Scanning Phylogenetic Networks is NP-hard

Vincent Berry, Celine Scornavacca, Mathias Weller

Scanning Phylogenetic Networks is NP-hard

Vincent Berry[1,2], Celine Scornavacca[2], and Mathias Weller[3]

[1]LIRMM, IBC, Université de Montpellier, France
[2]ISEM, IBC, Université de Montpellier, France
[3]LIGM, Université Paris EST, Marne-la-Vallée, France

October 12, 2022

Abstract

Phylogenetic networks are rooted directed acyclic graphs used to depict the evolution of a set of species in the presence of reticulate events. Reconstructing these networks from molecular data is challenging and current algorithms fail to scale up to genome-wide data. In this paper, we introduce a new width measure intended to help design faster parameterized algorithms for this task. We study its relation with other width measures and problems in graph theory and finally prove that deciding it is NP-complete, even for very restricted classes of networks.

1 Introduction

Phylogenetic networks are rooted directed acyclic graphs used to depict the evolution of a set of species in the presence of reticulate events such as hybridizations, where two species combine their genetic material to create a new species (see nodes H_1 and H_2 in Fig. 1(left)) [9]. Herein, leaves represent the studied species and the root their most recent common ancestor, from which time flows away (as indicated by the direction of the arcs). Internal vertices represent either speciation events (a single parent) or reticulation events (several parents). Each arc represents the evolution of a species in time, during which each gene in the species genome can change due to mutations, allowing different forms of a gene (*alleles*) to appear among species, and even among individuals within the same species. Though the species history is modeled by a network, the evolution of a single non-recombinant gene can always be depicted by a tree, see Fig. 1(center), embedded in the species network, see Fig. 1(right).

Usually, a species network is inferred from a DNA dataset $S = \{S_1, \ldots, S_L\}$ composed of L genes sequenced from the genome of one or several individuals for each studied species [15]. To find the best phylogenetic network explaining S, a possibility is to sample many different networks N and compute the probability $P(S|N)$ of each N given S. Without giving all details here (they can be found for instance in [15]), $P(S|N)$ can be computed from the individual probabilities $P(G_i|N)$ of gene trees $G_1, \ldots, G_L$ for the L loci given N. In turn, each $P(G_i|N)$ can be computed from the probabilities of all possible embeddings of G_i in N, weighted by their respective probability depending on S_i, i.e. $P(G_i|N, S_i)$. See Fig. 1(center) for a gene tree and Fig. 1(right) for one of its possible embeddings within the network. Thus, heavy computations are needed to obtain $P(S|N)$ and current algorithms fail to scale to genome-wide data. To design faster algorithms, it is possible to integrate out the possible gene trees and embeddings, as done in [5]. To apply this technique to network inference we designed new partial likelihood formulae to compute $P(S|N)$ and stumbled on a new width parameter for DAGs that clearly puts into evidence why our approach is faster than existing ones, allowing us to handle several real-world datasets within minutes instead of weeks [12]. In this paper, we introduce this new parameter, which we call *scanwidth*, we study its relation with other parameters and problems in graph theory and finally prove that deciding it is NP-hard. A common and intuitive idea when working with phylogenetic networks is to exploit the observation that reticulation should be rare in practice to design algorithms that are fast for only mildly reticulate networks. This tree-likeness is often measured by the tree-width of the input. However, tree decompositions are in no way obligated to follow the leaf-to-root structure that phylogenies naturally impose and this makes dynamic programming on decomposition trees unnecessarily complicated. The scanwidth remedies this problem by forcing the leaves of the network to correspond to the leaves of the decomposition tree, yielding a form of tree-like cutwidth. Thus, our work broadens the arsenal of width measures that can be – and recently have

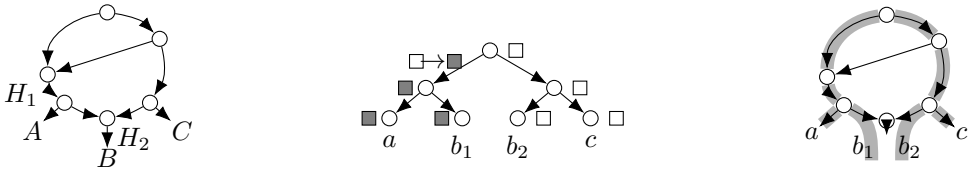


Figure 1: **Left:** A phylogenetic network N depicting the evolutionary history of species A, B, and C. **Center:** An evolution scenario for a gene, given the sequences of one individual from species A, one from C and two from B, where different alleles (boxes) are observed: gray for A and for one individual from B, and white for the other individuals. The arc containing the mutation from the white to the gray allele is marked. **Right:** An embedding (gray arcs) in N of this gene evolution scenario.

been – used to attack hard problems in phylogenetics [4, 8, 11]. To get an intuition, imagine a (possibly red) scanner line traversing a network from the leaves to the root; at any moment, its *width* is the number of arcs it cuts. As the line moves up, it traverses nodes, changing the set of arcs it cuts and, hence its width. The cutwidth of the network is the largest width achieved by such a traversing line. Now, consider multiple independent scanner lines, each one scanning an arc incoming to a different leaf of the network. Whenever a node could be passed by two different lines, they are merged to form a single one. This naturally generalizes the cutwidth to a stronger (that is, smaller) width measure that we call *scanwidth*. As with the cutwidth, different orders in which the nodes are passed imply different values of the final width and the goal is to minimize it. In many optimization approaches for phylogenetic networks, a network is traversed from the leaves up to its root, while computing some quantities. For some applications, computations on tree-parts can be done independently for each arc but, when meeting a reticulation node, computations on both arcs entering the node have to be considered jointly. This inter-dependence makes computing the required quantities more time consuming. In such cases, one really wants to process the network while minimizing the numbers of arcs considered jointly. This is captured by the scanwidth parameter.

In this work, we show that deciding the scanwidth of a network relates to an old problem in program optimization called Register Sufficiency (PO1 of Garey and Johnson [7]). Our proof comprises a non-trivial adaptation of an NP-hardness proof [13] for the latter problem to a very restricted class of rooted DAGs, on which Register Sufficiency coincides with deciding the cutwidth and the scanwidth (offset by 1). This hardness proof, as well as the scanwidth parameter itself, may be of independent interest to the design of algorithms for other problems on DAGs.

Note that computing the scanwidth and using it as a parameter for other algorithms are two different pairs of shoes and, though a parameterized algorithm may require a tree extension (see Section 2) to be given, there is still hope that the scanwidth can be approximated efficiently. Thus, in analogy with other highly successful (width) parameters such as the treewidth, the hybridization number or the hybridization level [2, 3, 10, 14], we point out that being NP-complete to compute does not hurt the practical usefulness of the scanwidth.

We defer some proofs to a long version of this paper.

2 Preliminaries

Phylogenetic Networks. Let G be a leaf-labelled, directed, acyclic graph with a single source (which is called "root"). The in-degree of a vertex v in G is $\deg^-_G(v)$ and its out-degree is $\deg^+_G(v)$, the sum of those being the degree of v. If all vertices of G have either in-degree one and out-degree zero (*leaves*), in-degree at most one and out-degree at least two (*tree-vertices*), and in-degree at least two and out-degree one (*reticulation*), then G is called *rooted phylogenetic network* (henceforth *network*). Note that the root is a special tree-vertex. We denote the set of leaves of G by $\mathcal{L}(G)$, the set of vertices by $V(G)$ and the tree-vertices by $V_T(G)$. If the root has degree two, the internal vertices have degree three, and the leaves have degree one, then G is called *binary*. If G contains a u-v-path for vertices u and v, we say that u is an ancestor of v (and v is a descendant of u) and we write $v <_G u$.

Vertex Orderings. A linear ordering σ of a subset V' of the vertices of a network G is called G-*respecting* if $u <_G v \Rightarrow u <_\sigma v$ for all $u, v \in V'$. A G-respecting ordering σ over $V(G)$ is called an *extension* (or "reverse topological order") of G, see Fig. 2. We call a tree Γ on $V(G)$ a *tree extension* for G if $x <_G y \Rightarrow x <_\Gamma y$ for all $x, y \in V(G)$. We denote the vertex at position i in σ by $\sigma(i)$ and $\sigma^{-1}(u)$

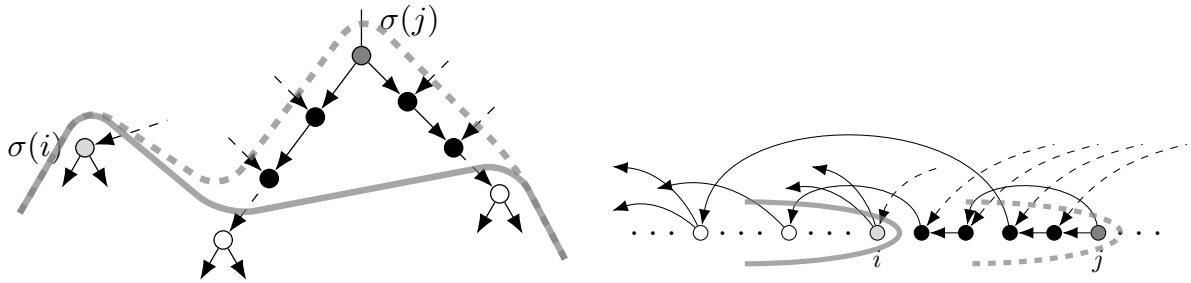


Figure 2: **Left:** For an extension σ each position i induces a cut through G separating the vertices in $\sigma[1..i]$ (below gray line) from $\sigma[i+1..]$ (above gray line). **Right:** G with vertices linearly arranged according to σ.

returns the position of the vertex u in sigma. Since positions and vertices are in bijection, we sometimes use vertices to represent their positions. We denote the sub-order of σ restricted to the elements of a set U by $\sigma[U]$ and we abbreviate $\sigma[\{\sigma(i), \sigma(i+1), \ldots, \sigma(j)\}] =: \sigma[i..j]$. A position i of σ is called a *milestone* if $\sigma(i)$ is a tree-vertex and σ is called *stable* if all maxima (wrt. $\leq_G$) in $\sigma[1..i]$ for any milestone i are tree-vertices (that is, each reticulation in $\sigma[1..i]$ has a parent in $\sigma[1..i]$). For disjoint orders σ and π let $\sigma \circ \pi$ denote the concatenation of σ with π (that is, σ followed by π). For a set X, we let (X) denote any order on the elements of X. Further, for distinct vertices or disjoint vertex sets $X_1, X_2, \ldots$, we abbreviate $(X_1) \circ (X_2) \circ \ldots =: (X_1, X_2, \ldots)$.

(Directed) Cutwidth. For an extension σ of a DAG G and a position i, we will use CW_i^σ to denote the set of arcs from a vertex in $\sigma[i+1..]$ to a vertex in $\sigma[1..i]$ and $\mathrm{cw}_i^\sigma := |\,\mathrm{CW}_i^\sigma\,|$ is called the *cutwidth* of σ at position i. The cutwidth of σ is $\mathrm{cw}(\sigma) := \max_i \mathrm{cw}_i^\sigma$ and the cutwidth of G, denoted $\mathrm{cw}(G)$, is the minimum of $\mathrm{cw}(\sigma)$ over all extensions σ of G. We allow i to be a vertex instead of a position, as σ is a bijection between the two.

(Directed) Register width. For an extension σ of G and a position i, we will use RW_i^σ to denote the set of vertices in $\sigma[1..i]$ that have a parent in $\sigma[i+1..]$ and $\mathrm{rw}_i^\sigma := |\,\mathrm{RW}_i^\sigma\,|$ is called the *register width* (also known as "vertex cut" or "separation" [6]) of σ at position i (again, we allow i to be a vertex instead of a position, as σ is a bijection between the two). The register width of σ is $\mathrm{rw}(\sigma) := \max_i \mathrm{rw}_i^\sigma$ and the register width of G, denoted $\mathrm{rw}(G)$, is the minimum over all extensions σ for G of $\mathrm{rw}(\sigma)$.

Theorem 1. *For all binary networks G, we have* $\mathrm{cw}(G) = \mathrm{rw}(G) + 1$.

In order to prove Theorem 1, we first need some definitions and intermediary observations. First, observe that swapping vertices u and v at position i and $i+1$ in an extension σ for G increases cw_i^σ by exactly $(\deg_G^-(v) - \deg_G^-(u)) + (\deg_G^+(u) - \deg_G^+(v))$, without changing cw_j^σ for any $j \neq i$. However, such a swap is only possible if u and v are incomparable in G as, otherwise, either σ or the result of the swap is not an extension of G.

Observation 1. *Let σ extend G and let u and v occur consecutively in σ.*
(a) Then, $\mathrm{cw}_v^\sigma = \mathrm{cw}_u^\sigma + \deg_G^-(v) - \deg_G^+(v)$.
(b) If $u \not<_G v$ and $\deg_G^-(v) - \deg_G^+(v) \leq \deg_G^-(u) - \deg_G^+(u)$ *then* $\mathrm{cw}(\pi) \leq \mathrm{cw}(\sigma)$ *where π results from swapping u with v in σ.*

For binary G, Observation 1(a) implies that $\mathrm{cw}_i^\sigma > \mathrm{cw}_{i+1}^\sigma$ if $\sigma(i+1)$ is a tree-vertex, and $\mathrm{cw}_i^\sigma < \mathrm{cw}_{i+1}^\sigma$, otherwise. Thus, the maximum cutwidth is attained at a reticulation or leaf followed by a tree-vertex.

Lemma 1. *Let i be a position of an extension σ for G such that* $\mathrm{cw}_i^\sigma = \mathrm{cw}(\sigma)$. *Then, $\sigma(i+1)$ is a tree-vertex and $\sigma(i)$ is not.*

Proof. If $u := \sigma(i)$ is a tree-vertex, then $\deg^-(u) < \deg^+(u)$ and, by Observation 1(a), $\mathrm{cw}_{i-1}^\sigma > \mathrm{cw}_i^\sigma$. If $v := \sigma(i+1)$ is a reticulation or a leaf, then $\deg^-(v) > \deg^+(v)$ and, by Observation 1(a), $\mathrm{cw}_i^\sigma < \mathrm{cw}_{i+1}^\sigma$. Both cases contradict $\mathrm{cw}_i^\sigma = \mathrm{cw}(\sigma)$. □

Observation 1(b) also implies that it suffices to consider stable orderings.

Lemma 2. *Every binary G has a stable extension σ with* $\mathrm{cw}(\sigma) = \mathrm{cw}(G)$.

Proof. Let σ be an extension of G with $\mathrm{cw}(\sigma) = \mathrm{cw}(G)$ and suppose that it is not stable. Then, there is a maximal (wrt. $\leq_\sigma$) milestone i such that $\sigma[1..i]$ contains a reticulation r whose parents are not in $\sigma[1..i]$. Since σ is G-respecting, $r \not<_G v$ for all $v \in \sigma[1..i]$. Since G is binary, r maximizes $\deg^-_G(r) - \deg^+_G(r)$ in G and, thus, we can use Observation 1(b) to move r to position i (by repeatedly swapping it with the next vertex in the order) without increasing the cutwidth of the ordering. Thus, σ can be transformed into a stable extension σ' with $\mathrm{cw}(\sigma') = \mathrm{cw}(\sigma) = \mathrm{cw}(G)$. □

Note that, for all $i \neq 1$, $\mathrm{RW}^\sigma_i \setminus \mathrm{RW}^\sigma_{i-1} = \{\sigma(i)\}$ and $\mathrm{RW}^\sigma_{i-1} \setminus \mathrm{RW}^\sigma_i$ contains only children of $\sigma(i)$ (not necessarily all of them).

Observation 2. $\mathrm{rw}^\sigma_i \geq \mathrm{rw}^\sigma_{i-1} - \deg^+_G(\sigma(i)) + 1$ *for all* $1 < i < |V(G)|$.

Lemma 3. *Let σ be an extension for G, let $u := \sigma(i)$ be a reticulation such that u is not a child of $v := \sigma(i+1)$. Let π be the order resulting from swapping u and v in σ. Then,* $\mathrm{rw}(\pi) \leq \mathrm{rw}(\sigma)$.

Proof. Observe that $\mathrm{rw}^\pi_j = \mathrm{rw}^\sigma_j$ for all $j \neq i$. For the sake of contradiction, suppose that $\mathrm{rw}^\pi_i > \mathrm{rw}^\sigma_i$ and $\mathrm{rw}^\pi_i = \mathrm{rw}(\pi)$. Let w be the unique child of u in G, note that $\mathrm{RW}^\pi_i \setminus \mathrm{RW}^\sigma_i \subseteq \{v, w\}$ and $\mathrm{RW}^\sigma_i \setminus \mathrm{RW}^\pi_i \supseteq \{u\}$. However, $\mathrm{rw}^\pi_i > \mathrm{rw}^\sigma_i$ (that is, $|\mathrm{RW}^\pi_i \setminus \mathrm{RW}^\sigma_i| > |\mathrm{RW}^\sigma_i \setminus \mathrm{RW}^\pi_i|$) implies $\mathrm{RW}^\pi_i \setminus \mathrm{RW}^\sigma_i = \{v, w\}$ and $\mathrm{RW}^\sigma_i \setminus \mathrm{RW}^\pi_i = \{u\}$ and, thus, $\mathrm{rw}^\pi_i = \mathrm{rw}^\sigma_i + 1$. This means that w is not a child of v and that each child z of v is in both RW^σ_i and RW^π_i, implying that z has another parent in $\sigma[i+1..] = \{v\} \cup \pi[i+2..]$. Thus, we know that each child of v has a parent in $\pi[i+2..] = \sigma[i+2..]$. But then, $\mathrm{RW}^\sigma_{i+1} = \mathrm{RW}^\sigma_i \cup \{v\}$, implying $\mathrm{rw}(\sigma) \geq \mathrm{rw}^\sigma_{i+1} = \mathrm{rw}^\sigma_i + 1 = \mathrm{rw}^\pi_i = \mathrm{rw}(\pi)$, which contradicts $\mathrm{rw}(\sigma) < \mathrm{rw}(\pi)$. □

Replacing Observation 1(b) by Lemma 3 in the proof of Lemma 2 gives an analogous result for the register width.

Corollary 1. *Every G has a stable extension σ with* $\mathrm{rw}(\sigma) = \mathrm{rw}(G)$.

We can now relate directed cut- and register width on binary networks.

Lemma 4. *Let G be a binary network and let σ be a stable extension of G. Then,* $\mathrm{cw}(\sigma) = \mathrm{rw}(\sigma) + 1$.

Proof. For any position j, let Y_j be the set of vertices in $\sigma[1..j]$ whose two parents are in $\sigma[j+1..]$ and note that each vertex in RW^σ_j has at least one incoming arc in CW^σ_j. As G is binary, $\mathrm{cw}^\sigma_j = \mathrm{rw}^\sigma_j + |Y_j|$ follows.

"$\leq$": Let i be any position such that $\mathrm{cw}^\sigma_i = \mathrm{cw}(\sigma)$. By Lemma 1, $v := \sigma(i+1)$ is a tree-vertex, implying that $i+1$ is a milestone. Since σ is stable, all reticulations $w <_\sigma v$ have at least one parent in $\sigma[1..i+1]$ and thus, $Y_{i+1} = \varnothing$, implying $\mathrm{cw}^\sigma_{i+1} = \mathrm{rw}^\sigma_{i+1}$. Now, since v has one incoming and two outgoing arcs, $\mathrm{cw}^\sigma_i = \mathrm{cw}^\sigma_{i+1} + 1 = \mathrm{rw}^\sigma_{i+1} + 1 \leq \mathrm{rw}(\sigma) + 1$.

"$\geq$": Towards a contradiction, assume that there are positions i for which $\mathrm{rw}^\sigma_i = \mathrm{rw}(\sigma)$ and $\mathrm{cw}^\sigma_i < \mathrm{rw}^\sigma_i + 1$ and let i be smallest among them. If $u := \sigma(i)$ is a leaf, then we have $\mathrm{rw}^\sigma_i = \mathrm{rw}^\sigma_{i-1} - 1$ and $\mathrm{cw}^\sigma_i = \mathrm{cw}^\sigma_{i-1} - 1$, contradicting the minimality of i. If u is a reticulation, then $\sigma(i) \in Y_i$, implying $\mathrm{cw}(\sigma) \geq \mathrm{cw}^\sigma_i \geq \mathrm{rw}^\sigma_i + 1 = \mathrm{rw}(\sigma) + 1$, contradicting $\mathrm{cw}^\sigma_i < \mathrm{rw}^\sigma_i + 1$. Thus, u is a tree-vertex and we let v and w be the children of u in G (recall that we disallow tree-vertices with a single child in binary networks).

Case 1: Neither v nor w has a parent in $\sigma[i+1,..]$ (that is, $v, w \notin Y_{i-1}$). Then, $\mathrm{RW}^\sigma_{i-1} = (\mathrm{RW}^\sigma_i \setminus \{u\}) \cup \{v, w\}$, implying $\mathrm{rw}^\sigma_{i-1} > \mathrm{rw}^\sigma_i$, which contradicts $\mathrm{rw}^\sigma_i = \mathrm{rw}(\sigma)$.

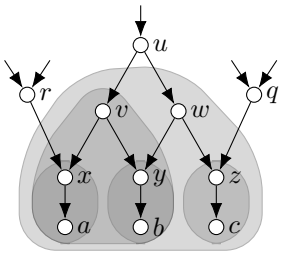
Case 2: Both v and w have a parent in $\sigma[i+1..]$ (that is, $v, w \in Y_{i-1}$, implying $\mathrm{cw}^\sigma_{i-1} \geq \mathrm{rw}^\sigma_{i-1} + 2$). Then $\mathrm{RW}^\sigma_i = \mathrm{RW}^\sigma_{i-1} \cup \{u\}$ and, thus, $\mathrm{cw}(\sigma) \geq \mathrm{cw}^\sigma_{i-1} \geq \mathrm{rw}^\sigma_{i-1} + 2 \geq \mathrm{rw}^\sigma_i + 1$.

Case 3: Exactly one of v and w (w.l.o.g. v) has a parent in $\sigma[i+1..]$, that is, $v \in Y_{i-1}$, implying $\mathrm{cw}^\sigma_{i-1} \geq \mathrm{rw}^\sigma_{i-1} + 1$. Then, $\mathrm{RW}^\sigma_i = (\mathrm{RW}^\sigma_{i-1} \setminus \{w\}) \cup \{u\}$ and, thus, $\mathrm{cw}(\sigma) \geq \mathrm{cw}^\sigma_{i-1} \geq \mathrm{rw}^\sigma_{i-1} + 1 = \mathrm{rw}^\sigma_i + 1$. □

Since, by Lemma 2 and Corollary 1 there are cutwidth-optimal and register width-optimal extensions of G that are stable, we conclude that $\mathrm{cw}(G) = \mathrm{rw}(G) + 1$, thus proving Theorem 1.

Scanwidth. Let σ be an extension for G and let $i \in \mathbb{N}$. We define SW^σ_i as the set of all arcs $uv \in \mathrm{CW}^\sigma_i$ for which v and $\sigma(i)$ are weakly connected in $G[\sigma[1..i]]$ (see Figure 3(left)). sw^σ_i is defined as $|\mathrm{SW}^\sigma_i|$, while the scanwidth of σ is $\mathrm{sw}(\sigma) := \max_i \mathrm{sw}^\sigma_i$ and the scanwidth of G, denoted by $\mathrm{sw}(G)$, is the minimum of $\mathrm{sw}(\sigma)$ over all extensions σ for G. Again, in our notations we allow i to be a vertex instead of a position, as σ is a bijection between the two.

Alternatively, $\mathrm{sw}(G)$ can be defined as follows. For a tree extension Γ for G, we define GW^Γ_v as the set of arcs $(x, y) \in E(G)$ with $x >_\Gamma v \geq_\Gamma y$. Further, we let $\gamma\mathrm{w}(\Gamma) := \max_v |\mathrm{GW}^\Gamma_v|$ and $\gamma\mathrm{w}(G) := \min_\Gamma \gamma\mathrm{w}(\Gamma)$. Although a tree extension is defined independently of a (full) extension for G, there is a link between the two notions. Indeed, the sets GW^Γ_v in an optimal tree extension correspond to the sets SW^σ_v in one or several optimal extensions σ (see Figure 3).



$\sigma(i)$	$\mathrm{SW}_i^\sigma = \mathrm{GW}_{\sigma(i)}^\Gamma$
a	$\{xa\}$
b	$\{yb\}$
c	$\{zc\}$
x	$\{rx, vx\}$
y	$\{vy, wy\}$
z	$\{wz, qz\}$
v	$\{rx, uv, wy\}$
w	$\{rx, uv, uw, qz\}$
$\dots$	$\dots$

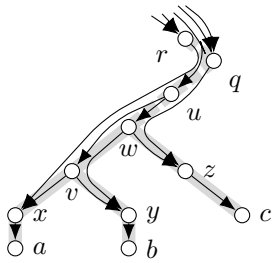


Figure 3: Illustration of the two definitions of scanwidth. **Left:** Lower part of a graph G where gray zones represent the weakly connected components induced by $\sigma[1..i]$ for $\sigma = (a, b, c, x, y, z, v, w, \dots)$ and $i \leq 8$. Here, $\mathrm{SW}_v^\sigma = \{rx, uv, wy\}$ since x, y, and v are weakly connected in $G[a, b, c, x, y, z, v]$. **Middle:** table indicating SW_i^σ for $i \leq 8$ corresponding to σ. **Right:** part of a tree Γ with $\mathrm{GW}_{\sigma(i)}^\Gamma = \mathrm{SW}_i^\sigma$ for all $i \leq 8$. For the extension $\pi = (c, a, z, b, y, x, v, w, \dots)$, we also have $\mathrm{GW}_{\pi(i)}^\Gamma = \mathrm{SW}_i^\pi$ for all $i \leq 8$.

Proposition 1. *Let G be a network. Then, (a) $\gamma\mathrm{w}(G) = \mathrm{sw}(G)$, (b) if G has only one leaf, then $\mathrm{sw}(G) = \mathrm{cw}(G)$. (c) If G is also binary, then $\mathrm{sw}(G) = \mathrm{rw}(G) + 1$.*

To prove the equivalence of the two definitions of the scanwidth, we first define an auxiliary graph associated to a network and then show some results linking the structure of both graphs.

Definition 1. *Given an extension σ for a network G, we define Γ^σ as the transitive reduction of*

$$(V(G), \{vu \mid u <_\sigma v \quad \wedge \quad u \text{ and } v \text{ are weakly connected in } G[\sigma[1..v]]\})$$

and we call Γ^σ canonical *for σ.*

Note that Γ^σ may contain arcs not present in G (cf. the tree in Figure 3 which is in fact canonical and it contains $wv \notin E(G)$). It turns out that Γ^σ has some convenient properties.

Lemma 5. *Let σ extend G and let Γ^σ be canonical for σ. Then,*
(a) For any $u, v \in V(G)$, we have $u \leq_\Gamma^\sigma v$ if and only if u and v are weakly connected in $G[\sigma[1..v]]$,
(b) each $u \in V(G)$ has a path in Γ^σ from the last vertex of σ (implying that Γ^σ is weakly connected),
(c) each $u \in V(G)$ is a leaf of G if and only if it is a leaf of Γ^σ,
(d) σ is an extension of Γ^σ,
(e) Γ^σ is a tree extension for G,
(f) For any $v \in V(G)$, the set $C := \{u \mid u \leq_\Gamma^\sigma v\}$ forms a weakly connected component in $G - \mathrm{GW}_v^\Gamma$.
(g) for each child v of any vertex u in Γ^σ there is a child w of u in G with $w \leq_\Gamma^\sigma v$, and
(h) for each vertex u, $\deg_{\Gamma^\sigma}^+(u) \leq \deg_G^+(u)$.
(i) for each extension π of Γ^σ and each $v \in V(G)$, it holds that $\mathrm{SW}_v^\pi = \mathrm{GW}_v^{\Gamma^\sigma}$.

Proof. (a) follows from the definition of Γ^σ and the fact that being weakly connected is a transitive property.

(b): Since G is weakly connected, the last vertex of σ is weakly connected to all vertices in $G[\sigma] = G$.

(c) Let $u \in V(G)$. Since σ is an extension of G, we know that $\sigma[1..u]$ contains all descendants and no ancestors of u. Thus, u is a leaf of G if and only if $\sigma[1..u]$ contains no vertex adjacent to u in G and, this holds if and only if u is not weakly connected to any vertex in $\sigma[1..u]$ which, by (a) is equivalent to u being a leaf of Γ^σ.

(d): Directly follows from the definition of Γ^σ, since $u <_\sigma v$ for all arcs vu of Γ^σ.

(e): First, to prove that Γ^σ respects G, consider any arc vu of G. Since σ respects G, we have $u <_\sigma v$. Further, as u and v are weakly connected in $G[\sigma[1..v]]$, by (a) we have $u <_\Gamma^\sigma v$.

Second, by (b) and (d), we know that Γ^σ is weakly connected and acyclic. It remains to show that Γ^σ is a tree. Towards a contradiction, assume that there is a reticulation r in Γ^σ with parents x and y. Without loss of generality, let $x <_\sigma y$. Then, x and y are weakly connected to r in $G[\sigma[1..y]]$ and, by transitivity of weak connectivity, to each other. By (a), this implies $x \leq_\Gamma^\sigma y$. But then, Γ^σ contains a directed path from y to r via x as well as the arc yr, contradicting Γ^σ being a transitive reduction.

(f): As, by (a), C is weakly connected in $G - \mathrm{GW}_v^{\Gamma^\sigma}$, it suffices to show that no node $x \notin C$ is weakly connected to C in $G - \mathrm{GW}_v^{\Gamma^\sigma}$. To this end, let $x \notin C$ (that is, $x \not\leq_\Gamma^\sigma v$) and assume towards a contradiction that the undirected graph underlying G contains an x-v-path p avoiding $\mathrm{GW}_v^{\Gamma^\sigma}$. Then, there is a first

vertex z with $z \leq^\sigma_\Gamma v$ on p. Let y be its predecessor on p and note that yz or zy is an edge of G. Since, by (e), Γ^σ extends G but $y \not\leq^\sigma_\Gamma v$ by minimality of z, we know that G contains the edge yz. Again, by (e), we have $y \leq^\sigma_\Gamma v <^\sigma_\Gamma z$, implying that $yz \in \mathrm{GW}_v^{\Gamma^\sigma}$, which contradicts p surviving in $G - \mathrm{GW}_v^{\Gamma^\sigma}$.

(g): By definition of Γ^σ, we know that u and v are weakly connected in $G[\sigma[1..u]]$, that is, there is a u-v-path p in the undirected graph underlying $G[\sigma[1..u]]$. As no parent of u is in $\sigma[1..u]$, the vertex w following u on p is a child of u in G. Let x be the maximum (wrt. σ) of $V(p) - u$ and note that, by (d), $u \not\leq^\sigma_\Gamma x$. Further, $G[\sigma[1..x]]$ contains the w-v-subpath of p which contains an x-v-subpath. Thus, v and x are weakly connected in $G[\sigma[1..x]]$, implying $v \leq^\sigma_\Gamma x$ by (a). Now, if $v <^\sigma_\Gamma x$, then $u \leq^\sigma_\Gamma x$ since u is the parent of v in Γ^σ, contradicting $u \not\leq^\sigma_\Gamma x$. Thus, $x = v$, implying that v and w are weakly connected in $G[\sigma[1..v]]$ and, by (a), $w \leq^\sigma_\Gamma v$.

(h): Follows directly from (g) and the fact that Γ^σ is a tree (see (e)).

(i): First, consider some $xy \in \mathrm{GW}_v^{\Gamma^\sigma}$, implying, $y \leq^\sigma_\Gamma v <^\sigma_\Gamma x$. Since π extends Γ^σ, we have $y \leq_\pi v <_\pi x$, implying $xy \in \mathrm{CW}_v^\pi$. By (a), y and v are weakly connected in $G[\sigma[1..v]]$, so the undirected graph underlying $G[\sigma[1..v]]$ contains an undirected v-y-path p. Clearly, each vertex z on p is weakly connected to v in $G[\sigma[1..v]]$, so $z \leq_{\Gamma^\sigma} v$ by (a) and, thus, $z \leq_\pi v$ since π extends Γ^σ. Thus, p also exists in $G[\pi[1..v]]$, implying $xy \in \mathrm{SW}_v^\pi$. Second, consider some $xy \in \mathrm{SW}_v^\pi$, implying that $xy \in \mathrm{CW}_v^\pi$ and y and v are weakly connected in $G[\pi[1..v]]$. Let p denote an undirected v-y-path in the undirected graph underlying $G[\pi[1..v]]$. Towards a contradiction, assume that there is a first vertex w on p with $w \not\leq_{\Gamma^\sigma} v$ and let u precede w on p (that is, $u \leq^\sigma_\Gamma v$). Thus, G contains either the arc uw or the arc wu and, by (e), either $w <_{\Gamma^\sigma} u$ or $u <_{\Gamma^\sigma} w$. However, since $u \leq_{\Gamma^\sigma} v$, but $w \not\leq_{\Gamma^\sigma} v$, we know that $v <_{\Gamma^\sigma} w$. Since π is an extension of Γ^σ, we also have $v <_\pi w$, contradicting that p lives in $G[\pi[1..v]]$. Thus, all vertices of p are descendants of v in Γ^σ, in particular $y \leq^\sigma_\Gamma v$. To show that $xy \in \mathrm{GW}_v^{\Gamma^\sigma}$, it remains to show that $v <^\sigma_\Gamma x$. Clearly, $x \not\leq^\sigma_\Gamma v$ since, otherwise, $x \leq_\pi v$ (since π extends Γ^σ), contradicting $xy \in \mathrm{SW}_v^\pi$. However, $y <^\sigma_\Gamma x$ since $y <_G x$ and, by Item e, Γ^σ is a tree extension for G. Thus, $v <^\sigma_\Gamma x$, implying $xy \in \mathrm{GW}_v^{\Gamma^\sigma}$. □

Proof of Proposition 1. (a): "≥": Let Γ be an extension tree for G with $\gamma\mathrm{w}(\Gamma) = \gamma\mathrm{w}(G)$ and let σ be any extension for Γ. Towards a contradiction, assume that $\gamma\mathrm{w}(\Gamma) < \mathrm{sw}(\sigma)$, that is, there is some $v \in V(G)$ with $\gamma\mathrm{w}(\Gamma) < \mathrm{sw}^\sigma_v$. In particular, $|\,\mathrm{GW}_v^\Gamma\,| < |\,\mathrm{SW}_v^\sigma\,|$, implying that there is an arc $xy \in \mathrm{SW}_v^\sigma \setminus \mathrm{GW}_v^\Gamma$. Since $xy \in \mathrm{CW}_v^\sigma$, we have $y \leq_\sigma v <_\sigma x$ and, as σ is an extension of Γ, we have $y \not>_\Gamma v \not\geq_\Gamma x$. Further, as $y <_G x$ and Γ is an extension tree for G, we have $y <_\Gamma x$. Then, v and y are incomparable in Γ since, otherwise, $y \leq_\Gamma v$, implying $y \leq_\Gamma v <_\Gamma x$ (since $y <_\Gamma x$, $x \not\leq_\Gamma v$ and Γ is a tree), which contradicts $xy \notin \mathrm{GW}_v^\Gamma$. Since $xy \in \mathrm{SW}_v^\sigma$, we know that y and v are weakly connected in $G[\sigma[1..v]]$, that is, there is a v-y-path p in the undirected graph underlying $G[\sigma[1..v]]$. For all j, let p_j be the j^{th} vertex of p. Since $p_0 = v \leq_\Gamma v$, but $y \not\leq_\Gamma v$, there is some j with $p_j \leq_\Gamma v$ but $p_{j+1} \not\leq_\Gamma v$. Moreover, Γ extends G and G contains an arc between p_j and p_{j+1}, we know that p_j and p_{j+1} are not incomparable in Γ. But then, as $p_j \leq_\Gamma v$ and $p_{j+1} \not\leq_\Gamma v$, we have $p_j <_\Gamma p_{j+1}$ and $v <_\Gamma p_{j+1}$. As σ extends Γ, we have $v <_\sigma p_{j+1}$, contradicting $p_{j+1} \in \sigma[1..v]$. Thus $\gamma\mathrm{w}(G) = \gamma\mathrm{w}(\Gamma) \geq \mathrm{sw}(\sigma) \geq \mathrm{sw}(G)$.

"≤": Let σ be an extension for G with $\mathrm{sw}(\sigma) = \mathrm{sw}(G)$ and let Γ^σ be its canonical extension tree. Then, by Lemma 5(e) and (i), $\gamma\mathrm{w}(G) \leq \gamma\mathrm{w}(\Gamma^\sigma) = \mathrm{sw}(\sigma) = \mathrm{sw}(G)$.

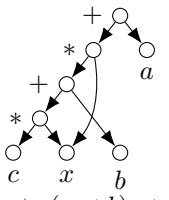
(b): This holds since all vertices in $\sigma[1..i]$ are weakly connected in $G[\sigma[1..i]]$ for all i.

(c): This follows from Theorem 1. □

Observe that the scanwidth differs largely from the directed path-width [1], which is always zero for DAGs. To relate the scanwidth to established parameters, let us mention that the scanwidth of any level-k network cannot exceed $k+1$ but it might even be constant. Regarding width-measures, the scanwidth is bounded by the cutwidth from below and the treewidth (of the underlying undirected graph) from above.

3 NP-completeness

To compute the value of a given algebraic expression such as $(cx + b)x + a$ using a computer, we need to store the values of a, b, c, and x in registers which can then be processed by the CPU. As registers can be overwritten, expressions involving more variables than the number of available registers can be evaluated. The problem of deciding whether a given expression can be evaluated on a CPU with k registers (without recomputing sub-expressions or relying on the costly *spilling* technique) is called Register Sufficiency. We suppose that the input expression is given as a **rooted** DAG of necessary computations. For example, to compute $(cx+b)x+a$, we need to compute $cx + b$, for which we need to compute cx (see figure on the right).



Register Sufficiency [PO1 in [7]] (RS)

Input: a rooted DAG G of an expression to be computed, $k \in \mathbb{N}$
Question: Can G be computed using at most k registers?

Register Sufficiency can be interpreted as a game played on G, where the player has k stones that have to be placed progressively on *all* vertices, using the following operations [13]:

1. remove a stone from any vertex
2. for a vertex p whose every child contains a stone,
   2a. place an available stone on p or
   2b. move a stone from a child of p to p,

so that each vertex receives a stone exactly *once* during these operations.

Stones represent registers and putting a stone on a vertex of the graph corresponds to computing the vertex and storing the result in that register (this is why we need stones on all children of a vertex when computing it). Removing a stone from a vertex corresponds to forgetting the value of the vertex, which should then be done only if we do not need it in other computations (as vertices cannot be recomputed), i.e. when all its parent vertices have already received a stone.

Winning the game means successfully computing the algebraic expression encoded in the graph while using at most k registers. In this context, an extension σ for a graph G indicates in which order the vertices receive stones. Note that the first stone enters G via applying Rule 2a to a leaf of G. Then, solving the optimization problem associated to Register Sufficiency can be seen as finding an extension of G that minimizes the number k of stones (registers) needed to win the game (compute the expression). As suggested by our formulation, this number equals the previously introduced "register width", $\mathrm{rw}(G)$.

Proposition 2. *A DAG G can be computed using $\leq k$ registers if and only if $\mathrm{rw}(G) \leq k$.*

Proof of Proposition 2. "⇒": Let π' be a sequence of moves in the Register Sufficiency game using at most k stones, let π be the result of removing all moves of type 1 from π', and let σ be the sequence of vertices that receive a stone in a move of type 2, in order. Since each vertex receives a stone exactly once, we know that σ is a permutation of $V(G)$. First, note that, by Rule 2 and transitivity of $\leq_\sigma$, we have $u \leq_\sigma v \Rightarrow u \leq_G v$, implying that σ extends G. Second, consider an index j and indices i, and ℓ such that $i \leq j < \ell$ and $\sigma(i)$ is a child of $\sigma(\ell)$ in G. Then, by Rule 2, $\sigma(i)$ keeps its stone at least until $\sigma(\ell)$ receives a stone. Thus, at step j (right after $\sigma(j)$ receives a stone), there is a stone on $\sigma(i)$. Since this holds for all such triples, we conclude that each vertex in $\sigma[1..j]$ with a parent in $\sigma[j+1..]$ holds a stone at step j for all j. By definition, this number is rw^σ_j.

"⇐": Let σ be an extension of G such that $\mathrm{rw}(\sigma) \leq k$. Let f be a function mapping each index $j \in [|V(G)|]$ to a move in the Register Sufficiency game as follows: If $\sigma(j)$ has a child v in G such that $\sigma[j+1..]$ does not contain a parent of v, then move the stone on v to $\sigma(j)$ (Rule 2b), otherwise, introduce a new stone to $\sigma(j)$ (Rule 2a). Then, construct a sequence π of moves by, for each $j \in [|V(G)|]$, first executing $f(j)$ and then removing all stones from vertices u whose every parent has a stone (Rule 1). Note that each vertex receives a stone exactly once during the execution of π since σ is a permutation of $V(G)$. Now, assume towards a contradiction that a move of π is not valid, that is, there is an index j such that $f(j)$ puts/moves a stone to a vertex $\sigma(j)$ who has a child $\sigma(i)$ without a stone at that point. Since σ is an extension of G, we know $i <_\sigma j$, implying that $\sigma(i)$ received a stone before step j. Thus, there is some ℓ with $i < \ell < j$ such that $\sigma(i)$ lost its stone between the execution of $f(\ell)$ and $f(\ell+1)$ in π. But this happens only if all parents of $\sigma(i)$ have a stone at step ℓ which, since $\sigma(j)$ is such a parent, contradicts the fact that each vertex receives a stone exactly once in π. □

With Proposition 2, the Register Sufficiency problem can be formulated as: given a rooted DAG G

and some integer k, decide if $\mathrm{rw}(G) \leq k$. Following Sethi [13], we will use a special, "initial" vertex in our reduction.

Definition 2. *Let (G, k) be an instance of* Register Sufficiency *such that G has k leaves and all leaves have a common parent ψ. Then, we say that ψ is an* initial vertex *and that (G, k)* has the initial vertex property*.*

Lemma 6 (See [13])**.** *Let (G, k) be a yes-instance of* Register Sufficiency *with an initial vertex ψ. Let σ be an extension of G with $\mathrm{rw}(\sigma) \leq k$. Then, $\sigma(k+1) = \psi$ and $\sigma[1..k]$ contains the k leaves of G in any order. Moreover, there is a leaf whose only parent is ψ.*

Proof of Lemma 6. Let $i := \sigma^{-1}(\psi)$. Let L be the set of all k leaves of G and recall that ψ is parent of all of them, implying $\mathrm{RW}^\sigma_{i-1} \supseteq L$. Since $\mathrm{rw}^\sigma_{i-1} \leq \mathrm{rw}(\sigma) \leq k = |L|$, we know that $\mathrm{RW}^\sigma_{i-1} = L$ and $i - 1 \geq k$. To show that $i - 1 \leq k$, assume that $\sigma[1..i-1]$ contains non-leaf vertices of G and let u be maximal (wrt. $<_\sigma$) among them. Let p be a parent of u. By maximality of u, $p \notin \sigma[1..i-1]$. But then, $u \in \mathrm{RW}^\sigma_{i-1}$, contradicting $\mathrm{RW}^\sigma_{i-1} = L$ (by definition $p \notin L$). Thus, $\sigma[1..i-1] = L$, implying $i - 1 \leq k$ and, thus, $i = k+1$. Further, since $\mathrm{RW}^\sigma_j \subseteq L$ for all $j \leq k$, choosing any order among the leaves in $\sigma[1..k]$ preserves rw^σ_j (and, thus, $\mathrm{rw}(\sigma)$). Finally, if all leaves have a second parent, then $|\mathrm{RW}^\sigma_i| = |L \cup \{\psi\}| = k + 1$ contradicting $\mathrm{rw}(\sigma) \leq k$ (unless ψ is the root of G; but then, as $\sigma^{-1}(\psi) = k + 1$, G contains only leaves and ψ, and the claim trivially holds). □

A corollary of Lemma 6 is that, in a yes-instance (G, k) with the initial vertex property, all children of the initial vertex are leaves. Thus, $\mathrm{rw}^\sigma_k = k$ for all extensions σ with $\mathrm{rw}(\sigma) \leq k$ and, thus, $\mathrm{rw}(G) = k$ for such yes-instances. Note that 3-SAT reduces to instances (G, k) of Register Sufficiency that have the initial vertex property [13].

Theorem 2 ([13])**.** *It is* NP*-hard to decide* Register Sufficiency *for instances (G, k) that have the initial vertex property.*

Below, we reduce Register Sufficiency on instances with the initial vertex property to Register Sufficiency on rooted, binary, single-leaf DAGs. To this end, we reduce from Weighted 2-Satisfiability instead of 3-Satisfiability and modify parts of the reduction in order to obtain a network that is already bifurcating in some crucial spots. Then, we present a number of polynomial-time executable transformation rules that take one such instance (G, k) of Register Sufficiency having the initial vertex property and replace all remaining high-degree vertices with binary ones without changing the answer for the instance. Finally, a reduction rule is given to ensure that the resulting DAG has a single leaf.

3.1 An Adaptation of a known NP-hardness Proof

As we will modify the construction of Sethi [13], we first quickly restate it here.

Construction 1 ([13])**.** *Given a formula φ in 3-CNF on variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, let $y_{i,j}$ denote the j^{th} literal in C_i. Construct the instance (G, k^*), where $k^* = 6n + 2m + 1$ and G is a DAG on the vertex set $A \uplus B \uplus C \uplus F \uplus M \uplus R \uplus S \uplus T \uplus U \uplus W \uplus X \uplus Z$ where $R = \bigcup_{i\in[n]} R^i$, $S = \bigcup_{i\in[n]} S^i$, $T = \bigcup_{i\in[n]} T^i$ and*

$$
\begin{aligned}
A &= \{a_i \mid i \in [2n+1]\} & C &= \{c_i \mid i \in [m]\} & F &= \{f_{i,1}, f_{i,2}, f_{i,3} \mid i \in [m]\}\\
B &= \{b_i \mid i \in [2n-m]\} & M &= \{\psi, d, \rho\} & R^i &= \{r_{i,j} \mid\ j \in [2n-2i+2]\}\\
U &= \{u_{i,1}, u_{i,2} \mid i \in [n]\} & W &= \{w_i \mid i \in [n]\} & S^i &= \{s_{i,j} \mid\ j \in [2n-2i+1]\}\\
X &= \{x_i, \overline{x}_i \mid i \in [n]\} & Z &= \{z_i \mid i \in [n]\} & T^i &= \{t_{i,j} \mid\ j \in [2n-2i+1]\}
\end{aligned}
$$

and the arc set $\bigcup_{i\in[10]} E_i$ where

$$
\begin{aligned}
E_1 &= \{\psi v \mid v \in A \uplus B \uplus F \uplus U\} & E_{7,1} &= \{z_{i+1}w_i, z_{i+1}z_i \mid i \in [n-1]\}\\
E_2 &= \{v\psi \mid v \in C \uplus R \uplus S \uplus T \uplus W\} & E_{7,2} &= \{c_i w_n, c_i z_n \mid i \in [m]\}\\
E_3 &= \{\rho v \mid v \in W \uplus X \uplus Z \uplus \{\psi, d\}\} & E_8 &= \{c_i f_{i,j} \mid i \in [m],\ j \in [3]\}\\
E_4 &= \{x_i z_i, \overline{x}_i z_i, x_i u_{i,1}, \overline{x}_i u_{i,2} \mid i \in [n]\} & E_9 &= \{dv \mid v \in B \uplus C\}\\
E_5 &= \{w_i u_{i,1}, w_i u_{i,2} \mid i \in [n]\} & E_{10} &= \{y_{i,j} f_{i,j}, \overline{y}_{i,j} f_{i,\ell} \mid i \in [m],\ j \in [3],\ j < \ell \leq 3\}.\\
E_6 &= \{z_i r_{i,j}, x_i s_{i,j}, \overline{x}_i t_{i,j} \mid i \in [n],\ r_{i,j} \in R^i,\ s_{i,j} \in S^i,\ t_{i,j} \in T^i\}
\end{aligned}
$$

where $y_{i,j} \in X$ is the j^{th} literal appearing in clause c_i, and $\overline{y}_{i,j} \in X$ is its negation.

Sethi [13] makes several observations on Construction 1 and on any extension σ for G with $\mathrm{rw}(\sigma) = k$. First, G has exactly k leaves (the vertices in $A \uplus B \uplus F \uplus U$) which are all children of ψ and, thus, $\mathrm{rw}(G) \geq k$, implying that σ is optimal. Second, by Definition 2, ψ is initial for (G, k) and, by Lemma 6, σ starts with these leaves (in any order), followed by ψ. Third, the vertices w_i appear in order (that is, $w_i <_\sigma w_j$ iff $i < j$), as well as the vertices z_i. Further, the true literals in X appear in order of their index. Finally, Sethi [13] shows that $\mathrm{RW}_d^\sigma \cap X$ contains exactly one of x_i and $\overline{x}_i$ for each i and that the corresponding literals form a satisfying assignment for φ.

For our result, we will strengthen Construction 1 to construct a *binary* DAG with a *single leaf* and *without degree-two vertices*. The hardest part to make G binary is to deal with the polytomies at vertices in C. Apart from being parent of ψ, each of these vertices is parent of w_n, z_n (see $E_{7,2}$ in Construction 1) and of three vertices of F (see E_8). To remedy this, we will instead reduce from an NP-hard 2-SAT variant called Monotone Weighted 2-Satisfiability (also known as Vertex Cover). In this variant, all variables occur non-negated in the instance formula φ, each variable is used at least once, and we ask for an assignment that satisfies φ while setting at most k variables to true. Our modifications to Construction 1 come in two stages. First, we modify it to work for Monotone Weighted 2-Satisfiability and replace the degree-two vertices in $R \uplus S \uplus T$, then we show how to "binarize" all remaining polytomies and establish a single leaf.

Construction 2 (See Figure 4)**.** *Given an integer k and a formula φ in monotone 2-CNF on variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, construct the DAG G by following Construction 1 and make the following modifications:*

1. *remove $f_{i,3}$ from G for all $i \in [m]$, thus obtaining the set $F' \subset F$,*
2. *add k new leaves a_i to A, thus forming A',*
3. *add n new leaves b_i to B, thus forming B', and replace E_9 by $E'_9 = \{dv \mid v \in B' \uplus C\}$,*
4. *add new leaves $H := \{h_{i,1}, h_{i,2} \mid i \in [m]\}$,*
5. *add k new vertices to each R^i, S^i, and T^i and modify E_6 accordingly, forming R'^i, S'^i, T'^i and E'_6,*
6. *for each $i \in [n]$, turn R^i into a path by adding the arcs $r_{i,j}r_{i,j+1}$ for each $j \in [2n - 2i + k + 1]$, and do the same with S^i and T^i,*
7. *replace E_2 by $E'_2 := \{v\psi \mid v \in R'^i \uplus S'^i \uplus T'^i\}$,*
8. *add new vertices $X' := \{x'_i \mid i \in [n]\}$ with arcs $E_{11} = \{x'_i x_i \mid i \in [n]\}$,*
9. *add a new vertex z_{n+1} to Z, forming Z', and add arcs $z_{n+1}z_n$ and $z_{n+1}w_n$ to $E_{7,1}$, forming $E'_{7,1}$ and replace $E_{7,2}$ by $E'_{7,2} := \{c_i z_{n+1} \mid i \in [m]\}$,*
10. *add a new leaf α as child of z_{n+1},*
11. *for all new leaves v, add ψv to E_1, forming E'_1,*
12. *add new vertices $P := \{p_i \mid i \in [m]\}$,*
13. *add new vertices $P' := \{p'_i, p''_i, p'''_i \mid i \in [m]\}$,*
14. *replace E_8 by $E'_8 := \{c_i p_i, p_i f_{i,1}, p_i f_{i,2}, p'''_i f_{i,2}, p'''_i p''_i, p''_i p'_i, p''_i h_{i,2}, p'_i h_{i,1} \mid i \in [m]\}$,*
15. *add to E_3 all arcs ρv with $v \in X' \cup \{p'_i, p'''_i \mid i \in [m]\} \cup \{z_{n+1}\}$, forming E'_3, and*
16. *letting $x_{i,j}$ denote the j^{th} variable in C_i (and its corresponding vertex with the same name), letting $\overline{x}_{i,j}$ denote the negation of $x_{i,j}$ (and its corresponding vertex with the same name) and letting $x'_{i,j}$ denote the vertex in X' such that $x'_{i,j}x_{i,j} \in E_{11}$, replace E_{10} by $E'_{10} = \{x'_{i,1}f_{i,1}, x'_{i,2}h_{i,2}, \overline{x}_{i,1}h_{i,1} \mid i \in [m]\}$.*

Finally, return the instance (G, k') with $k' = |A'| + |B'| + |F'| + |U| + |H| + |\{\alpha\}| = 7n + 3m + k + 2$.

The idea behind Construction 2 is that the "variable-assignment phase" of Sethi [13] still works as before (with k more stones in each step to account for the k additional vertices in $R'^i \setminus R^i$, $S'^i \setminus S^i$, and $T'^i \setminus T^i$). Note that Sethi's observations to explain how the graph has to be processed still hold, up until the moment w_n is reached. In particular, if we computed both x_i and $\overline{x}_i$, then we would not have enough stones to complete S'^{i+1} or T'^{i+1}, and we would thus be stuck. The same holds if we computed both x_i and x'_i: not enough stones would remain to complete S'^{i+1} or T'^{i+1}. In more detail, this process is as follows: in the beginning, all k' stones have to go to all the leaves, at which point ψ is computed using one stone of a vertex in A', while the other $k + 2n$ stones of A' are now free (unlike stones on other leaves still having other parents). These $k + 2n$ stones need to go to R'^1 (otherwise we will not have enough stones later for this vertex), allowing to compute z_1, who will keep one stone. The $k + 2n - 1$ other stones from R'^1 are free to go to either S'^1 allowing to compute x_1 or to T'^1 allowing to compute $\overline{x}_1$. The chosen literal allows exactly one stone from U to move to w_1 (e.g. $u_{1,1}$ if x_1 is chosen and $u_{1,2}$, otherwise), who will keep this stone. Thus, $k + 2n - 2$ stones (from either S'^1 or T'^1) are now free to compute R'^2, followed by z_2. This process continues until w_n receives a stone, at which point we ended Sethi's variable assignment phase. Now, the stone on α moves to z_{n+1} and we are left with the k free stones, coming

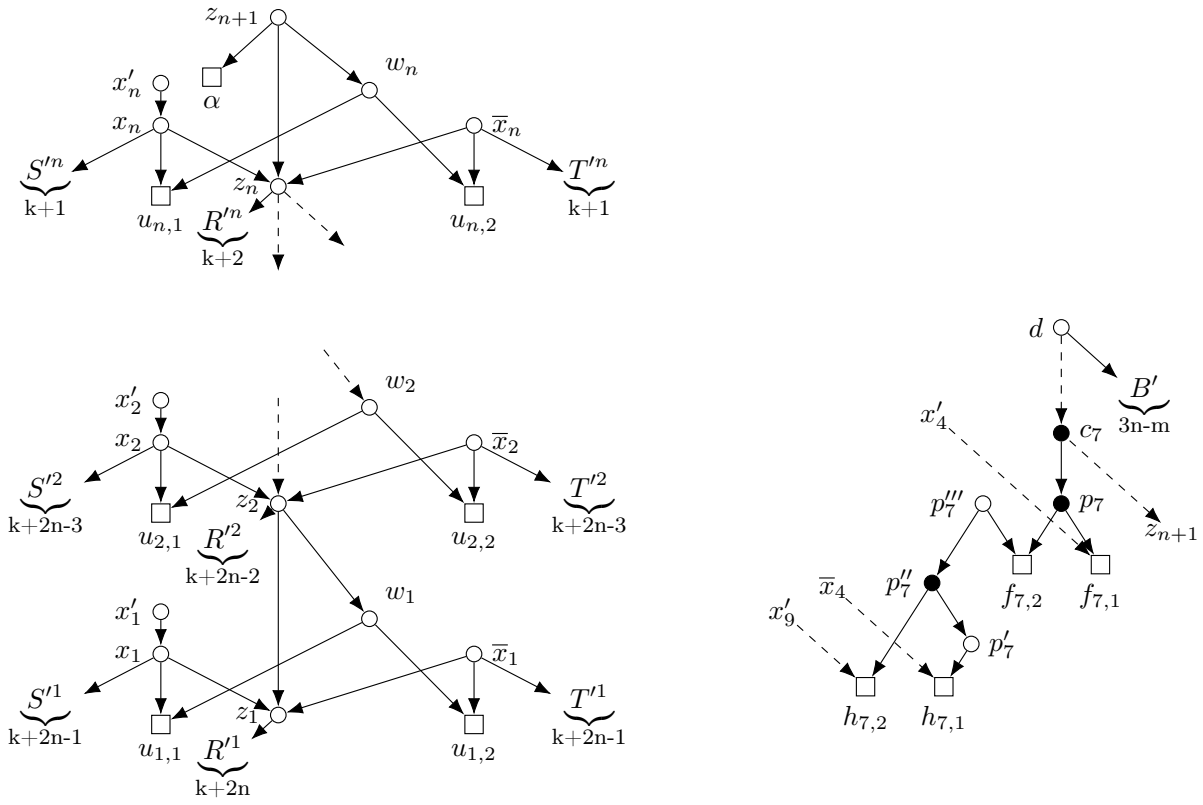


Figure 4: Illustration of Construction 2; white circle vertices are children of the root ρ, gray triangles are leaves and children of ψ. This allows us to omit drawing ψ and ρ. We also omit the leaves in A'. **Left:** "variable-assignment" gadget (arcs of E'_2 omitted). **Right:** clause gadget for the clause $C_7 = (x_4 \vee x_9)$. Note that all w_i, p_i and c_i are bifurcating.

from either S'^n or T'^n, that we can spend on vertices $x'_j \in X'$ whose corresponding $x_j \in X$ has received a stone before.

Consider what happens if the described "variable assignment" phase chooses k vertices in X satisfying the formula and the k corresponding vertices of X^* receive a stone right after this phase. Consider the gadget corresponding to clause $C_i = (x_j \vee x_\ell)$ and recall that $f_{i,1}$, $f_{i,2}$, $h_{i,1}$, $h_{i,2}$ already hold stones. In analogy with Sethi [13], each c_i receives a stone as follows:

- If C_i is satisfied by x_j, then x_j^* holds a stone, so the stone on $f_{i,1}$ can move to p_i (this is allowed since p_i's children all hold stones).
- Otherwise, both $\overline{x}_j$ and x_ℓ^* hold stones. The first one allows the stone on $h_{i,1}$ to move to p'_i. The second one allows the stone on $h_{i,2}$ to move to p''_i and then to p'''_i, allowing the stone on $f_{i,2}$ to move to p_i.

Thus, in both cases p_i gets a stone which it then passes to c_i. Finally, when all c_i have received a stone, d receives a stone from one of them, freeing up $|B'| = 3n - m$ stones on the vertices in B' and $|C| - 1 = m - 1$ stones on the vertices in C. Since $k \leq n$, these $3n - 1$ stones can then be placed on T'^1 (if x_1 already holds a stone) or S'^1 (if $\overline{x}_1$ already holds a stone) and one of them can then move to $\overline{x}_1$ or x_1, respectively. In this way, all $2n$ vertices x_i and $\overline{x}_i$ progressively receive a stone. Since n of them already got stones in the variable assignment phase, this leaves us with $(3n-1)-n$ stones, $n-k$ of which are then put on the $n-k$ remaining stoneless vertices of X' which immediately move to the remaining vertices of X^*. At this point, all stones on all $h_{i,1}$ and $h_{i,2}$ move to p'_i and p''_i followed by p'''_i if they did not already do so before. Finally, ρ receives a stone from any of its children.

In order to prove Construction 2 correct, we observe that all leaves of G are children of ψ and there are k' many of them.

Observation 3. *ψ is initial for (G, k').*

In the following, we call a path $(w, \ldots, \ell)$ in G *critical* for a vertex v if $v <_G w$ and ℓ is a leaf of G. We often make use of the fact that, for each extension σ, and each vertex v, RW_v^σ intersects each critical path of v (but not in its first vertex). We omit mentioning v if its clear from context.

Theorem 3. *Let $k \in \mathbb{N}$, let φ be a formula in monotone 2-CNF, and let (G, k') be an instance of* Register Sufficiency *constructed by Construction 2 on input (φ, k). Then, φ has a satisfying assignment with $\leq k$ true variables if and only if $\operatorname{rw}(G) \leq k'$.*

Proof of Theorem 3. "⇒": Let Q be a size-k set of variables of φ that intersects all clauses. Let $\sigma_1 := (A', B', F', U, H, \alpha, \psi)$. Let $\sigma_{2,i} := (r_{i,1}, r_{i,2}, \ldots, r_{i,|R^i|}, z_i) \circ Y_i \circ (q_i, w_i)$ for all $i \in [n]$, where $Y_i = (s_{i,1}, \ldots, s_{i,|S^i|})$ and $q_i = x_i$ whenever $x_i \in Q$ and otherwise $Y_i = (t_{i,1}, \ldots, t_{i,|T^i|})$ with $q_i = \overline{x}_i$. Abbreviate $\sigma_2 := \sigma_{2,1} \circ \sigma_{2,2} \circ \ldots \circ \sigma_{2,n}$. Let $X'_Q := \{x'_i \mid x_i \in Q\}$, let $X^*_Q := \{x^*_i \mid x_i \in Q\}$, and for all $i \in [m]$, let $\sigma_{3,i} := (p'_i, p''_i, p'''_i)$ if both $h_{i,1}$ and $h_{i,2}$ have a parent in $\sigma_2 \cup X^*_Q$ and $\sigma_{3,i} = ()$, otherwise. Let $\sigma_{4,i} := (\overline{Y}_i, \overline{q}_i)$ for all $i \in [n]$, where $\overline{Y}_i = (s_{i,1}, \ldots, s_{i,|S^i|})$ and $\overline{q}_i = x_i$ whenever $x_i \notin Q$ otherwise $\overline{Y}_i = (t_{i,1}, \ldots, t_{i,|T^i|})$ and $\overline{q}_i = \overline{x}_i$. Let $\sigma_{5,i} := (p'_i, p''_i, p'''_i)$ for all $i \in [m]$ with $\sigma_{3,i} = ()$ and $\sigma_{5,i} := ()$, otherwise. Abbreviate $\sigma_j := \sigma_{j,1} \circ \sigma_{j,2} \circ \ldots$ for all $j \in \{3, 4, 5\}$. Finally, let $\sigma := \sigma_1 \circ \sigma_2 \circ (z_{n+1}, X'_Q, X^*_Q) \circ \sigma_3 \circ (P, C, d) \circ \sigma_4 \circ (X' \setminus X'_Q, X^* \setminus X^*_Q) \circ \sigma_5 \circ (\rho)$. It is tedious, but possible to verify that σ is an extension for G. In the following, we show $\operatorname{rw}^\sigma_v \leq k'$ for all vertices v of G.

Case $v \in \sigma_1$: Note that $|\sigma_1| = k' + 1$ and ψ is its last vertex. However, since A' avoids $\operatorname{RW}^\sigma_\psi$, we have $|\operatorname{RW}^\sigma_\psi| \leq k' + 1 - |A'| = k' - 2n - k$. Thus, $\operatorname{rw}^\sigma_v \leq k'$ for all $v \in \sigma_1$.

Case $v \in \sigma_2$: For each $i \in [n]$, the only vertices in $\sigma_{2,i}$ that have a parent after w_i in σ are z_i, q_i, and w_i. Then, by a simple induction, $\operatorname{RW}^\sigma_{w_i} \subseteq \operatorname{RW}^\sigma_\psi \cup \bigcup_{\ell \leq i} \{z_\ell, q_\ell, w_\ell\}$ for all $i \in [n]$, and, since $\operatorname{RW}^\sigma_{w_i}$ contains both parents of either $u_{1\ell}$ or $u_{2\ell}$ for all $\ell \leq i$, we conclude $\operatorname{rw}^\sigma_{w_i} \leq (k' - 2n - k) + 2i \leq k'$ for all $i \in [n] \cup \{0\}$. Aliasing $w_0 := \psi$, it is easy to see that $\operatorname{RW}^\sigma_{r_{i,j}} \subseteq \operatorname{RW}^\sigma_{w_{i-1}} \cup R'^i$ and, thus, $\operatorname{rw}^\sigma_{r_{i,j}} \leq (k' - 2n - k) + 2(i-1) + (2n - 2i + k + 2) = k'$, implying $\operatorname{rw}^\sigma_{z_i} \leq k'$ since $R'^i \subseteq \operatorname{RW}^\sigma_{r_{i,|R'^i|}} \setminus \operatorname{RW}^\sigma_{z_i}$ for all i. Similarly, for all $\gamma \in Y_i$, we have $\operatorname{RW}^\sigma_\gamma \subseteq \operatorname{RW}^\sigma_{w_{i-1}} \cup Y_i \cup \{z_i\}$, implying $\operatorname{rw}^\sigma_\gamma \leq (k' - 2n - k) + 2(i-1) + (2n - 2i + 1 + k) + 1 = k'$. Further, $\operatorname{RW}^\sigma_{q_i} = \operatorname{RW}^\sigma_{w_{i-1}} \cup \{z_i, q_i\}$, implying $\operatorname{rw}^\sigma_{q_i} \leq (k' - 2n - k) + 2(i-1) + 2 = k' - 2(n-i) - k \leq k'$. In summary, $\operatorname{rw}^\sigma_v \leq k'$ for all $v \in \sigma_{2,i}$ and all $i \in [n]$.

Case $v \in (z_{n+1}, X'_Q, X^*_Q)$: Note that $\operatorname{RW}^\sigma_{z_{n+1}} = \operatorname{RW}^\sigma_{w_n} \cup \{z_{n+1}\} \setminus \{\alpha\}$ and, thus, $\operatorname{rw}^\sigma_{z_{n+1}} = \operatorname{rw}^\sigma_{w_n} = (k' - 2n - k) + 2n = k' - k \leq k'$. Since $|X'_Q| \leq k$, we also have $\operatorname{rw}^\sigma_{x'_i} \leq \operatorname{rw}^\sigma_{z_{n+1}} + |X'_Q| \leq k'$ for all $x'_i \in X'_Q$. Then, $\operatorname{RW}^\sigma_{x^*_i} - x^*_i = \operatorname{RW}^\sigma_{x'_i} - x'_i$, implying $\operatorname{rw}^\sigma_{x^*_i} = \operatorname{rw}^\sigma_{x'_i} \leq k'$ for all $x^*_i \in X^*_Q$.

Case $v \in \sigma_3$: For each $i \in [m]$, let v'_{i-1} be the predecessor of p'_i in σ. Note that, for each $i \in [m]$ such that $\sigma_{3,i} \neq ()$, both $h_{i,1}$ and $h_{i,2}$ have one of their two parents in $\sigma_2 \cup X^*_Q$ (the other one being p'_i, resp. p''_i). This implies $\operatorname{RW}^\sigma_{p'_i} = \operatorname{RW}^\sigma_{v'_{i-1}} \cup \{p'_i\} \setminus \{h_{i,1}\}$, $\operatorname{RW}^\sigma_{p''_i} = \operatorname{RW}^\sigma_{p'_i} \cup \{p''_i\} \setminus \{h_{i,2}\}$, and $\operatorname{RW}^\sigma_{p'''_i} = \operatorname{RW}^\sigma_{p''_i} \cup \{p'''_i\} \setminus \{p''_i\}$. Thus, $\operatorname{rw}^\sigma_{p'''_i} = \operatorname{rw}^\sigma_{p''_i} = \operatorname{rw}^\sigma_{p'_i} = \operatorname{rw}^\sigma_{v'_{i-1}} \leq k'$ for all $i \in [n]$ with $\sigma_{3,i} \neq ()$. As we already showed $\operatorname{rw}^\sigma_{v'_0} \leq k'$, a simple induction shows $\operatorname{rw}^\sigma_v \leq \operatorname{rw}^\sigma_{v_0} \leq k'$ for all $v \in \sigma_3$.

Case $v \in (P, C)$: Let γ denote the last vertex of σ_3, let $i \in [m]$, and let $(x_j \vee x_\ell)$ be any clause of φ. Since Q covers it, we have $x_j \in Q$ (implying that the parent x^*_j of $f_{i,1}$ precedes p_i in σ) or $x_\ell \in Q$ (implying that the parent p'''_i of $f_{i,2}$ precedes p_i in σ). In either case, p_i is the last parent in σ of some leaf which, thus, is not in $\operatorname{RW}^\sigma_{p_i}$. By a simple induction, $\operatorname{rw}^\sigma_{p_i} \leq \operatorname{rw}^\sigma_\gamma$ for all $i \in [m]$. Further, as c_i is the only parent of p_i for all $i \in [m]$, we have $\operatorname{rw}^\sigma_{c_i} \leq \operatorname{rw}^\sigma_{p_i} \leq \operatorname{rw}^\sigma_\gamma$.

Case $v = d$: With γ denoting the predecessor of d in σ, we have $B' \uplus C \subseteq \operatorname{RW}^\sigma_\gamma \setminus \operatorname{RW}^\sigma_d$, so $\operatorname{rw}^\sigma_d \leq k' - |B'| - |C| + 1 = k' - (3n - m) - m + 1 = k' - 3n + 1$.

Case $v \in \sigma_4$: Let $\overline{q}_0 := d$. For each $i \in [n]$, we have $\operatorname{RW}^\sigma_{\overline{q}_i} \setminus \operatorname{RW}^\sigma_{\overline{q}_{i-1}} = \{\overline{q}_i\}$ as only vertices in $\overline{Y}_i$ are between $\overline{q}_{i-1}$ and $\overline{q}_i$ in σ and vertices in $\overline{Y}_i$ have $\overline{q}_i$ as unique parent. However, $\operatorname{RW}^\sigma_{\overline{q}_{i-1}} \setminus \operatorname{RW}^\sigma_{\overline{q}_i}$ contains either u_{i1} or u_{i2}, and, thus $\operatorname{rw}^\sigma_{\overline{q}_i} = \operatorname{rw}^\sigma_{\overline{q}_{i-1}} = \operatorname{rw}^\sigma_d \leq k' - 3n + 1 \leq k'$. Further, if $v \in \overline{Y_i}$, then $\operatorname{rw}^\sigma_v \leq \operatorname{rw}^\sigma_{\overline{q}_{i-1}} + |\overline{Y_i}| \leq k'$.

Case $v \in X' \setminus X'_Q, X^* \setminus X^*_Q$: As $|X' \setminus X'_Q| \leq n$, we have $\operatorname{rw}^\sigma_{x'_i} \leq \operatorname{rw}^\sigma_{\overline{q}_n} + n \leq k' - 3n + 1 + n \leq k'$ for each $x'_i \in X' \setminus X'_Q$. Further, $\operatorname{RW}^\sigma_{x^*_i} - x^*_i = \operatorname{RW}^\sigma_{x'_i} - x'_i$, implying $\operatorname{rw}^\sigma_{x^*_i} = \operatorname{rw}^\sigma_{x'_i} \leq k'$ for all $x^*_i \in X^*_Q$.

Case $v \in \sigma_5$: This case is analog to the case that $v \in \sigma_3$. For each $i \in [m]$, let v'_{i-1} be the predecessor of p'_i in σ. Since $X \cup X^* \cup P <_\sigma v$, p'_i is the last parent of $h_{i,1}$ in σ for all $i \in [m]$, implying $h_{i,1} \notin \operatorname{RW}^\sigma_{p'_i}$ and the same holds for p''_i (with respect to $h_{i,2}$) and p'''_i (with respect to $f_{i,2}$). Thus, $\operatorname{rw}^\sigma_{p'''_i} = \operatorname{rw}^\sigma_{p''_i} = \operatorname{rw}^\sigma_{p'_i} = \operatorname{rw}^\sigma_{v'_{i-1}} \leq k'$.

"⇐": Let σ be an extension for G with $\operatorname{rw}(\sigma) = k'$ and note that, by Observation 3 and Lemma 6, σ begins with A', B', F', U, H and α in any order, followed by ψ.

Claim 1. *Let $j \in [m]$ and let v be such that $|\sigma[1..v] \cap \{x_i, \overline{x}_i\}| \leq 1$ for all $i \in [n]$. Then, $\operatorname{RW}^\sigma_v$ contains at least four of $\{f_{j,1}, f_{j,2}, h_{j,1}, h_{j,2}, p'_j, p''_j, p'''_j, c_j\}$.*

Proof of Claim 1. If $p_j \not\leq_\sigma v$, then $\operatorname{RW}^\sigma_v$ contains $f_{j,1}$, $f_{j,2}$ and one in each of the paths $(\rho, p'_j, h_{j,1})$ and $(\rho, p'''_j, p''_j, h_{j,2})$ and, thus, the claim holds. Hence, suppose that $p_j \leq_\sigma v$. Then, $\operatorname{RW}^\sigma_v$ contains at least

one of $\{f_{j,1}, h_{j,1}\}$ (as at most one of their parents in $X \cup X^*$ precedes v by our choice of v) and at least one in the path (d, c_j, p_j). If $p_j''' \leq_\sigma v$, then RW_v^σ further contains p_j', p_j'''. Otherwise, RW_v^σ further contains $f_{j,2}$ and one in the path $(p_j''', p_j'', h_{j,2})$. ■

Claim 2. *Let $v \in \sigma[\psi..d]$. Then, $|\sigma[1..v] \cap \{x_i, \overline{x}_i\}| \leq 1$ for all $i \in [n]$.*

Proof of Claim 2. Note that, since x_i and $\overline{x}_i$ are children of the root ρ for each i, they are in RW_v^σ if and only if they are in $\sigma[1..v]$. We prove Claim 2 by induction on the index of v in σ. Clearly, the claim holds for $v = \psi$. Assume Claim 2 is false for v (that is, $x_i, \overline{x}_i \in \mathrm{RW}_v^\sigma$ for some $i \in [n]$) but holds for its predecessor w in σ. Then, $v = \overline{x}_i$ or $v = x_i$. Without loss of generality, let $v = \overline{x}_i$, implying $\mathrm{RW}_w^\sigma \supseteq T'^i \uplus \{x_i\}$. Further, $w <_\sigma d, \rho$, $z_i \leq_\sigma w$ and $w \not<_G z_i$ as σ extends G. Thus, RW_w^σ contains, B', $u_{j,3}$ for all $j \in [n]$, ψ, T'^i and x_i, along with

- $u_{i,2}$ and z_i (the other children of $\overline{x}_i$),
- one of the path $(\rho, w_i, u_{i,1})$
- one of the path (ρ, z_{n+1}, α),
- four of $\{f_{j,1}, f_{j,2}, h_{j,1}, h_{j,2}, p_j', p_j'', p_j''', p_j, c_j\}$ for all j (by Claim 1)
- z_ℓ and w_ℓ for all $\ell < i$ (since $z_\ell, w_\ell <_G w$ and both are children of ρ)
- one of each of the critical paths $(\rho, x_\ell, u_{\ell,1})$ and $(\rho, \overline{x_\ell}, u_{\ell,2})$

Thus,

$$\begin{aligned}
\mathrm{rw}_w^\sigma &\geq |B'| + n|\{\psi\}| + |T'^i| + |\{x_i\}| + |\{u_{i,2}, z_i\}| + 1/2|\{u_{i,1}, w_i\}| + 1/2|\{z_{n+1}, \alpha\}| \\
&\quad + {}^4\!/_9|\{f_{j,1}, f_{j,2}, h_{j,1}, h_{j,2}, p_j', p_j'', p_j''', p_j, c_j \mid j \in [m]\}| + |\{z_\ell, w_\ell \mid \ell < i\}| \\
&\quad + {}^2\!/_4|\{x_l, u_{l,1}, \overline{x}_l, u_{l,2} \mid l \in [n] - i\}| \qquad (1) \\
&= (3n - m) + n + 1 + (2n - 2i + k + 1) + 1 + 2 + 1 + 1 + 4m + 2(i-1) + 2(n-1) \\
&= 8n + 3m + k + 3 = k' + 1
\end{aligned}$$

contradicting $\mathrm{rw}(\sigma) = k'$. ■

In the following, consider the predecessor v of d in σ and call an index ℓ *dirty* if

$$|\,\mathrm{RW}_v^\sigma \cap \{f_{\ell,1}, f_{\ell,2}, p_\ell, c_\ell, h_{\ell,1}, h_{\ell,2}, p_\ell', p_\ell'', p_\ell'''\}| \geq 5.$$

Denote $X_Q := X \cap \sigma[1..v] = X \cap \mathrm{RW}_v^\sigma$ and let Q be the result of, for each dirty index j, adding an arbitrary variable of C_j to X_Q.

Claim 3. $|Q| \leq k$.

Proof of Claim 3. Let Y be the set of dirty indices. Then, $\mathrm{RW}_v^\sigma \setminus X^*$ contains

- B', $\{u_{j,3} \mid j \in [n]\}$, ψ, z_{n+1},
- z_i and w_i for all $i \in [n]$,
- at least four of $\{f_{j,1}, f_{j,2}, h_{j,1}, h_{j,2}, p_j', p_j'', p_j''', p_j, c_j\}$ for each $j \in [m] \setminus Y$ (by Claim 1),
- at least five of $\{f_{j,1,}, f_{j,2}, h_{j,1}, h_{j,2}, p_j', p_j'', p_j''', p_j, c_j\}$ for each $j \in Y$, and
- one of each of the critical paths $(\rho, x_i, u_{i,1})$ and $(\rho, \overline{x}_i, u_{i,2})$

Note that those are at least $8n + 3m + 2 + |Y| = k' - (k - |Y|)$ vertices. To obtain the full set RW_v^σ, it remains to count the vertices in $\mathrm{RW}_v^\sigma \cap X^* = X_Q$. Since $\mathrm{rw}_v^\sigma \leq \mathrm{rw}(\sigma) \leq k'$, we have $k' \geq |\,\mathrm{RW}_v^\sigma| \geq |X_Q| + (k' - (k - |Y|))$, implying $|X_Q| \leq k - |Y|$. As $|Q| = |X_Q| + |Y|$ we obtain $|Q| \leq k$. This also implies that at most $k - |Y|$ vertices of X^* precede z_{n+1} in σ (because of arcs in E_3', all vertices of X^* preceding $z_{n+1} \leq_\sigma d$ are still in RW_v^σ). ■

In order to show that Q covers all clauses of φ, assume that a clause $C_\ell = (x_i, x_j)$ is not covered by Q. Then, $x_i^*, x_j^* \notin \sigma[1..v]$ and, thus, $f_{\ell,1}, h_{\ell,2} \in \mathrm{RW}_v^\sigma$. Further, $c_\ell \in \mathrm{RW}_v^\sigma$ since c_ℓ is a child of d in G. Finally, $(\rho, p_\ell''', f_{\ell,2})$ and $(\rho, p_\ell', h_{\ell,1})$ are critical paths for v. Thus, ℓ is dirty and, by definition of Q, one of x_i and x_j is in Q, contradicting Q not covering C_ℓ. □

Note that networks G created by Construction 2 contain non-binary vertices, as well as many leaves. However, all non-binary vertices of G have nice properties that allow us to "binarize" them using reduction rules that we present in Section 3.2.

Observation 4. *Let (G', k') result from Construction 2 and let $u \in V(G')$.*
(a) If u is a non-leaf with $\deg^+(u) \geq 3$, then u has a child with in-degree 1.
(b) If u is a non-leaf with at least 3 parents, then the root ρ is a parent of u.
(c) If u is a leaf with at least 3 parents, then u has exactly 3 parents and ψ is one of them.

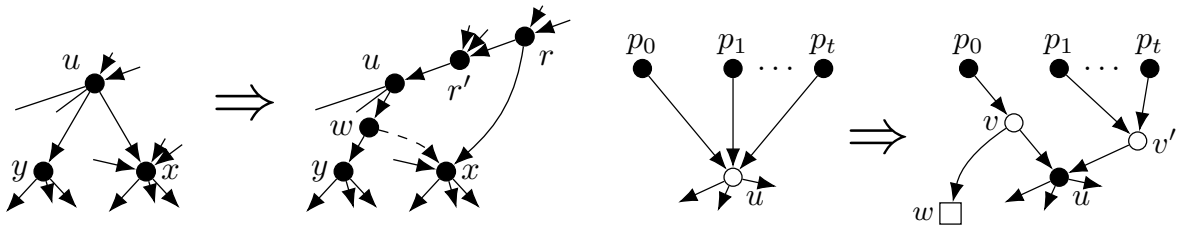
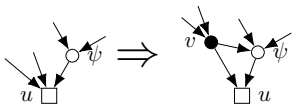


Figure 5: Illustration of Rule 3 (left) and Rule 4 (right). Triangles are leaves and children of the initial vertex ψ, white vertices are children of the root ρ. Note that, on the left, u has a tree-vertex child before and after the modification.

3.2 Reducing Nice Polytomies and Leaves

The following reduction rule is used to turn all leaves binary since many leaves constructed in Construction 2 have in-degree three.

Rule 1. *Let (G, k) have an initial vertex ψ and let u be a leaf in G with at least three parents, one of which is ψ. Then, add a new parent v to u, add the arc $v\psi$, and replace all xu by xv except ψu.*



The next rule splits vertices of in- and out-degree at least two into a reticulation and a tree-vertex.

Rule 2. *Let u be a vertex of G, let P and C be its parents and children, respectively, and let $|P| > 1$ and $|C| > 1$. Then, "split" u, i.e. add a new vertex v, add the arc uv and, for all $c \in C$, replace the arc uc by the arc vc.*

Rule 3 (See Figure 5(left))**.** *Let u be a vertex with at least three children, let x and y be children of u such that y is a tree-vertex and x is either a tree-vertex or x has a parent $q \neq u$ that is comparable to u in G. Then, "split" u into ru (that is, create a new parent r for u and make all parents of u parents of r instead), subdivide ru with a new vertex r', for all parents q of x with $q >_G u$ replace qx with qr', add the arc rx, subdivide uy with a new vertex w, remove the arc ux and, unless x has a parent $q <_G u$ in G, add the arc wx.*

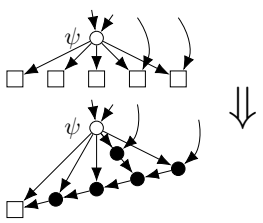
Correctness proofs of Rules 1–3 are deferred to the full version of this paper. Note that Rule 3 only increases $\deg^-_G(x)$ if x has no parents $q <_G u$ in G. But then, either x is a tree-vertex in G, in which case no new polytomies are created, or x has a parent $q >_G u$, in which case this parent becomes a parent of r' instead. Further, although Rule 3 may introduce degree-two vertices, all of them are parents of tree-vertices and can thus be removed using the following:

The following rule turns polytomous reticulations into binary ones. We make use of the fact that G has a root and an initial vertex (see Def. 2).

Rule 4 (see Figure 5, right)**.** *Let (G, k) have an initial vertex ψ, let ρ be the root of G, let u be a non-leaf with parents $p_0, p_1, \ldots, p_t, p_{t+1} = \rho$ $(t \geq 1)$. Then, add a new leaf w, increase k by one, subdivide p_0u with a vertex v, replace arc ρu by ρv, add a new parent v' of u, replace arc p_iu by p_iv' for all $i \in [t+1]$, and add the arcs $\rho v'$, vw, ψw.*

Note that Rule 4 effectively turns a vertex of in-degree $t+2$ (for $t \geq 1$) into a vertex of in-degree $t+1$. Further, note that the instance (G', k') constructed by Construction 2 has an initial vertex ψ.

Rule 5. *Let (G, k) have an initial vertex ψ, let X be the set of leaves of G and let $Y \subseteq X$ contain the leaves that have more than one incoming arc. Let $Y \neq \varnothing$ and let $x_1, x_2, \ldots, x_k$ be an arbitrary total order of X with $x_k \in Y$. Then, turn X into a path by adding the arc $x_{i+1}x_i$ for all i. Further, for all $y \in Y - x_k$, subdivide ψy with a new vertex z, and replace all arcs uy occurring in G by uz.*



Note that the graphs produced by Construction 2 satisfy $\varnothing \subsetneq Y \subsetneq X$.
Note that Rule 5 destroys the initial vertex property, preventing any further use of Rule 4 and Rule 5. However, Rule 5 does not create new polytomies and, for turning ψ binary by applying Rule 3, it is sufficient that ψ is "primal" (a weaker condition than being initial).

Lemma 7. *Let u be a vertex of G, let P and C be the set of parents and children, respectively, of u. Let G' be the result of removing all arcs in $\{u\} \times C$, adding a new vertex v as only child of u, and adding the arcs $\{v\} \times C$. Then,* $\mathrm{rw}(G') = \mathrm{rw}(G)$.

Proof. "$\leq$": Let σ be an extension for G and let σ' be the result of inserting v right before u in σ. By Lemma 10, σ' is an extension for G'. Towards a contradiction, assume that $\mathrm{rw}_x^{\sigma'} > \mathrm{rw}(\sigma)$ for some x. If $x <_{\sigma'} v$ or $x \geq_{\sigma'} u$, then $\mathrm{RW}_x^{\sigma'} = \mathrm{RW}_x^{\sigma}$. Thus, $x = v$ and, by Observation 2, $\mathrm{rw}_u^{\sigma'} \geq \mathrm{rw}_v^{\sigma'} - \deg_G^+(u) + 1 = \mathrm{rw}_v^{\sigma'}$, contradicting $\mathrm{rw}_u^{\sigma'} = \mathrm{rw}_u^{\sigma} \leq \mathrm{rw}(G)$.

"$\geq$": Let σ' be an extension for G', let $\sigma := \sigma'_{v \to \sqcup}$, and note that σ is an extension for G. Then, for all $x <_{\sigma'} v$ and all $x \geq_{\sigma'} u$, we have $\mathrm{RW}_x^{\sigma'} = \mathrm{RW}_x^{\sigma}$. Finally, RW_u^{σ} is the result of swapping v for u in $\mathrm{RW}_v^{\sigma'}$. Thus, $\mathrm{rw}_x^{\sigma} \leq \mathrm{rw}(\sigma')$ for all x. □

Lemma 8. *Let G' be the result of applying Rule 1 to a leaf u in G. Then,* $\mathrm{rw}(G') = \mathrm{rw}(G)$.

Proof of Lemma 8. Since ψ is initial for both G and G', we assume all extensions for G and G' correspond to Lemma 6.

"$\leq$": Let σ be an extension for G with $\mathrm{rw}(\sigma) = \mathrm{rw}(G)$ and let σ' result from σ by inserting v right after ψ. Clearly, σ' is an extension for G'. Now, for all $x <_{\sigma'} v$, we have $\mathrm{RW}_x^{\sigma'} = \mathrm{RW}_x^{\sigma}$. Further, $u \in \mathrm{RW}_{\psi}^{\sigma'} \setminus \mathrm{RW}_v^{\sigma'}$, implying $\mathrm{RW}_v^{\sigma'} - v \subseteq \mathrm{RW}_{\psi}^{\sigma'} - u$. Finally, for all $x >_{\sigma'} v$, we have $v \in \mathrm{RW}_x^{\sigma'}$ if and only if $u \in \mathrm{RW}_x^{\sigma}$ and, thus, $\mathrm{RW}_x^{\sigma'} - v = \mathrm{RW}_x^{\sigma} - u$. In all cases, $\mathrm{rw}_x^{\sigma'} \leq \mathrm{rw}(\sigma)$.

"$\geq$": Let σ' be an extension for G' with $\mathrm{rw}(\sigma') = \mathrm{rw}(G')$ and let $\sigma := \sigma'_{v \to \sqcup}$ (which is clearly an extension for G). Then, for all $x >_{\sigma'} v$, RW_x^{σ} results from $\mathrm{RW}_x^{\sigma'}$ by replacing v with u. Further, for all $x <_{\sigma'} v$, $\mathrm{RW}_x^{\sigma} = \mathrm{RW}_x^{\sigma'}$. In both cases, $\mathrm{rw}_x^{\sigma} \leq \mathrm{rw}_x^{\sigma'}$. □

In the following, we call a vertex u *primal* if there is an optimal extension σ for G such that, for all children v of u and all parents q of v, we have $u \leq_{\sigma} q$ ($u \geq_{\sigma} q$). While, in general, it may not be easy to decide if a vertex is primal or not, primality is implied by being initial.

Lemma 9. *Let G' be the result of applying Rule 3 to G. Then,* $\mathrm{rw}(G') = \mathrm{rw}(G)$. *Further, if u is primal in G, then u is primal in G'.*

To show correctness of Lemma 9, we use the following lemma.

Lemma 10. *Let G be a DAG containing an arc uv, but no other u-v-path, and let σ be a permutation of $V(G)$ extending (v, u). Let G' be the result of contracting v onto u and let $\sigma' := \sigma_{v \to \sqcup}$. Then, σ' extends G' if and only if σ extends G.*

Proof of Lemma 10. Note that, by construction, G' is a DAG. Further, for all $x, y \in V(G')$, we have $x <_{\sigma} y \iff x <_{\sigma'} y$.

"$\Rightarrow$": Let σ' extend G' and assume towards a contradiction that G contains an arc xy with $x <_{\sigma} y$. Then $xy \neq uv$ since σ extends (v, u). If $x = v$, then G' contains the arc uy and, thus, $y <_{\sigma'} u$, implying $y <_{\sigma} v$, contradicting $x <_{\sigma} y$. If $y = v$ and $x \neq u$, then G' contains xu and, thus, $u <_{\sigma'} x$, implying $v <_{\sigma} x$, contradicting $x <_{\sigma} y$. Finally, if $\{x, y\} \cap \{u, v\} = \varnothing$, then xy is also an arc of G' and, thus, $x <_{\sigma'} y$ implying $x <_{\sigma} y$ again.

"$\Leftarrow$": Let σ extend G and assume towards a contradiction that G' contains an arc xy with $x <_{\sigma'} y$. If xy is an arc of G, then $y <_{\sigma} x$, implying $y <_{\sigma'} x$. Thus, $x = u$ and G contains an arc vy and, thus, $y < \sigma v$, implying $y < \sigma u$ and $y <_{\sigma'} u$. □

Observation 5. *Let σ be an extension of G and let u and v be vertices such that v has in-degree one and $v <_G u$. Then, $v \notin \mathrm{RW}_u^{\sigma}$.*

Observation 6. *Let v be a tree vertex and let u be its parent. Then, there is an extension σ of G such that* $\mathrm{rw}(\sigma) = \mathrm{rw}(G)$ *and v is the predecessor of u in σ.*

Proof of Lemma 9. "$\leq$": Let σ be an extension of G with $\mathrm{rw}(\sigma) = \mathrm{rw}(G)$. Let σ' be the result of replacing u by (w, u, r', r) in σ. First, we show that σ' extends G'. To this end, consider the result G^* of making all parents q of x with $q >_G u$ parents of u in G. If σ extends G^* then, by Lemma 10, σ' extends G'. Clearly, any counter example must be an arc of G^* that is not in G, so let qu be such an arc (where q is a parent of x in G). If x is a tree-vertex, then qu does not exist. Otherwise, we know $u <_{\sigma} q$ by our choice of q. Thus, qu is not a counter example.

Second, we show that $\mathrm{rw}(\sigma') \leq \mathrm{rw}(\sigma)$. Assume towards a contradiction that there is some z with $\mathrm{rw}_z^{\sigma'} > \mathrm{rw}(\sigma)$ and let $\gamma \in \mathrm{RW}_z^{\sigma'} \setminus \mathrm{RW}_z^{\sigma}$. Observe that, if $z <_{\sigma'} w$ and γ has a parent in $\sigma'[w..]$ in G', then γ has a parent in $\sigma[u..]$ in G. Thus, $z \geq_{\sigma'} w$. Suppose that $z >_{\sigma'} r$. If $r' \in \mathrm{RW}_z^{\sigma'}$, then $x \in \mathrm{RW}_z^{\sigma} \setminus \mathrm{RW}_z^{\sigma'}$ and if $r \in \mathrm{RW}_z^{\sigma'}$, then $u \in \mathrm{RW}_z^{\sigma} \setminus \mathrm{RW}_z^{\sigma'}$. Thus, $\mathrm{rw}_z^{\sigma'} \leq \mathrm{rw}_z^{\sigma}$ in this case. Hence, $z \in \{w, u, r', r\}$. Let v denote the predecessor of u in σ (which is also the predecessor of w in σ') and note that RW_v^{σ} contains all children of u in G. Then, by Observation 5, $y \notin \mathrm{RW}_w^{\sigma'}$, $w \notin \mathrm{RW}_u^{\sigma'}$, and $u \notin \mathrm{RW}_{r'}^{\sigma'}$. Thus, $\mathrm{RW}_w^{\sigma'} - w \subseteq \mathrm{RW}_v^{\sigma} - y$, $\mathrm{RW}_u^{\sigma'} - u \subseteq \mathrm{RW}_w^{\sigma'} - w$, $\mathrm{RW}_{r'}^{\sigma'} - r' \subseteq \mathrm{RW}_u^{\sigma'} - u$, and $\mathrm{RW}_r^{\sigma'} - r \subseteq \mathrm{RW}_{r'}^{\sigma'} - x$. Together, we have $\mathrm{rw}_r^{\sigma'} \leq \mathrm{rw}_{r'}^{\sigma'} \leq \mathrm{rw}_u^{\sigma'} \leq \mathrm{rw}_w^{\sigma} \leq \mathrm{rw}_v^{\sigma} \leq \mathrm{rw}(\sigma)$, contradicting $z \in \{w, u, r', r\}$. Finally, since w and y are tree vertices in G', we conclude that, for all parents q of children of u in G, $q \leq_{\sigma} u$ implies $q \leq_{\sigma'} u$. Thus, if u is primal in G (witnessed by σ), then it is also primal in G' (witnessed by σ').

"$\geq$": Let σ' be an extension of G' such that $\mathrm{rw}(\sigma') = \mathrm{rw}(G')$ and w is the predecessor of u in σ' (σ' exists by Observation 6). Let $\sigma := \sigma'_{\{w,r',r\} \to \sqcup}$. First, σ is an extension for the result G^* of contracting $\{w, r', r\}$ onto u in G'. Since $x <_{G'} u$ (even if wx has been removed), we have $x <_{\sigma'} u$ and, thus, $x <_{\sigma} u$, σ is also an extension for any graph resulting from replacing any arcs qu with qx in G^*. Since G is such a graph, σ extends G. Second, we show that $\mathrm{rw}(\sigma) \leq \mathrm{rw}(\sigma')$. Assume towards a contradiction that there is some z with $\mathrm{rw}_z^{\sigma} > \mathrm{rw}(\sigma')$ and let $\gamma \in \mathrm{RW}_z^{\sigma} \setminus \mathrm{RW}_z^{\sigma'}$. Note that $\gamma \in \{x, u\}$ because otherwise $\gamma \in \mathrm{RW}_z^{\sigma'}$. Observe that, if $z <_{\sigma} u$ and γ has a parent in $\sigma[u..]$ in G, then γ has a parent in $\sigma'[w..]$ in G'. Thus, $z \geq_{\sigma} u$. Suppose that $z >_{\sigma} u$. Then, if $x \in \mathrm{RW}_z^{\sigma}$, then $r' \in \mathrm{RW}_z^{\sigma'} \setminus \mathrm{RW}_z^{\sigma}$ and if $u \in \mathrm{RW}_z^{\sigma}$, then $r \in \mathrm{RW}_z^{\sigma'} \setminus \mathrm{RW}_z^{\sigma}$. Thus, $\mathrm{rw}_z^{\sigma} \leq \mathrm{rw}_z^{\sigma'}$ in this case. Thus, $z = u$. But we have $\mathrm{RW}_u^{\sigma} - u \subseteq \mathrm{RW}_y^{\sigma} - y$ since $y \notin \mathrm{RW}_u^{\sigma}$ by Observation 5 and, thus, $\mathrm{rw}_u^{\sigma} \leq \mathrm{rw}_y^{\sigma}$. In all cases, $\mathrm{rw}_z^{\sigma} \leq \mathrm{rw}(\sigma')$. □

Lemma 11. *Let $(G', k+1)$ be the result of applying Rule 4 to a reticulation u in (G, k). Then, (G, k) is a yes-instance if and only if $(G', k+1)$ is. Further, ψ is an initial vertex of $(G', k+1)$.*

Proof. First, since u is not a leaf, the children of ψ in G' are exactly the children of ψ in G, plus w, which is a leaf. Thus, ψ is initial for $(G', k+1)$.

"$\Rightarrow$:" Let σ be an extension for G with $\mathrm{rw}(\sigma) \leq k$. We construct σ' by adding w as first element to σ and adding v and v' (in that order) after u. Clearly, σ' is an extension for G' and $\mathrm{rw}_x^{\sigma'} \leq \mathrm{rw}_x^{\sigma} + 1 \leq k+1$ for all $x \leq_{\sigma} u$ as $\mathrm{RW}_x^{\sigma} = \mathrm{RW}_x^{\sigma'} - w$. Further, $\mathrm{rw}_x^{\sigma'} \leq \mathrm{rw}_x^{\sigma} + 1$ for all $u <_{\sigma} x <_{\sigma} \rho$ as $\mathrm{RW}_x^{\sigma} - u = \mathrm{RW}_x^{\sigma'} \setminus \{v, v'\}$. Finally, $\mathrm{RW}_v^{\sigma'} = (\mathrm{RW}_u^{\sigma'} \cup \{v\}) - w$ and $\mathrm{RW}_{v'}^{\sigma'} = (\mathrm{RW}_v^{\sigma'} \cup \{v\}) - u$. Thus, $\mathrm{rw}_{v'}^{\sigma'} = \mathrm{rw}_v^{\sigma} = \mathrm{rw}_u^{\sigma'} \leq k+1$.

"$\Leftarrow$:" Let σ' be an extension for G' with $\mathrm{rw}(\sigma) \leq k+1$ and let $\sigma := \sigma'_{\{v,v',w\} \to \sqcup}$. Then, $\mathrm{rw}_x^{\sigma} \leq k$ for all $x <_{\sigma'} w$ since $|\sigma'[1..w] - w| \leq k$. Further, $\mathrm{rw}_x^{\sigma} \leq k$ for all $w <_{\sigma'} x \leq_{\sigma'} u$ by construction. Towards a contradiction, assume that there is some x with $\mathrm{rw}_x^{\sigma} > k$. Clearly, $u <_{\sigma'} x <_{\sigma'} \rho$. If $\{v, v'\} \leq_{\sigma'} x$, then $v, v' \in \mathrm{RW}_x^{\sigma'}$ and $\mathrm{RW}_x^{\sigma} \subseteq (\mathrm{RW}_x^{\sigma'} \cup \{u\}) \setminus \{v, v'\}$, implying $\mathrm{rw}_x^{\sigma} \leq \mathrm{rw}_x^{\sigma'} - 1 \leq k$. Otherwise, $u \in \mathrm{RW}_x^{\sigma'}$ and, since G' contains the path (ρ, v, w), we know that $\mathrm{RW}_x^{\sigma'}$ intersects $\{v, w\}$, implying $\mathrm{RW}_x^{\sigma} \subseteq \mathrm{RW}_x^{\sigma'} \setminus \{v, w\}$ and, thus, $\mathrm{rw}_x^{\sigma} \leq \mathrm{rw}_x^{\sigma'} - 1$. □

The following observation helps show correctness of the next reduction rule. Consider a fixed ordering σ of some G and note that the only effect that removing any arc uv from G has on $\mathrm{rw}(\sigma)$ is that rw_v^{σ} decreases.

Observation 7. *Let G' be a subgraph of G with $V(G) = V(G')$. Then, $\mathrm{rw}(G') \leq \mathrm{rw}(G)$.*

Lemma 12. *Let (G', k) be the result of applying Rule 5 to (G, k). Then, $\mathrm{rw}(G') = \mathrm{rw}(G)$. Furthermore, ψ is primal in G'.*

Proof. "$\geq$": Let G^* be the result of removing the arcs $x_{i+1}x_i$ for all i from G'. Then, G^* is isomorphic to the result of adding, to G, a new leaf child to all vertices of $Y - x_k$. By Lemma 7, $\mathrm{rw}(G^*) = \mathrm{rw}(G)$. Then, $\mathrm{rw}(G') \geq \mathrm{rw}(G^*)$ by Observation 7, as G^* is the result of deleting arcs from G'.

"$\leq$": Let σ be an extension for G with $\mathrm{rw}(\sigma) = \mathrm{rw}(G)$. Since ψ is initial for (G, k), Lemma 6 allows us to suppose without loss of generality that $\sigma[1..\psi] = (x_1, x_2, \ldots, x_k, \psi)$. Then, let σ' result from σ by, for all $y \in Y - x_k$, adding the parent of y in G' that is not in X right after y. We show that $\mathrm{rw}_{x_i}^{\sigma'} < k$ for all $i < k$. To this end, for all $i < k$, let $X_i := \{x_j \mid j \leq i\}$, let $Y_i := \{x_j \in Y \mid j < i\}$ and let Z_i denote the set of parents z of vertices $y \in Y_i$ with $z \notin X_i$ (note that $|Z_i| = |Y_i|$). Then, all $y \in Y_i$ have both their parents in $\sigma'[1..x_i]$. Thus, $\mathrm{RW}_{x_i}^{\sigma'} = (X_i \uplus Z_i) \setminus Y_i$, implying $\mathrm{rw}_{x_i}^{\sigma'} = i < k$. Further, $\mathrm{RW}_{x_k}^{\sigma'} = \mathrm{RW}_{x_{k-1}}^{\sigma'} \cup \{x_k\}$, implying $\mathrm{rw}_{x_k}^{\sigma'} \leq k$. Then, $\mathrm{rw}_{\psi}^{\sigma'} \leq k$ since $X \setminus Y$ contains some x_j, implying $x_j \notin \mathrm{RW}_{\psi}^{\sigma'}$. Also, for each parent $z \notin X$ of each $x_i \in Y$, we have $\mathrm{RW}_z^{\sigma'} - x_i = \mathrm{RW}_{x_{i+1}}^{\sigma'} - x_{i+1}$, implying $\mathrm{rw}_z^{\sigma'} \leq k$. Since, for all $q >_{\sigma} \psi$, $\mathrm{RW}_q^{\sigma'}$ results from RW_q^{σ} by replacing all $y \in Y$ with their parent outside

X in G', we conclude $\mathrm{rw}(\sigma') \leq \mathrm{rw}(\sigma)$. Finally, since σ' is optimal, we conclude that ψ is primal in G'. □

Theorem 4. Register Sufficiency *is* NP*-complete on rooted, single-leaf binary DAGs.*

Proof of Theorem 4. Given an instance (G, k) of Register Sufficiency produced by Construction 2, we employ the following algorithm:

1. use Rule 3 to binarize all tree vertices (except ψ and ρ)
2. use Rule 1 to turn all leaves binary (no non-binary leaf is created in the following steps)
3. use Rule 4 to reduce all in-degrees to at most two
4. use Rule 5 to establish a single leaf (none of the following steps creates leaves)
5. subdivide the two arcs incoming to the unique leaf with vertices χ and ϕ such that χ is a child of ψ and add the arc $\chi\phi$
6. use Rule 3 to turn ψ and ρ binary
7. use Rule 2 to ensure that each non-leaf vertex is either a reticulation or a tree vertex

We let G_i denote the resulting graph after Step i. By correctness of the rules, the resulting instance (G_7, k') is a yes-instance if and only if the input instance (G, k) is.

In the following, we verify that $G' := G_7$ is indeed binary and has a single leaf. To this end, notice that G contains the following tree polytomies with their children (ρ and ψ missing; tree-vertex children boxed; children of ψ or ρ underlined or underwaved).

$$
\begin{array}{lcl}
d & \to & \underline{B'}, \boxed{C} \\
z_{n+1} & \to & \boxed{\alpha}, \underline{w_n}, \underset{\sim}{z_n} \\
z_i & \to & \underset{\sim}{z_{i-1}}, \underline{w_{i-1}}, \boxed{r_{i,1}}, r_{i,2}, r_{i,3}, \dots \\
x_i & \to & \underline{u_{i,1}}, \underset{\sim}{z_i}, \boxed{s_{i,1}}, s_{i,2}, s_{i,3}, \dots \\
\overline{x}_i & \to & \underset{\sim}{z_i}, \underline{u_{i,2}}, \underline{\text{some } h_{j,1} \in H}, \boxed{t_{i,1}}, t_{i,2}, t_{i,3}, \dots \\
x_i^* & \to & \boxed{x_i'}, \underline{\text{some } f_{j,1} \in F'}, \underline{\text{some } h_{j,2} \in H}
\end{array}
$$

Notice that all these tree polytomies are ancestors of ψ and children of ρ. Further, all parents of all $r_{i,j}$ are descendants of z_i, all parents of all $s_{i,j}$ are descendants of x_i, and all parents of all $t_{i,j}$ are descendants of $\overline{x}_i$. Thus, in Step 1, Rule 3 turns all of them binary. Further, each of these tree polytomies has at most one non-binary non-leaf reticulation child (underwaved). In applying Rule 3, we can afford to never touch this child and just "merge" all other children into one. Since Rule 3 does not create vertices with higher in-degree than that of the child it is applied for, we can be sure that all non-binary reticulations in G_1 have already been non-binary reticulations in G. They are ψ, z_i, the leaves in U, F' and H. The leaves in U, F', and H have in-degree at most three in G (in fact, all leaves do) and applying Rule 3 to some of their parents does not increase their in-degree. Thus, in Step 2, Rule 1 turns each of them into a binary leaf and a vertex of in- and out-degree two. Further, Rule 1 keeps ψ initial for (G_2, k). In Step 3, we use Rule 4 to reduce the in-degree of ψ and all z_i. Note that this increases k (and we let k' denote this new number) and Again, no non-binary vertices are created in this process and ψ is initial for (G_3, k'). Thus, in Step 4, we can use Rule 5 to establish a unique leaf in the graph without creating non-binary vertices. Note, however, that (G_4, k') no longer has the initial-vertex property. Further, ψ does not have a tree child in G_4, which is required to turn ψ binary using Rule 3. To remedy this, we install a new tree vertex χ below ψ in Step 5. It can be readily seen that this does not affect the register width, that is, $\mathrm{rw}(G_5) = \mathrm{rw}(G_4)$. Now, in Step 6, we apply Rule 3 to ψ, using χ as tree vertex. This requires all other children of ψ (except one) to have a parent that is comparable to ψ in G_5. In G_4, many of these children are no longer leaves. However, for each child v of ψ in G_4, there is a leaf in G_3 whose parents are exactly the parents of v in G_4. Thus, it suffices to show that all leaves in G_3 have a parent in G_3 that is an ancestor of ψ in G_3 (note that all ancestors of ψ in G_3 are also ancestors of ψ in G_4).

- All leaves in $A' \cup \{\alpha\}$ have, in G_4, a parent that is a leaf in G_3 and, thus, comparable to ψ.
- All leaves with in-degree three in G (that is, U, F', H) have received a parent that is also a parent of ψ in Step 2. This vertex is still an ancestor of ψ in G_3 and, thus, is G_4.
- All leaves in B' have d as their other parent in G_3, which is an ancestor of ψ in G_3 and, thus, in G_4.
- All leaves that were added in Step 3 have a parent that is an ancestor of either ψ or z_i in G_3 (as they were added when applying Rule 4 to ψ or z_i, respectively). As z_i is an ancestor of ψ in G_3 and, thus, in G_3, all these leaves have an ancestor of ψ as parent in G_3.

As this covers all leaves of G_3, we conclude that Step 6 succeeds at turning ψ binary. Finally, Rule 3 can be used to turn ρ binary since it has a leaf child (for example, p_1''') and all vertices are descendants of

ρ (which does not change during the repeated application of Rule 3 to it). However, this turns all tree vertices that are children of ρ into vertices with both in- and out-degree two, so we need to remember to apply Rule 2 in the end (which also splits the nodes created when binarizing U, F', and H in Step 2 and the nodes in R', S' and T', as they all have both in- and out-degree two in G_6). □

Finally, with Proposition 1, Theorem 4 implies NP-hardness of deciding the scanwidth for binary networks with a single leaf. Further, since a vertex-ordering is a polynomial-size certificate and its scanwidth can be checked in linear time, we conclude that the problem is complete for NP.

Acknowledgments. We thank Fabio Pardi to have brought the problem to our attention and the Genome Harvest project, ref. ID 1504-006 ("Investissements d'avenir", ANR-10-LABX-0001-01).

References

[1] J. Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, Jun 2006. ISSN 1435-5914.

[2] M. Bordewich and C. Semple. Computing the hybridization number of two phylogenetic trees is fixed-parameter tractable. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(3):458–466, 2007.

[3] M. Bordewich, C. Scornavacca, N. Tokac, and M. Weller. On the fixed parameter tractability of agreement-based phylogenetic distances. *Journal of Mathematical Biology*, 74(1):239–257, 2017. ISSN 1432-1416.

[4] D. Bryant and J. Lagergren. Compatibility of unrooted phylogenetic trees is fpt. *Theoretical Computer Science*, 351 (3):296 – 302, 2006. ISSN 0304-3975.

[5] D. Bryant, R. Bouckaert, J. Felsenstein, N. A. Rosenberg, and A. RoyChoudhury. Inferring species trees directly from biallelic genetic markers: bypassing gene trees in a full coalescent analysis. *Mol. Biol. Evol.*, 29(8):1917–1932, Aug 2012.

[6] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, Sept. 2002. ISSN 0360-0300.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co Ltd, 1979.

[8] A. Grigoriev, S. Kelk, and N. Lekić. On low treewidth graphs and supertrees. In *Proc. 1st AlCoB*, pages 71–82. Springer, 2014. ISBN 978-3-319-07953-0.

[9] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press, 2010.

[10] S. Kelk and C. Scornavacca. Constructing minimal phylogenetic networks from softwired clusters is fixed parameter tractable. *Algorithmica*, 68(4):886–915, 2014.

[11] S. Kelk, G. Stamoulis, and T. Wu. Treewidth distance on phylogenetic trees. *Theor. Comput. Sci.*, 731:99–117, 2018.

[12] C.-E. Rabier, V. Berry, F. Pardi, and C. Scornavacca. On the inference of complicated phylogenetic networks by markov chain monte-carlo. submitted.

[13] R. Sethi. Complete register allocation problems. *SIAM Journal on Computing*, 4(3):226–248, 1975.

[14] C. Whidden, R. G. Beiko, and N. Zeh. Fixed-parameter algorithms for maximum agreement forests. *SIAM Journal on Computing*, 42(4):1431–1466, 2013.

[15] C. Zhang, H. A. Ogilvie, A. J. Drummond, and T. Stadler. Bayesian Inference of Species Networks from Multilocus Sequence Data. *Mol. Biol. Evol.*, 35(2):504–517, 02 2018.