning heuristic based on causal graph analysis. In: Proc. ICAPS 2004. AAAI, pp. 161–170.
[47] Jiang, T., Li, M., 1995. On the approximation of shortest common supersequences and longest common subsequences. SIAM J. Comput. 24 (5), 1122–1139.
[48] Jonsson, A., Jonsson, P., Lööw, T., 2014. Limitations of acyclic causal graphs for planning. Artif. Intell. 210, 36–55.
[49] Jonsson, P., Bäckström, C., 1998. State-variable planning under structural restrictions: Algorithms and complexity. Artif. Intell. 100 (1-2), 125–176.
[50] Katz, M., Domshlak, C., 2008. New islands of tractability of cost-optimal planning. J. Artif. Intell. Res. 32, 203–288.
[51] Katz, M., Domshlak, C., 2010. Optimal admissible composition of abstraction heuristics. Artif. Intell. 174 (12–13), 767–798.

[52] Katz, M., Keyder, E., 2012. Structural patterns beyond forks: Extending the complexity boundaries of classical planning. In: Proc. AAAI 2012. AAAI Press, pp. 1779–1785.
[53] Kilby, P., Slaney, J. K., Thiébaux, S., Walsh, T., 2005. Backbones and backdoors in satisfiability. In: Proc. AAAI 2005. AAAI Press / The MIT Press, pp. 1368–1373.
[54] Knoblock, C. A., 1994. Automatically generating abstractions for planning. Artif. Intell. 68 (2), 243–302.
[55] Kronegger, M., Ordyniak, S., Pfandler, A., 2014. Backdoors to planning. In: Proc. AAAI 2014. AAAI Press, pp. 2300–2307.
[56] Kronegger, M., Ordyniak, S., Pfandler, A., 2015. Variable-deletion backdoors to planning. In: Proc. AAAI 2015. AAAI Press, pp. 3305–3312.
[57] Kronegger, M., Pfandler, A., Pichler, R., 2013. Parameterized complexity of optimal planning: A detailed map. In: Proc. IJCAI 2013. AAAI Press, pp. 954–961.
[58] Niedermeier, R., 2006. Invitation to Fixed-Parameter Algorithms. Oxford University Press.
[59] Nishimura, N., Ragde, P., Szeider, S., 2004. Detecting backdoor sets with respect to Horn and binary clauses. In: Proc. SAT 2004. pp. 96–103.
[60] Nissim, R., Brafman, R. I., 2014. Distributed heuristic forward search for multi-agent planning. J. Artif. Intell. Res. 51, 293–332.
[61] Ordyniak, S., Paulusma, D., Szeider, S., 2013. Satisfiability of acyclic and almost acyclic CNF formulas. Theor. Comput. Sci. 481, 85–99.
[62] Papadimitriou, C. H., 1994. Computational complexity. Addison-Wesley.
[63] Pfandler, A., Rümmele, S., Szeider, S., 2013. Backdoors to abduction. In: Proc. IJCAI 2013. AAAI Press, pp. 1046–1052.
[64] Pfandler, A., Rümmele, S., Wallner, J. P., Woltran, S., 2015. On the parameterized complexity of belief revision. In: Proc. IJCAI 2015. AAAI Press, pp. 3149–3155.
[65] Pietrzak, K., 2003. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. J. Comput. Syst. Sci. 67 (4), 757–771.
[66] Russell, S. J., Norvig, P., 2010. Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education.
[67] Sacerdoti, E. D., 1974. Planning in a hierarchy of abstraction spaces. Artif. Intell. 5 (2), 115–135.
[68] Samer, M., Szeider, S., 2009. Backdoor sets of quantified Boolean formulas. J. Autom. Reasoning 42 (1), 77–97.
[69] Williams, R., Gomes, C., Selman, B., 2003. Backdoors to typical case complexity. In: Proc. IJCAI 2003. Morgan Kaufmann, pp. 1173–1178.