Some Fixed Parameter Tractability Results for Planning with Non-Acyclic Domain-Transition Graphs

Christer Bäckström
Linköping University, SE-58183 Linköping, Sweden.
Email: christer.backstrom@liu.se

Abstract

Bäckström studied the parameterised complexity of planning when the domain-transition graphs (DTGs) are acyclic. He used the parameters d (domain size), k (number of paths in the DTGs) and w (treewidth of the causal graph), and showed that planning is fixed-parameter tractable (fpt) in these parameters, and fpt in only parameter k if the causal graph is a polytree. We continue this work by considering some additional cases of non-acyclic DTGs. In particular, we consider the case where each strongly connected component (SCC) in a DTG must be a simple cycle, and we show that planning is fpt for this case if the causal graph is a polytree. This is done by first preprocessing the instance to construct an equivalent abstraction and then apply Bäckströms technique to this abstraction. We use the parameters d and k, reinterpreting this as the number of paths in the condensation of a DTG, and the two new parameters c (the number of contracted cycles along a path) and p_{max} (an upper bound for walking around cycles, when not unbounded).

1 Introduction

Identifying tractable classes of planning problems is important for several reasons. From a theoretical perspective it helps us to understand the factors influencing how difficult it is to plan. From a practical perspective, there are applications that can be modelled in currently known tractable fragments (cf. Cooper, Maris, and Régnier (2014)). Even more importantly, tractable fragments have provided the basis for many successful planning heuristics (cf. Helmert (2004), Katz and Domshlak (2008), Katz and Domshlak (2010)).

Early work on planning complexity studied mainly syntactical restrictions (Bylander 1994; Bäckström and Nebel 1995). One of the two main avenues for non-syntactical approaches is to study restrictions on the domain-transition graphs (DTGs) for the variables (Jonsson and Bäckström 1998a; 1998b). The other one is to study restrictions on the causal graph. For instance, planning is known to be **NP**-complete for forks and inverted forks (Domshlak and Dinitz 2001), chains (Giménez and Jonsson 2009), polytrees (Giménez and Jonsson 2008) as well as fences and polypaths (Bäckström and Jonsson 2013). Also many tractable cases exist by imposing further restrictions. Recently, Bäckström (2014) combined these two approaches by studying combined restrictions on both the DTGs and the causal graph under parameterised complexity analysis.

While standard complexity only considers the size of the instance, parameterised complexity is multi-dimensional in the sense that it allows one or more parameters in addition to the instance size, treating these parameters as independent. In standard complexity, a problem is considered tractable if it can be solved in time $O(n^c)$, for some constant c, where n is the size of the instance. For many problems, like the **NP**-complete problems, we do not know of any better way to solve them than combinatorial search. Somewhat simplified, we do not know if the problem can be solved faster than time $O(2^n)$. However, in practice the time is often not exponential in the size of the instance, but in some smaller part of it. Sometimes we can characterize this smaller part by some parameter k that does not directly depend on n, and it may be that the required time is only exponential in k, not in n. Parameterised complexity uses the more relaxed concept of fixed-parameter tractability (fpt). For example, if a problem can be solved in time $O(2^k n^c)$, where the parameter k is independent of n, then the problem is fpt in k. This is intended to have more practical relevance since many fpt problems can be considered as tractable in practice.

Parameterised complexity is a widely used technique in many areas of computer science, including many subareas of AI, like non-monotonic reasoning, constraints, social choice and argumentation. There is also a growing body of parameterised results for planning. For instance, some early results for STRIPS planning (Downey, Fellows, and Stege 1999) were followed by parameterised analysis of previously studied classes using plan length as parameter (Bäckström et al. 2012) as well as using other parameters (Kronegger, Pfandler, and Pichler 2013). Recent studies also include plan reuse (de Haan, Roubícková, and Szeider 2013) and backdoors (Kronegger, Ordyniak, and Pfandler 2014).

Bäckströms (2014) study assumed that all domain-transition graphs (DTGs) are acyclic. He considered the parameters domain size (d), number of paths in the DTGs (k) and the treewidth of the causal graph (w). He showed that planning with acyclic DTGs is fpt in these three parameters and that it is fpt in only k if also the causal graph is a polytree. Although acyclic DTGs may appear very limited, it was shown that the results could be used to prove that computing the delete relaxation heuristic, h^+, (Hoffmann 2005) is fpt, while it is **NP**-complete under standard analysis.

In this paper, we consider some classes of non-acyclic DTGs with increasing generality. The most general one (cycle-DAGs) allows arbitrary DTGs where every strongly connected component (SCC) is a simple cycle. We prove that planning when the causal graph is a polytree and all DTGs are cycle-DAGs is fpt in parameters c, d, k and p_{max}. Parameter d is the domain size, while k is the number of paths in the condensation of a DTG and c is the number of contracted cycles along the path. We prove that there is a gap in the sense that we can either go around a cycle any number of times or there is a specific upper bound that can be determined in advance. Parameter p_{max} is the maximum of these bounds for all cycles that are not unbounded.

Bäckströms (2014) results exploit that the number of paths in a DTG is bounded by k to build a corresponding CSP instance with variable domains that are bounded by the parameters, and then prove that solving this CSP instance is equivalent to solving the planning instance. We exploit the gap mentioned above to define an abstract CSP instance, that does not distinguish values above p_{max} and prove that this is equivalent to Bäckströms CSP instance in our case.

The rest of the paper is organised as follows. Section 2 contains some necessary definitions from graph theory, parameterised complexity and planning. Section 3 briefly recapitulates some results from Bäckström (2014). Section 4 analyses the case where the DTGs are paths or cycles, and shows how to construct an abstract CSP instance that is equivalent to the planning instance. Section 5 extends this result to the more general case where DTGs are cycle DAGs. This section also contains the fpt results. The paper closes with a discussion in Section 6.

2 Preliminaries

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

2.1 Graphs

If $G = \langle V, E\rangle$ is a directed graph, then $U(G) = \langle V, E_U\rangle$ is the undirected version of G, where $E_U = \{\{v, w\} \mid \langle v, w\rangle \in E\}$. A *polytree* is a directed acyclic graph G such that $U(G)$ is a tree, i.e. if ignoring the direction of the edges then G is connected and contains no cycles.

A *tree decomposition* of an undirected graph $G = \langle V, E\rangle$ is a tuple $\langle N, T\rangle$ where $N = \{N_1, \ldots, N_n\}$ is a family of subsets of V and T is a tree with nodes $N_1, \ldots, N_n$, satisfying the following properties: (1) The union of all sets N_i equals V, i.e. each graph vertex is contained in at least one tree node. (2) For each $v \in V$, the nodes in N that contain v form a connected subtree of T. (3) For every edge $\{v, w\}$ in the graph, there is a node N_i such that $v, w \in N_i$, i.e. adjacent vertices in G must have a tree node in common. The *width* of a tree decomposition is the size of its largest node N_i minus one. The *treewidth* of a graph G is the minimum width among all possible tree decompositions of G. We define the treewidth of a directed graph G as the treewidth of the corresponding undirected graph $U(G)$. Hence, a polytree has treewidth 1 since a tree has treewidth 1.

2.2 Parameterised Complexity

We define the basic notions of parameterised complexity and refer to other sources (Downey and Fellows 1999; Flum and Grohe 2006) for an in-depth treatment. A *parameterised problem* is a set of pairs $\langle \mathbb{I}, k\rangle$, the *instances*, where $\mathbb{I}$ is the main part and k is the *parameter*. The parameter is usually one or more non-negative integers. A parameterised problem is *fixed-parameter tractable (fpt)* if there exists an algorithm that solves any instance $\langle \mathbb{I}, k\rangle$ of size n in time $f(k) \cdot n^c$ where f is an arbitrary computable function and c is a constant independent of both n and k. That is, the expression can be separated into a function $f(k)$ that depends only on the parameter(s) and a polynomial function n^c that depends only on the instance size. **FPT** is the class of all fixed-parameter tractable decision problems. A parameter is not the same thing as assuming that the value is a constant. For instance, although $O(n^k)$ is tractable in the classical sense if k is a constant, the expression is not fpt when k is the parameter since it is not separable. Although the parameter value is often assumed much smaller than the instance size, it does not even need to be polynomially bounded in the instance size. Parameterised complexity offers a completeness theory, similar to the theory of **NP**-completeness, based on a hierarchy of complexity classes $\mathbf{FPT} \subseteq \mathbf{W}[1] \subseteq \mathbf{W}[2] \subseteq \mathbf{W}[3] \subseteq \cdots$, where the class $\mathbf{W}[1]$ is usually considered as a parameterised analogue of **NP**, although neither class is included in the other. We will not go further into this since we primarily prove tractability results in this paper.

2.3 Planning Framework

We use the SAS$^+$ planning framework (Bäckström and Nebel 1995). A *planning instance* is a tuple $\mathbb{P} = \langle V, D, A, s_I, s_G\rangle$ where V is a finite set of *variables* $v_1, \ldots, v_n$, each with a finite *domain* $D(v_i)$, defining the space of *total states* $S(V, D) = D(v_1) \times \ldots, \times D(v_n)$. Furthermore, A is a set of *actions* and $s_I, s_G \in S(V, D)$ is the *initial state* and *goal*, respectively. The value of a variable v_i in a state s is called the *projection* of s onto v_i and is denoted $s[v_i]$. A *partial state* also allows the undefined value u and $\mathrm{vars}(s)$ denotes the set of variables with a defined value in a partial state s, i.e. $s[v_i] \neq \mathsf{u}$ for $v_i \in \mathrm{vars}(s)$. Each action $a \in A$ has a *precondition* $\mathrm{pre}(a)$ and an *effect* $\mathrm{eff}(a)$, which are both partial states. An action a is *valid* in a state s if $\mathrm{pre}(a)[v_i] = s[v_i]$ for all $v_i \in \mathrm{vars}(\mathrm{pre}(a))$. The *result* of a in s is the state t defined such that $t[v_i] = \mathrm{eff}(a)[v_i]$ for $v_i \in \mathrm{vars}(\mathrm{eff}(a))$ and $t[v_i] = s[v_i]$ otherwise.

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

Projection is extended as follows. Let $V' \subseteq V$. If s is a state, then $s[V']$ is a partial state that agrees with s on all variables in V' and is otherwise undefined. If a is an action, then $a[V']$ is an action a' with precondition pre$(a)[V']$ and effect eff$(a)[V']$. For a planning instance $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$, define $\mathbb{P}[V'] = \langle V', A[V'], s_I[V'], s_G[V'] \rangle$, where $A[V'] = \{a[V'] \mid a \in A \text{ and } \text{eff}(a[V']) \neq \varnothing\}$, i.e. actions with no projected effect are ignored. For an action sequence $\omega = \langle a_1, \ldots, a_\ell \rangle$, first define the sequence $\omega' = \langle a'_1, \ldots, a'_\ell \rangle$ such that $a'_i = a_i[V']$ for all i ($1 \leq i \leq \ell$), and then define $\omega[V']$ as the subsequence of ω' that contains only those a'_i where eff$(a'_i) \neq \varnothing$. This follows common practice in the literature (cf. Helmert (2004)).

The *transition graph* for $\mathbb{P}$ is the labelled directed graph TG$(\mathbb{P}) = \langle S, E \rangle$, where $S = S(V, D)$ and $E \subseteq S \times A \times S$ such that for all $s, t \in S$ and $a \in A$, $\langle s, a, t \rangle \in E$ if a is valid in s and t is the result of a in s. Obviously, the paths from s_I to s_G in TG$(\mathbb{P})$ correspond to the plans for $\mathbb{P}$. The *domain transition graph (DTG)* for a variable $v \in V$ is DTG$(v) =$ TG$(\mathbb{P}[v])$, i.e. the paths from $s_I[v]$ to $s_G[v]$ in DTG(v) describe all possible ways to go from the initial state to the goal for this particular variable treated in isolation. Both the transition graph and the DTGs are multigraphs since different actions can induce different edges between the same pair of vertices. The *causal graph* for $\mathbb{P}$ is the directed graph CG$(\mathbb{P}) = \langle V, E \rangle$ where E contains the edge $\langle v, w \rangle$ for every pair of distinct vertices $v, w \in V$ such that (1) $v \in$ vars(pre(a)) $\cup$ vars(eff(a)) and (2) $w \in$ vars(eff(a)) for some action $a \in A$. The *causal graph* describes how the variables of the instance depend on each other, as implicitly defined by the actions.

3 Acyclic DTGs

Bäckström (2014) has studied the parameterised complexity of planning when all DTGs of an instance are acyclic. He considered the following three parameters: the maximum domain size of the variables (parameter d), the maximum number of paths between the initial and goal values in the DTGs (parameter k) and the treewidth of the causal graph (parameter w). He proved that planning is fpt in the parameters d, k and w in the general case and fpt in only parameter k when the causal graph is a polytree. His results exploited that a planning instance with acyclic DTGs can be modelled as an equivalent CSP instance with variable domains that are bounded by the parameters. We will use the same technique and thus need to recapitulate some definitions and results.

Definition 1. An instance of the *constraint satisfaction problem (CSP)* is a triple $\mathbb{C} = \langle X, D, C \rangle$, where X is a set of variables, D is a function assigning a domain to each variable and C is a finite set of constraints. Each constraint is a tuple $\langle t, R \rangle$ where t is a sequence $\langle x_{i_1}, \ldots, x_{i_r} \rangle$ of variables from X and R is a relation $R \subseteq D(x_{i_1}) \times \ldots \times D(x_{i_r})$. A *solution* for $\mathbb{C}$ is a mapping α that maps each $x_i \in X$ to an element in $D(x_i)$ such that all constraints in C are satisfied, i.e. $R(\alpha(x_{i_1}), \ldots, \alpha(x_{i_r}))$ holds for every constraint $\langle \langle x_{i_1}, \ldots, x_{i_r} \rangle, R \rangle$ in C. When all constraint relations are binary, the *constraint graph* for $\mathbb{C}$ is the graph $G = \langle X, E \rangle$ where E contains the edge $\{x_i, x_j\}$ whenever there is some constraint $\langle t, R \rangle$ such that $t = \langle x_i, x_j \rangle$ or $t = \langle x_j, x_i \rangle$.

The following construction defines an equivalent CSP instance for every planning instance.

Construction 2. (Bäckström 2014, Construction 6) Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance and let $\langle N, T \rangle$ be a tree decomposition of CG$(\mathbb{P})$. Define a corresponding CSP instance $\mathbb{C} = \langle X, D, C \rangle$ as follows. The set X contains one variable x_i for each node $N_i \in N$ where the domain $D(x_i)$ for x_i is the set of plans for $\mathbb{P}[N_i]$. For each pair of adjacent nodes N_i, N_j in T such that $i < j$, define the relation $R_{i,j} \subseteq D(x_i) \times D(x_j)$ such that $R_{i,j}$ contains exactly those tuples $\langle \omega_i, \omega_j \rangle$ where $\omega_i[N_i \cap N_j] = \omega_j[N_i \cap N_j]$, i.e. those tuples where ω_i and ω_j agree on the actions affecting the common variables.

Lemma 3. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance, let $\langle N, T \rangle$ be a tree decomposition of CG$(\mathbb{P})$ and let $\mathbb{C} = \langle X, D, C \rangle$ be the corresponding CSP instance according to Construction 2. Then, $\mathbb{P}$ has a solution if and only if $\mathbb{C}$ has a solution.*

Proof. Immediate from Lemma 8 in Bäckström (2014) by noting that neither acyclic DTGs nor finite CSP domains are necessary for the proof. $\square$

4 Path-or-cycle DTGs

As a first step away from acyclic DTGs we consider planning instances where every DTG is either a directed path graph or a directed cycle graph. Although we will later generalize this, most of the interesting problems arise already here. Hence, we make a more detailed presentation of this simpler case and allow ourselves to be more sketchy later.

Let $G = \langle V, E \rangle$ be an edge labelled digraph. A sequence $\sigma = v_0, \ell_1, v_1, \ell_2, \ldots, \ell_n, v_n$ such that $\langle v_{i-1}, \ell_i, v_i \rangle \in E$ for all i ($1 \leq i \leq n$) is a *walk* from v_0 to v_n. Repetitions of vertices and labels are allowed in a walk. We also define the corresponding *vertex walk* $V\langle \sigma \rangle = v_0, v_1, \ldots, v_n$ and *label walk* $L\langle \sigma \rangle = \ell_1, \ell_2, \ldots, \ell_n$, as well as the corresponding *vertex set* $V\{\sigma\} = \{v_0, \ldots, v_n\}$ and *label set* $L\{\sigma\} = \{\ell_1, \ldots, \ell_n\}$. We will frequently assume that a graph has two designated vertices v_{init} and v_{goal}, which may coincide, and we refer to a walk as *complete* if it is a walk from v_{init} to v_{goal}. In particular, $v_{init} = s_I[v]$ and $v_{goal} = s_G[v]$ in DTG(v) for a variable v.

Let C be a directed cycle graph with vertices $v_1, \ldots, v_n$ and edges $\langle v_1, \ell_1, v_2 \rangle, \ldots, \langle v_n, \ell_n, v_1 \rangle$. Suppose $v_{init} = v_1$ and $v_{goal} = v_k$, for some k ($1 \leq k \leq n$). The walk $v_1, \ell_1, v_2, \ldots, v_n, \ell_n, v_1$ can be written as α, β, v_1, where $\alpha = v_1, \ell_1, \ldots, \ell_{k-1}, v_k$ and $\beta = \ell_k, v_{k+1}, \ldots, v_n, \ell_n$. Note that α must at least contain vertex v_1 (in the case where $v_1 = v_k$) while β must always contain at least one label. The cycle C can be defined by α and β and we will frequently write $C = \alpha\beta$ to make such definitions. Every complete walk (i.e. from v_1 to v_k) can be written as $\alpha(\beta\alpha)^p$ for some $p \geq 0$. We refer to p as the *signature* of the walk and it specifies how many times we walk around the cycle. We also use the shorthand $C^p = \alpha(\beta\alpha)^p$. Note that C and C^1 are different; C denotes the actual cycle graph while C^1 denotes the walk $\alpha\beta\alpha$ in C. There is a one-to-one correspondence between the natural numbers and the set of complete walks in C, so the signature of a complete walk is a sufficient description of it. We also define the signature of a path graph as 0. This allows for treating paths and cycles alike and it is logical; if α is a path, then $\alpha = \alpha(\beta\alpha)^0$ for any β.

We allow the special signature value $*$ to denote an unbounded value, and allow arithmetics with this value by treating $*$ as infinity, that is, for all integers p we have $p \leq *$, $p \neq *$, $\min\{p, *\} = p$ etc., and we have $* = *$, $* \leq *$, $\min\{*, *\} = *$ etc. The set $\mathbb{N}_*$ is defined as $\mathbb{N}_* = \mathbb{N} \cup \{*\}$.

Let σ be a walk in DTG(v) for some variable v. We define the function $R_u\langle\sigma\rangle$ as the sequence of *requested values* from another variable u as follows. Assume $L\langle\sigma\rangle = a_1, \ldots, a_n$, which is a sequence of actions. Define the sequence $\rho = \text{pre}(a_1)[u], \ldots, \text{pre}(a_n)[u]$ of preconditions on variable u. First remove all occurences of the undefined value u. Then remove all consecutive repetitions of values. The resulting sequence is $R_u\langle\sigma\rangle$. For instance, if $\rho = x, \mathsf{u}, x, y, y, x, y, \mathsf{u}, y$, then $R_u\langle\sigma\rangle = x, y, x, y$. We also define $R_u\{\sigma\}$ as the set of values occuring in $R_u\langle\sigma\rangle$.

A sequence σ is a *subsequence* of another sequence σ' if all symbols in σ appear in the same order in σ'. For instance, x, y, z, z, y is a subsequence of $w, y, x, w, y, z, x, y, z, w, z, y$.

Definition 4. Let $\mathbb{P} = \langle V, D, A, s_I, s_G\rangle$ be a planning instance, let $u, v \in V$, let DTG$(v) = C_v$ and let DTG$(u) = C_u$. Then:

(1) For all $p, q \geq 0$, C_u^q matches C_v^p if $R_u\langle C_v^p\rangle$ is a subsequence of $V\langle C_u^q\rangle$.

(2) For all $q \geq 0$, C_u^q matches C_v^* if C_u^q matches C_v^p for all $p \geq 0$.

(3) For all $p \geq 0$, C_u^* matches C_v^p if C_u^q matches C_v^p for some $q \geq 0$.

(4) C_u^* matches C_v^* if for every $p \geq 0$ there is some $q \geq 0$ such that C_u^q matches C_v^p.

Note that if C_u^q matches C_v^p, then C_u^r matches C_v^p for all $r \geq q$. Matching is immediately related to planning in the following way.

Lemma 5. *Let $\mathbb{P}$ be a planning instance such that CG$(\mathbb{P})$ is acyclic, let $\langle u, v\rangle$ be an edge in CG$(\mathbb{P})$ and let DTG$(u) = C_u$ and DTG$(v) = C_v$, where each of C_u and C_v is a path or a cycle. Then:*

(1) If there is a plan ω for $\mathbb{P}[\{u, v\}]$, such that $\omega[u] = C_u^q$ and $\omega[v] = C_v^p$, then C_u^q matches C_v^p.

(2) If C_u^q matches C_v^p, then there is a plan ω for $\mathbb{P}[\{u, v\}]$ such that $\omega[u] = C_u^q$ and $\omega[v] = C_v^p$.

Proof sketch. (1) Straightforward. (2) Assume that $R_u\langle C_v^p\rangle = z_1, \ldots, z_h$ for some h. Then $L\langle C_v^p\rangle$ can be partitioned into h consecutive blocks $\alpha_1, \ldots, \alpha_h$ such that $R_u\{\alpha_i\} = \{z_i\}$ for each i $(1 \leq i \leq h)$. Similarily, $L\langle C_u^q\rangle$ can be partitioned into $h + 1$ consecutive blocks $\beta_1, \ldots, \beta_{h+1}$ such that for each i $(1 \leq i \leq h)$, the last action a in β_i has eff$(a)[v] = z_i$. Obviously, $\beta_1, \alpha_1, \ldots, \beta_h, \alpha_h, \beta_{h+1}$ is a plan for $\mathbb{P}[\{u, v\}]$. ☐

The following property of matching is crucial for the forthcoming results. It says that when matching two walks C_u^q and C_v^p, there is a gap such that either p is upper bounded by $q + 1$ or unbounded; it cannot be the case that there is a match for some $p > q + 1$ but not for all $p > q + 1$.

Lemma 6. *Let $\mathbb{P}$ be a planning instance and let $\langle u, v\rangle$ be an edge in CG$(\mathbb{P})$. Let DTG$(u) = C_u$ and DTG$(v) = C_v$ be cycle graphs. For all $q \geq 0$, if C_u^q matches C_v^{q+2}, then C_u^q matches C_v^*.*

Proof. Assume C_u^q matches C_v^{q+2}. There are two cases:

1) If $|R_u\{C_v\}| \leq 1$, then $|R_u\{C_v\}| = 1$ since $\langle u, v\rangle$ is an edge in CG$(\mathbb{P})$. There is thus a value x such that $R_u\langle C_v\rangle = x$ and the walk C_u^q must include vertex x since C_u^q matches C_v^{q+2}. Hence, C_u^1 must match C_v^p for all $p \geq 0$.

2) If $|R_u\{C_v\}| \geq 2$, then there must be at least two distinct values x and y in $R_u\{C_v\}$. Each of x and y must occur at least $q + 2$ times in $R_u\langle C_v^{q+2}\rangle$, but since x and y are vertices in C_u each of them can appear at most $q + 1$ times in $V\langle C_u^q\rangle$. Hence, C_u^q cannot match C_v^{q+2} so this case is impossible. ☐

It is still possible that C_u^q matches C_v^{q+1}, without matching C_v^*. Let $C_v = \alpha\beta$ and $C_u = \gamma\delta$. If $R_u\{\alpha\} = \varnothing$, $|R_u\{\beta\}| = 2$ and $R_u\langle\beta\rangle$ is a subsequence of $V\langle\gamma\rangle$, then C_u^q matches C_v^{q+1} but not C_v^{q+2} for all $q \geq 0$.

Algorithm MaxMatch in Figure 1 exploits Lemma 6. It takes a specified signature q for C_u and computes the maximum signature p for C_v such that C_u^q matches C_v^p. The algorithm returns $p = *$ if there is no upper bound on p and it returns -1 if there is no match even for $p = 0$.

Algorithm PreProcess in Figure 2 computes the largest value p_v for each v such that $C_u^{p_u}$ matches $C_v^{p_v}$ for all edges $\langle u, v\rangle$. The algorithm terminates since a variable can be remarked only when its p_v value decreases, which can happen only a finite number of times. Note that the assignment of p_v values does not guarantee that there is a simultaneous matching for all edges, it only guarantees that if there is such a matching, then there is one with values $q_v \leq p_v$ for all v. Furthermore, if $p_v < 0$ for any v, then there cannot be a simultaneous matching.

We define an abstract CSP instance corresponding to a planning instance by exploiting the gap demonstrated in Lemma 6. Due to this gap, the signature for every variable is either bounded or unbounded. Define $p_{max} = \max_{v \in V} p_v$, where the p_v values are as computed by algorithm PreProcess. Then p_{max} is the upper bound for all variables with bounded signature. We can then define the abstract signature domain $\{0, 1, \ldots, p_{max}, *\}$, where the value $*$ is an aggregation of the values $\{p_{max} + 1, p_{max} + 2, \ldots\}$, and use this to define a CSP instance.

```
1  MaxMatch($C_u$,$q$,$C_v$)
2     $p := 0$
3     **while** $p \leq q + 2$ **do**
4        **if** $R_u\langle C_v^p\rangle$ is not a subsequence of $V\langle C_u^q\rangle$
5           **then return** $p - 1$
6        **else** $p := p + 1$
7     **return** $*$
```

Figure 1: Find maximal matching signature.

```
PreProcess($G = \langle V, E\rangle$)
  for all $v \in V$ do
    Mark $v$
    if DTG($v$) is a path then $p_v := 0$
    else $p_v := *$
  repeat
    Choose a marked $u \in V$
    Unmark $u$
    for all outgoing edges $\langle u, v\rangle$ do
      $q :=$ MaxMatch($C_u, p_u, C_v$)
      if $q < p_v$ then
        $p_v := q$
        Mark $v$
  until all $v \in V$ are unmarked
```

Figure 2: Signature preprocessing for arbitrary CG.

Construction 7. Let $\mathbb{P} = \langle V, D, A, s_I, s_G\rangle$ be a planning instance such that CG($\mathbb{P}$) is a polytree and DTG(v) is a cycle or path for each $v \in V$. Let $\langle N, T\rangle$ be an optimal tree decomposition of CG($\mathbb{P}$). Define a corresponding CSP instance $\mathbb{C}^A = \langle X^A, D^A, C^A\rangle$ as follows: (1) Let X^A contain one variable x_i for each $N_i \in N$. (2) For each $N_i \in N$, assume $N_i = \{u, v\}$ and that $\langle u, v\rangle$ is an edge in CG($\mathbb{P}$). Let $D^A(x_i) = \{\{\langle u, q\rangle, \langle v, p\rangle\} \mid q, p \in \{0, \ldots, p_{max}, *\}$ and C_u^q matches $C_v^p\}$. (3) For each pair of adjacent nodes N_i and N_j in T such that $i < j$, define $R^A_{i,j} \subseteq D^A(x_i) \times D^A(x_j)$ such that $R^A_{i,j}$ contains those tuples $\langle s_i, s_j\rangle$ where s_i and s_j share an identical element.

The restriction to polytree causal graphs guarantees that the width of T is 1 and that if $N_i = \{u, v\}$, then exactly one of $\langle u, v\rangle$ and $\langle v, u\rangle$ is an edge in CG($\mathbb{P}$). We now prove that this construction is equivalent to Construction 2.

Lemma 8. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G\rangle$ be a planning instance such that CG($\mathbb{P}$) is a polytree and DTG(v) is a path or cycle graph for each $v \in V$. Let $\langle N, T\rangle$ be an optimal tree decomposition of CG($\mathbb{P}$). Let $\mathbb{C}$ and $\mathbb{C}^A$ be the corresponding CSP instances. Then $\mathbb{C}$ has a solution if and only if $\mathbb{C}^A$ has a solution.*

Proof. $\Rightarrow$: Suppose $\mathbb{C}$ has a solution α. Tree decompositions require that all nodes containing a variable v form a connected subtree, so there is a unique action sequence ω_v for each $v \in V$ such that $\alpha(x_i)[v] = \omega_v$ for each node N_i in T that contains v. Furthermore, $\omega_v = C_v^{q_v}$ for some $q_v \geq 0$. Without losing generality, assume that $q_v \neq p_{max} + 1$ for all $v \in V$, which is possible since all signatures greater than p_{max} must be unbounded. Define $r_v = q_v$ if $q_v \leq p_{max}$ and otherwise $r_v = *$. Construct an assignment α^A for $\mathbb{C}^A$ as follows. For each $N_i = \{u, v\}$ in T, assume $\langle u, v\rangle$ is an edge in CG($\mathbb{P}$) and define $\alpha^A(x_i) = \{\{\langle u, r_u\rangle, \langle v, r_v\rangle\}\}$. Clearly, both r_u and r_v are in $\{0, \ldots, p_{max}, *\}$ and Lemma 5 guarantees that $C_u^{q_u}$ matches $C_v^{q_v}$ since $\alpha(x_i)$ is a plan for $\mathbb{P}[N_i]$. It remains to prove that $C_u^{r_u}$ matches $C_v^{r_v}$. The only non-trivial case is when $q_u \leq p_{max}$ and $q_v > p_{max}$, i.e. $r_u = q_u$ and $r_v = *$. Then $q_v > p_{max} + 1$, by assumption, so $q_v \geq q_u + 2$ and it follows from Lemma 6 that $C_u^{q_u}$ matches C_v^* and, thus, that $C_u^{r_u}$ matches $C_v^{r_v}$. Hence, $\alpha^A(x_i) \subseteq D^A(x_i)$ for all $x_i \in X^A$. Thus, for all adjacent N_i, N_j in T with common variable v, there are $s_i \in \alpha^A(x_i)$ and $s_j \in \alpha^A(x_j)$ s.t. $\langle v, r_v\rangle \in s_i$ and $\langle v, r_v\rangle \in s_j$, and thus $R^A_{i,j}(S_i, S_j)$ holds. It follows that α^A is a solution for $\mathbb{C}^A$.

$\Leftarrow$: Suppose $\mathbb{C}^A$ has a solution α^A. Let N_i and N_j be arbitrary adjacent nodes in T with common variable v. Then there is some q_v such that both $\alpha^A(x_i)$ and $\alpha^A(x_j)$ contain $\langle v, q_v\rangle$, since $R^A_{i,j}(\alpha^A(x_i), \alpha^A(x_j))$ must hold. Hence, α^A assigns a unique value q_v to each $v \in V$, since all nodes containing v must form a connected subtree of T. Construct an assignment α for $\mathbb{C}$ as follows. For each $v \in V$ define a value r_v such that $r_v = q_v$ if $q_v \neq *$ and otherwise set r_v to a suffciently high value. This can be done such that $C_u^{r_u}$ matches $C_v^{r_v}$ for all edges $\langle u, v\rangle$ in CG($\mathbb{P}$) since $C_u^{q_u}$ matches $C_v^{q_v}$ and CG($\mathbb{P}$) is acyclic. For each $N_i = \{u, v\}$ in T, assume that $\langle u, v\rangle$ is an edge in CG($\mathbb{P}$). Let $\alpha(x_i)$ assign values r_u to u and r_v to v. Then $C_u^{r_u}$ matches $C_v^{r_v}$ so according to Lemma 5 there is a plan ω_i for $\mathbb{P}[N_i]$. For all adjacent nodes N_i and N_j in T with common variable v, both $\alpha^A(x_i)$ and $\alpha^A(x_j)$ contain the tuple $\langle v, q_v\rangle$. Hence, $\omega_i[v] = \omega_j[v]$ for their common variable v. Let $\alpha(x_i) = \omega_i$ for all $x_i \in X$. Then α is a solution for $\mathbb{C}$. $\square$

Corollary 9. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G\rangle$ be a planning instance where CG($\mathbb{P}$) is a polytree and DTG(v) is a path or cycle for each $v \in V$. Let $\langle N, T\rangle$ be an optimal tree decomposition of CG($\mathbb{P}$). Let $\mathbb{C}^A$ be the corresponding CSP instance. Then $\mathbb{P}$ has a plan if and only if $\mathbb{C}^A$ has a solution.*

Proof. Combine Lemmata 3 and 8. $\square$

5 DTGs with Simple-cycle SCCs

A *strongly connected component (SCC)* of a directed (multi)graph $G = \langle V, E\rangle$ is a maximal subset $C \subseteq V$ such that for all $u, v \in C$, v is reachable from u in the subgraph $G_{|C}$. The *condensation* of G is the graph resulting from contracting each SCC in G into a single vertex.

Definition 10. A directed graph G is a *cycle-path graph* if (1) each SCC of G is a simple directed cycle and (2) the condensation of G is a directed path graph.

Let P be a cycle-path graph with m cycles $C_1, \ldots, C_m$, in that order, where $C_i = \alpha_i\beta_i$ for each i. Every complete walk in P is then on the form $\sigma = \gamma_0 C_1^{p_1}\gamma_1 C_2^{p_2}\gamma_2 \ldots \gamma_{m-1}C_m^{p_m}\gamma_m$. Either or both of γ_0 and γ_m may be empty, in the case P starts and/or ends with a cycle, while $\gamma_1, \ldots, \gamma_{m-1}$ must contain at least one label each, since two cycles could otherwise share a vertex which would violate condition 1 of the definition. We define the signature of σ as $\langle p_1, \ldots, p_m\rangle$ and refer to m as the size of the signature. Since all walks in P must have the same signature size, we also refer to this as the signature size of the graph P. For signatures of the same size, we define comparisons as follows. Let $s_1 = \langle q_1, \ldots, q_m\rangle$ and $s_2 = \langle p_1, \ldots, p_m\rangle$. Then $s_1 \leq s_2$ if $q_i \leq p_i$ for all i ($1 \leq i \leq m$) and $s_1 = s_2$ if both $s_1 \leq s_2$ and $s_2 \leq s_1$. We also define $\min\{s_1, s_2\} = \langle \min\{q_1, p_q\}, \ldots, \min\{q_m, p_m\}\rangle$. Also here there is a one-to-one correspondence between the tuples in $\mathbb{N}^m$ and the complete walks for P, so every complete walk in P can be uniquely characterized by a signature. Also here, $p_i = *$ is allowed in signatures. Note that maximal matching signatures are no longer unique. For instance, it may happen that two cycles require the same precondition values so the sum of walks around these cycles is limited. A simple method to find an upper bound for signatures is the following. Assume we are matching P_u with P_v. Compute the maximal signature p_i for cycle $C_{v,i}$ in P_v by computing $r_j = MaxMatch(C_{u,j}, q_j, C_{v,i})$ for each cycle $C_{u,j}$ in P_u and then let p_i be the maximum of these r_j values. Algorithm PreProcess can then be modified to use this matching method instead of MaxMatch.

Definition 11. Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance, let $u, v \in V$, let DTG$(v) = P_v$ be a cycle path with m cycles $C_{v,1}, \ldots, C_{v,m}$ and let DTG$(u) = P_u$ be a cycle path with n cycles $C_{u,1}, \ldots, C_{u,n}$. Then: (1) For arbitrary $s \in \mathbb{N}^m$ and $t \in \mathbb{N}^n$, P_u^t matches P_v^s if $R_u\langle P_v^s \rangle$ is a subsequence of $V\langle P_u^t \rangle$. (2) For arbitrary $s \in \mathbb{N}_*^m$ and $t \in \mathbb{N}^n$, P_u^t matches P_v^s if P_u^t matches $P_v^{\min\{s,r\}}$ for all $r \in \mathbb{N}^m$.

For instance, if $P_u^{\langle q_1,q_2,q_3,q_4 \rangle}$ matches $P_v^{\langle p_1,*,p_3,* \rangle}$, then $P_u^{\langle q_1,q_2,q_3,q_4 \rangle}$ matches $P_v^{\langle r_1,r_2,r_3,r_4 \rangle}$ for all $r_1, \ldots, r_4$ ($0 \leq r_1 \leq p_1$, $0 \leq r_3 \leq p_3$ and $r_2, r_4 \geq 0$). We now arrive at our final case, which subsumes all the previous ones.

Definition 12. A directed multigraph G is a *cycle DAG* if each SCC of G is a simple directed cycle.

Note that there cannot be two different edges between the same SCCs in G with the same label, since this would either violate the definition or require disjunctive preconditions. On the other hand, edges with different labels will remain as separate edges also in the condensation of G, which must be a multigraph. Hence, there is a one-to-one correspondence between the cycle-paths in G and the paths in its condensation. We can thus alternatively characterize G by its set of cycle paths, which we exploit to construct a CSP instance for every planning instance with cycle-DAG DTGs.

Construction 13. Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance such that CG$(\mathbb{P})$ is a polytree and DTG$(v) = G_v$ is a cycle DAG for each $v \in V$. For each $v \in V$, let k_v be the number of paths in the condensation of G_v and let $m_{v,h}$ be the signature size for cycle path h in G_v for each h ($1 \leq h \leq k_v$). Let $\langle N, T \rangle$ be an optimal tree decomposition of CG$(\mathbb{P})$. Define a corresponding CSP instance $\mathbb{C}^A = \langle X^A, D^A, C^A \rangle$ as follows: (1) Let X^A contain one variable x_i for each node $N_i \in N$. (2) For each $N_i \in N$, assume $N_i = \{u, v\}$ and that $\langle u, v \rangle$ is an edge in CG$(\mathbb{P})$. Let $D^A(x_i) = \{\{\langle u, g, s \rangle, \langle v, h, t \rangle\} \mid s \in \{0, \ldots, p_{max}, *\}^{m_{u,g}}$, $t \in \{0, \ldots, p_{max}, *\}^{m_{v,h}}$, $0 \leq g \leq k_u$, $0 \leq h \leq k_v$ and P_u^s matches $P_v^t\}$. (3) For all pairs of adjacent nodes N_i and N_j in T, define the relation $R_{i,j}^A \subseteq D^A(x_i) \times D^A(x_j)$ such that $R_{i,j}^A$ contains those tuples $\langle s_i, s_j \rangle$ where s_i and s_j share an identical element.

Also this construction is equivalent to Construction 2.

Lemma 14. *Let $\mathbb{P} = \langle V, D, A, s_I, s_G \rangle$ be a planning instance such that CG$(\mathbb{P})$ is a polytree and DTG(v) is a cycle DAG for each $v \in V$. Let $\langle N, T \rangle$ be an optimal tree decomposition of CG$(\mathbb{P})$. Let $\mathbb{C}$ and $\mathbb{C}^A$ be the corresponding CSP instances. Then $\mathbb{C}$ has a solution if and only if $\mathbb{C}^A$ has a solution.*

Proof sketch. Analogous to the proof of Lemma 8, with two exceptions. 1) Signatures are tuples of integers. 2) $D(x_i)$ contains tuples for all combinations of abstract signatures and paths for the two variables in N_i. Each tuple is marked with variable and path index, so $R_{i,j}^A$ still holds if adjacent nodes share a tuple, i.e. if $\alpha^A(x_i)$ and $\alpha^A(x_j)$ specify the same path and signature for the common variable. $\square$

We can, thus, decide if a planning instance $\mathbb{P}$ with cycle-DAG DTGs and a polytree causal graph has a solution by constructing the corresponding instance $\mathbb{C}^A$ and solve it.

Theorem 15. *Let C be a class of planning instances $\mathbb{P}$ such that CG$(\mathbb{P})$ is a polytree and all DTGs are cycle DAGs. Then the problem of deciding if an instance of C has a solution is fpt in the parameters c, d, k and p_{max}.*

Proof. Combining Lemmata 3 and 14 yields that $\mathbb{P}$ has a plan if and only if $\mathbb{C}^A$ has a solution, so it remains to prove that constructing and solving $\mathbb{C}^A$ is fpt.

To find all cycle-paths in a DTG we first identify the SCCs. This takes polynomial time (Tarjan 1972), and so does computing the condensation. Finding the (at most) k paths in the condensation takes time $O(k \cdot \mathit{poly}(n))$ (Bäckström 2014, Lemma 3), so generating all cycle paths takes time $O(k \cdot \mathit{poly}(n))$. There are at most $k(p_{max}+2)^c$ different abstract signatures for each DTG, so $|D(x_i)| \leq k^2(p_{max}+2)^{2c}$ for each x_i. Only bounded cycles need to be matched exactly. Expanding these in a cycle path results in a walk of length at most $d^{p_{max}}$, so matching all signature pairs for the two variables in a node N_i to construct $D(x_i)$ takes time $O(d^{p_{max}}k^2(p_{max}+2)^{2c} \cdot \mathit{poly}(n))$. Constructing $R_{i,j}^A$ for two nodes requires comparing all values in two domains pairwise, which takes time $O(c(k^2(p_{max} + 2)^{2c})^2 \cdot \mathit{poly}(n))$, since each signature is of size c at most. Hence, constructing $\mathbb{C}^A$ is fpt in c, d, k and p_{max}. Solving a CSP instance where the constraint graph is a tree takes time $O(D^2N)$, where D is the domain size and N the number of variables (Dechter and Pearl 1989), so solving $\mathbb{C}^A$ takes time $O((k^2(p_{max}+2)^{2c})^2 \cdot \mathit{poly}(n))$, which is fpt in c, k and p_{max}. $\square$

The parameter c is not strictly necessary since $c \leq d$, but using it allows for a more fine-grained analysis. We also get the corollary that planning for the two previous cases is fpt in parameters c,d and p_{max}.

6 Discussion

Katz and Keyder (2012) have previously considered the number of paths in the condensation of a DTG, but only for a very particular case and standard complexity analysis. So-called *factored planning* also uses CSP techniques in a similar way (cf. Brafman and Domshlak (2013), Fabre et al. (2010)). However, there the primary goal is to use CSP techniques to solve planning in general, while our goal is different and CSP techniques are only used as a tool for the proofs. This topic is further discussed in Bäckström (2014).

For acyclic DTGs it is also fpt to generate a plan (Bäckström 2014), but this is not obviously true for non-acyclic DTGs. There are instances where each DTG is only a cycle, but where the shortest plans are of exponential length. It thus takes exponential time to generate a plan. However, there is one possibility to investigate. There are classes with exponential shortest plans but where it is still possible to generate a solution in polynomial time, in the form of a hierarchical macro plan (Jonsson 2009). Also plan length optimisation is fpt in the case of acyclic DTGs (Bäckström 2014), but it is not obvious that this problem is fpt for the classes studied in this paper. While a bounded walk in a cycle implicitly specifies a certain number of actions, this is lost in the abstraction where * signatures are allowed.

One obvious continuation of this work is to consider instances with cycle-DAG DTGs but arbitrary causal graphs. Another way forward is to consider more relaxed types of SCCs. For instance, one may use the number of vertices or the number of cycles in an SCC as a parameters. There may also be cases where the whole DTG is strongly connected, but where it is possible to find some recursive structure in it. For instance, one might find a simple cycle that can be viewed as an 'outer cycle', treating all other vertices as 'interior'. This interior subgraph is not necessarily strongly connected, so one may consider it recursively and identify its SCCs. Finding such interior structure may provide a new way to study DTGs.

Acknowledgments

The reviewers provided very detailed and useful comments.

References

Bäckström, C., and 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.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Computat. Intell.* 11:625–656.

Bäckström, C.; Chen, Y.; Jonsson, P.; Ordyniak, S.; and Szeider, S. 2012. The complexity of planning revisited - a parameterized analysis. In *Proc. 26th AAAI Conf. Artif. Intell. (AAAI-12), Toronto, ON, Canada*, 1735–1741.

Bäckström, C. 2014. Parameterising the complexity of planning by the number of paths in the domain-transition graphs. In *Proc. 21st European Conf. Artif. Intell. (ECAI-14), Prague, Czech Republic*, 33–38.

Brafman, R. I., and Domshlak, C. 2013. On the complexity of planning for agent teams and its implications for single agent planning. *Artif. Intell.* 198:52–71.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artif. Intell.* 69(1–2):165–204.

Cooper, M.; Maris, F.; and Régnier, P. 2014. Monotone temporal planning: Tractability, extensions and applications. *J. Artif. Intell. Res.* 50:447–485.

de Haan, R.; Roubícková, A.; and Szeider, S. 2013. Parameterized complexity results for plan reuse. In *Proc. 27th AAAI Conf. Artif. Intell. (AAAI-13), Bellevue, WA, USA*.

Dechter, R., and Pearl, J. 1989. Tree clustering for constraint networks. *Artif. Intell.* 38(3):353–366.

Domshlak, C., and Dinitz, Y. 2001. Multi-agent off-line coordination: Structure and complexity. In *Proc. 6th European Conf. Planning (ECP-01), Toledo, Spain*.

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

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

Fabre, E.; Jezequel, L.; Haslum, P.; and Thiébaux, S. 2010. Cost-optimal factored planning: Promises and pitfalls. In *Proc. 20th Int'l Conf. Automated Planning and Scheduling (ICAPS-10), Toronto, ON, Canada*, 65–72.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer.

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

Giménez, O., and 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.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. 14th Int'l Conf. Automated Planning and Scheduling (ICAPS-04), Whistler, BC, Canada*, 161–170.

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

Jonsson, P., and Bäckström, C. 1998a. State-variable planning under structural restrictions: Algorithms and complexity. *Artif. Intell.* 100(1-2):125–176.

Jonsson, P., and Bäckström, C. 1998b. Tractable plan existence does not imply tractable plan generation. *Ann. Math. Artif. Intell.* 22(3-4):281–296.

Jonsson, A. 2009. The role of macros in tractable planning. *J. Artif. Intell. Res.* 36:471–511.

Katz, M., and Domshlak, C. 2008. New islands of tractability of cost-optimal planning. *J. Artif. Intell. Res.* 32:203–288.

Katz, M., and Domshlak, C. 2010. Implicit abstraction heuristics. *J. Artif. Intell. Res.* 39:51–126.

Katz, M., and Keyder, E. 2012. Structural patterns beyond forks: Extending the complexity boundaries of classical planning. In *Proc. 26th AAAI Conf. Artif. Intell. (AAAI-12), Toronto, ON, Canada*, 1779–1785.

Kronegger, M.; Ordyniak, S.; and Pfandler, A. 2014. Backdoors to planning. In *Proc. 28th AAAI Conf. Artif. Intell. (AAAI-14), Québec City, QC, Canada.*, 2300–2307.

Kronegger, M.; Pfandler, A.; and Pichler, R. 2013. Parameterized complexity of optimal planning: A detailed map. In *Proc. 23rd Int'l Joint Conf. Artif. Intell. (IJCAI-13), Beijing, China*, 954–961.

Tarjan, R. E. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1(2):146–160.