Reconstructing semi-directed level-1 networks using few quarnets

Frohn, Martin; Holtgrefe, Niels; van Iersel, Leo; Jones, Mark; Kelk, Steven

DOI
10.1016/j.jcss.2025.103655

Publication date
2025

Document Version
Final published version

Published in
Journal of Computer and System Sciences

Journal of Computer and System Sciences 152 (2025) 103655

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

Reconstructing semi-directed level-1 networks using few quarnets ☆

Martin Frohn [a], Niels Holtgrefe [b,*], Leo van Iersel [b], Mark Jones [b], Steven Kelk [a]

[a] Department of Advanced Computing Sciences, Maastricht University, Paul-Henri Spaaklaan 1, Maastricht, 6229 EN, the Netherlands
[b] Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, Delft, 2628 CD, the Netherlands

ARTICLE INFO

Article history:
Received 9 September 2024
Received in revised form 12 February 2025
Accepted 5 March 2025
Available online 19 March 2025

Keywords:
Phylogenetics
Semi-directed network
Quarnets
Quartets
Tree-of-blobs
Splits

ABSTRACT

Semi-directed networks are partially directed graphs that model evolution where the directed edges represent reticulate evolutionary events. We present an algorithm that reconstructs binary n-leaf semi-directed level-1 networks in $\mathcal{O}(n^2)$ time from its quarnets (4-leaf subnetworks). Our method assumes we have direct access to all quarnets, yet uses only an asymptotically optimal number of $\mathcal{O}(n \log n)$ quarnets. When the network is assumed to contain no triangles, our method instead relies only on four-cycle quarnets and the splits of the other quarnets. A variant of our algorithm works with quartets rather than quarnets and we show that it reconstructs most of a semi-directed level-1 network from an asymptotically optimal $\mathcal{O}(n \log n)$ of the quartets it displays. Additionally, we provide an $\mathcal{O}(n^3)$ time algorithm that reconstructs the tree-of-blobs of any binary n-leaf semi-directed network with unbounded level from $\mathcal{O}(n^3)$ splits of its quarnets.

1. Introduction

Phylogenetic networks are directed acyclic graphs that model the evolutionary history of taxa, represented by the leaves of the graph. Unlike phylogenetic trees, these networks can account for reticulate events such as hybridization and horizontal gene transfer, making them essential tools in evolutionary studies [1]. Among these, phylogenetic *level-1* networks (originally called 'galled trees' [2,3]) form a fundamental class [4], characterized by tree-like components and isolated cycles with a single reticulation. Solís-Lemus and Ané [5] introduced the semi-directed topology of a phylogenetic network (see Fig. 1): partially directed graphs that only model the direction of evolutionary change for reticulate events while the location of the root of the network remains unknown. If an outgroup (i.e. a more distantly related taxon) is included in the analysis, the directed topology can be recovered from the semi-directed network [5]. At the expense of containing less evolutionary information than their directed counterparts, semi-directed level-1 networks show favourable identifiability results under many evolutionary models [6–12]. That is, under these models, it is theoretically possible to infer (most of) the semi-directed level-1 network from different types of biological data (such as aligned nucleotide sequences or gene trees).

A major challenge in phylogenetics is the reliable construction of networks that best fit the finite biological data that is available. Various existing practical software tools can be used to infer phylogenetic level-1 networks (both directed and semi-directed) from biological data under different model assumptions and using different techniques. For example,

☆ This work was supported by grants OCENW.M.21.306 and OCENW.KLEIN.125 of the Dutch Research Council (NWO).
* Corresponding author.
E-mail addresses: martin.frohn@maastrichtuniversity.nl (M. Frohn), n.a.l.holtgrefe@tudelft.nl (N. Holtgrefe), l.j.j.vaniersel@tudelft.nl (L. van Iersel), m.e.l.jones@tudelft.nl (M. Jones), steven.kelk@maastrichtuniversity.nl (S. Kelk).

https://doi.org/10.1016/j.jcss.2025.103655

PhyloNet [13,14], SNaQ [5,15] and PhyNEST [16] first build a starting network and then explore the space of networks to choose a best network based on a likelihood criterion. On the other hand, Leviathan [17], TriLoNet [18], and NANUQ [19] utilize information on substructures with few leaves to puzzle together the structure of the full network. Recently, two new programs that combine 4-leaf substructures to construct a semi-directed level-1 network were introduced: NANUQ$^+$ [20] and Squirrel [21]. Several theoretical contributions focused on building phylogenetic level-1 networks from substructures have also appeared, see e.g. [22–26] and references therein.

A common starting point in many of these methods is to first infer a set of unrooted displayed *quartets* (4-leaf trees) for every subset of four leaves [5,19,20,22,24,25]. Note that the displayed quartets of a semi-directed network are the 4-leaf trees displayed in the 4-leaf subnetworks of the network. Baños [6] showed that the displayed quartets of a semi-directed level-1 network do not provide enough information to reconstruct every possible semi-directed level-1 network. On the positive side, Huber et al. [27] recently showed that all semi-directed level-1 (and even level-2) networks can be distinguished by the semi-directed *quarnets* (4-leaf subnetworks) they induce. Motivated by positive identifiability results for inferring quarnets under certain group-based models of evolution (Jukes-Cantor, Kimura-2 and Kimura-3) [10–12], some practical programs have already arisen that infer such quarnets from practical sequence data [21,28,29]. These programs have inspired the software tool Squirrel [21], which employs quarnets without triangles (3-cycles) and then builds a semi-directed triangle-free level-1 network. Additionally, Huebler et al. [23] described two theoretical algorithms that reconstruct an n-leaf semi-directed level-1 network using all of its $\Theta(n^4)$ induced quarnets, although full correctness was not formally proved. This is in stark contrast with the best algorithms that reconstruct binary phylogenetic trees from quartets. Assuming direct access to all quartets is available, this can be done in $\mathcal{O}(n \log n)$ time, while using only $\mathcal{O}(n \log n)$ quartets [30,31]. Here, assuming direct access means that an 'oracle' is available from which the algorithm can query a quartet on any set of leaves.

Reducing the computational overhead by utilizing only the essential quarnets could significantly speed up practical network construction algorithms, enabling more efficient and scalable approaches to reconstructing semi-directed level-1 networks. We present an algorithm that constructs a semi-directed level-1 network from its induced quarnets with bounds similar to those for tree reconstruction, thus taking a first step towards network construction methods that rely only on a small subset of quarnets. Specifically, our algorithm reconstructs an n-leaf semi-directed level-1 network using $\mathcal{O}(n \log n)$ quarnets (again assuming direct access to all quarnets), and running in $\mathcal{O}(n^2)$ time. Similar to trees, we show that one needs $\Omega(n \log n)$ quarnets, thus proving optimality of the number of quarnets that are used.

Our algorithm works by repeatedly attaching leaves to construct a canonical form of a network while using only the splits of the quarnets. Subsequently, the four-cycle quarnets are used to determine the remaining parts of the larger cycles, while the triangles in this canonical form can be inferred from the triangles in the quarnets. The structure of our algorithm aligns well with the earlier mentioned existing quarnet inference methods which provide evidence that inferring the splits of quarnets from data is easier than inferring four-cycle quarnets, while identifying triangles in quarnets is the most challenging [21,28,29]. Having access to the displayed quartets of a network allows one to deduce most of the topology of the quarnets of the network [6]. Consequently, our approach also provides similar complexity improvements for the related problem of constructing (most of) a semi-directed level-1 network from its displayed quartets.

Additionally, this paper covers the *tree-of-blobs* of a semi-directed network, originally introduced by Gusfield et al. [32] for directed phylogenetic networks. Such a tree shows only the tree-like aspects of a network and was shown to be identifiable under several models [8,33,34]. The tree-of-blobs is for example useful in the context of [34]. There it was shown that under several evolutionary models, once the tree-of-blobs is known, displayed quartets can be used to infer the circular order in which subnetworks attach to an outer-labeled planar blob. On the algorithmic side, Allman et al. [35] recently introduced the practical quartet-based program TINNIK that constructs a tree-of-blobs from gene trees under the network multispecies coalescent model. Gambette, Berry, and Paul [24] presented an algorithm to construct the 'SN-tree' from a set of quartets, which can also be used to construct the tree-of-blobs of an n-leaf semi-directed network from all $\Theta(n^4)$ quartets displayed by the network. Furthermore, the IQ* algorithm [36] can be applied to construct the tree-of-blobs using all $\Theta(n^4)$ splits of the quarnets induced by a semi-directed network (see the proof in the supplementary material of [21]). In our paper, we improve upon these aforementioned algorithms by reconstructing the tree-of-blobs of any n-leaf semi-directed network in $\mathcal{O}(n^3)$ time using $\mathcal{O}(n^3)$ splits of its quarnets, which can also be deduced from the displayed quartets of the network.

The remainder of the paper is structured as follows. Section 2 contains definitions and notation, while Section 3 introduces some crucial results on splits in semi-directed networks. We continue with the tree-of-blobs reconstruction algorithm in Section 4 and refine our approach in Section 5 to reconstruct level-1 networks. We end with a discussion in Section 6.

2. Preliminaries

Phylogenetic trees and networks A *blob* of a directed graph is a maximal subgraph without any cut-edges and it is an *m-blob* if it has m edges incident to it. A *(binary) directed phylogenetic network* on a set of at least two leaves $\mathcal{X}$ is a rooted directed acyclic graph with no edges in parallel such that (i): it has no 1-blobs or 2-blobs, other than possibly a blob with no incoming and two outgoing edges; (ii): the root has out-degree two; (iii): each vertex with out-degree zero has in-degree one and the set of such vertices is $\mathcal{X}$; (iv): all other vertices either have in-degree one and out-degree two, or in-degree two and out-degree one. A vertex of the last type is a *reticulation vertex*, and the two edges directed towards it are called

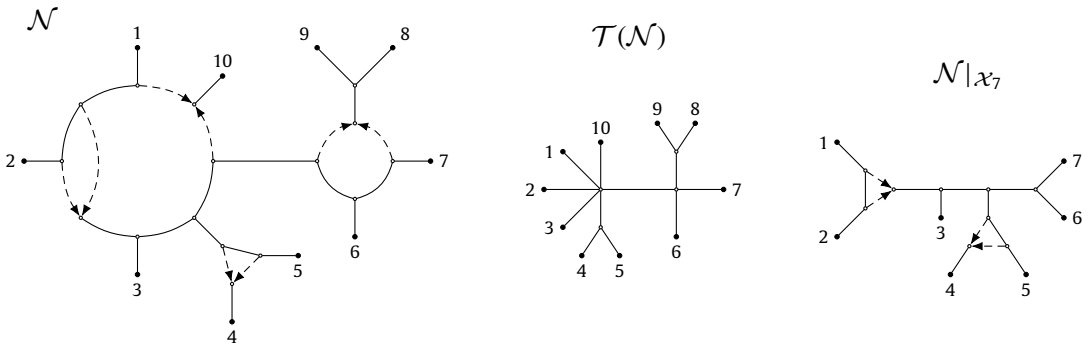

Fig. 1. A semi-directed level-2 network $\mathcal{N}$ on $\mathcal{X} = \{1, \ldots, 10\}$, its tree-of-blobs $\mathcal{T}(\mathcal{N})$, and its subnetwork $\mathcal{N}|_{\mathcal{X}_7}$ induced by the first seven leaves $\mathcal{X}_7 = \{1, \ldots, 7\}$.

reticulation edges. A directed phylogenetic network without reticulation vertices is a *(binary) directed phylogenetic tree*. The type of network this paper revolves around can be obtained from a directed phylogenetic network as follows.

Definition 1 *(Semi-directed network).* A *(binary) semi-directed phylogenetic network* $\mathcal{N}$ on $\mathcal{X}$ is a partially directed graph that (i): can be obtained from a directed phylogenetic network by undirecting all non-reticulation edges and suppressing the former root; (ii): does not have any parallel edges, degree-2 vertices, 1-blobs or 2-blobs.

For the sake of brevity, we refer to the above networks simply as *semi-directed networks*. Since we do not undirect reticulation edges, we can still refer to the reticulation vertices and edges of a semi-directed network. The *skeleton* of a semi-directed network is the undirected graph that is obtained from the network by removing all edge directions. A *(binary undirected) phylogenetic tree* on at least two leaves $\mathcal{X}$ is an undirected binary tree with leaf set $\mathcal{X}$ and no degree-2 vertices. Clearly, such a tree is a specific type of semi-directed network. The non-leaf vertices of such a tree are *internal vertices*. We say that a phylogenetic tree is a *displayed tree* of a semi-directed network on $\mathcal{X}$ if the tree can be obtained from the network by deleting one reticulation edge per reticulation vertex, undirecting the remaining reticulation edges, removing all vertices not on any path between a pair of leaves in $\mathcal{X}$, and suppressing degree-2 vertices.

Analogous to a directed phylogenetic network, a *blob* of a semi-directed network is a maximal subgraph without any cut-edges and it is an *m-blob* or has *degree-m* if it has m edges incident to it. The blob is *trivial* if it consists of a single vertex and it is *internal* if it is not a leaf. A semi-directed network is *simple* if it has at most one internal blob, it is *level-ℓ* if every blob contains at most ℓ reticulation vertices, and it is *strict level-ℓ* if it is not level-$(\ell - 1)$. Consequently, a semi-directed network is level-1 if every blob is either a single vertex or a *k-cycle*: a cycle of $k \geq 3$ vertices where directions are disregarded. We refer to 3-cycles as *triangles* and call a semi-directed network *triangle-free* if none of its blobs are triangles.

The *tree-of-blobs* $\mathcal{T}(\mathcal{N})$ of a semi-directed network $\mathcal{N}$ is obtained by contracting every blob $\mathcal{B}$ to a single vertex v, in which case we say that *v represents $\mathcal{B}$*. By definition, the tree-of-blobs $\mathcal{T}(\mathcal{N})$ must be an undirected phylogenetic tree on $\mathcal{X}$ since a semi-directed network contains no 2-blobs. It is well-known that constructing the tree-of-blobs of a given network takes linear time [37]. To illustrate some of the previous definitions, we refer to Fig. 1, which shows a semi-directed level-2 network and its tree-of-blobs.

An *up-down path* between two leaves x_1 and x_2 of a semi-directed network is a path of k edges where the first ℓ edges are directed towards x_1 and the last $k - \ell$ edges are directed towards x_2. Here, we consider undirected edges to be bidirected. With this notion, we can define the (induced) subnetwork of a semi-directed network (illustrated by Fig. 1). This was shown to be well-defined in [27], while Baños [6] showed the useful fact that first taking the semi-directed network of a directed network and then inducing a subnetwork is equivalent to first inducing a directed subnetwork and then making it a semi-directed network. In case the original network is a phylogenetic tree, we often call the subnetwork a *subtree* instead.

Definition 2 *(Subnetwork).* Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and some $\mathcal{Y} \subseteq \mathcal{X}$ with $|\mathcal{Y}| \geq 2$, the *subnetwork* of $\mathcal{N}$ induced by $\mathcal{Y}$ is the semi-directed network $\mathcal{N}|_{\mathcal{Y}}$ obtained from $\mathcal{N}$ by taking the union of all up-down paths between leaves in $\mathcal{Y}$, followed by exhaustively suppressing all 2-blobs and degree-2 vertices, and identifying parallel edges.

An interesting fact is that, in general, $\mathcal{T}(\mathcal{N})|_{\mathcal{Y}} \neq \mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$. In other words, the subtree of a tree-of-blobs need not be equal to the tree-of-blobs of a subnetwork. In particular, the latter can be more refined. As an example, consider Fig. 1, where we have that $\mathcal{T}(\mathcal{N})|_{\mathcal{X}_7} \neq \mathcal{T}(\mathcal{N}|_{\mathcal{X}_7})$. This property shows why reconstructing the tree-of-blobs of a semi-directed network is inherently different than doing so for undirected networks. The difficulty for semi-directed networks is that a cut-edge separating two sets of leaves in a subnetwork might not exist in the full network, which would be the case for undirected networks.

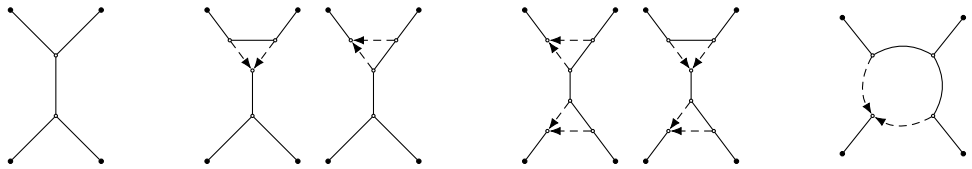

Fig. 2. The six possible quarnets of a semi-directed level-1 network, up to labeling the leaves (indicated by the filled black vertices).

As a technical convention, whenever we consider a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and some $\mathcal{Y} \subseteq \mathcal{X}$, we implicitly assume that $|\mathcal{Y}| \geq 2$ such that $\mathcal{N}|_{\mathcal{Y}}$ is a valid semi-directed network. Similarly, whenever we consider some $\mathcal{Y} \subset \mathcal{X}$ with $x \in \mathcal{X} \setminus \mathcal{Y}$, we assume that $|\mathcal{Y}| \geq 2$ such that $|\mathcal{X}| \geq 3$.

Splits, quarnets, quarnet-splits and displayed quartets Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and a partition $A|B$ of $\mathcal{X}$ (with A and B both non-empty), we say that $A|B$ is a *split* in $\mathcal{N}$ if there exists a cut-edge of $\mathcal{N}$ whose removal disconnects the leaves in A from those in B. The split and the cut-edge are *non-trivial* if the corresponding partition is non-trivial, that is, if $|A|, |B| \geq 2$. For example, $\{1, 2, 3, 4, 5, 10\}|\{6, 7, 8, 9\}$ is a non-trivial split in the semi-directed network $\mathcal{N}$ from Fig. 1. We may sometimes omit the word 'non-trivial' if it is clear from the context. A split $A|B$ is *induced by the cut-edge* uv if u (resp. v) is *on the side of* A (resp. B), meaning that u (resp. v) is in the component of $\mathcal{N}$ containing A (resp. B) after removing uv. For splits with few leaves, we sometimes omit the set notation, e.g. $ab|cd$ instead of $\{a, b\}|\{c, d\}$. A very well-known result is that an undirected (possibly non-binary) phylogenetic tree is uniquely determined by its (non-trivial) splits [38]. Consequently, the tree-of-blobs of a semi-directed network is the unique undirected phylogenetic tree with the same set of splits as the network itself.

Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and four different leaves $a, b, c, d \in \mathcal{X}$, the *(semi-directed) quarnet* of $\mathcal{N}$ on $\{a, b, c, d\}$ is the subnetwork of $\mathcal{N}$ induced by $\{a, b, c, d\}$. A semi-directed level-1 network can have six different types of quarnets, up to labeling the leaves: the *quartet tree*, two *single triangles* that share the same skeleton, two *double triangles* that share the same skeleton, and the *four-cycle* (which we distinguish from a 4-cycle in a network by writing out the number 4). Fig. 2 shows these six quarnet types. Any quarnet $\mathcal{N}|_{\{a,b,c,d\}}$ has four trivial splits: one cutting off each leaf. Next to that, the quarnet either has no non-trivial split at all, or it has exactly one non-trivial split ($ab|cd$, $ac|bd$, or $ad|bc$). We say that $ab|cd$ is a *quarnet-split* of $\mathcal{N}$ if $ab|cd$ is a split of the quarnet $\mathcal{N}|_{\{a,b,c,d\}}$. We call the set of quartet trees that are displayed by the quarnets of a semi-directed network the *(displayed) quartets* of the semi-directed network. Note that there are $\Theta(n^4)$ quarnets and $\Theta(n^4)$ displayed quartets of a semi-directed network on n leaves. An important observation we will often implicitly use is the following.

Observation 1. Let $\mathcal{N}$ be a semi-directed network on $\mathcal{X}$ with $\{a, b, c, d\} \subseteq \mathcal{Y} \subseteq \mathcal{X}$. Then, the quarnet of $\mathcal{N}$ on $\{a, b, c, d\}$ is equal to the quarnet of $\mathcal{N}|_{\mathcal{Y}}$ on $\{a, b, c, d\}$. That is, $\mathcal{N}|_{\{a,b,c,d\}} = (\mathcal{N}|_{\mathcal{Y}})|_{\{a,b,c,d\}}$.

The algorithms in this paper revolve around quarnet-splits and quarnets of semi-directed networks. We emphasize that from an informational perspective, having access to the displayed quartets of a semi-directed network lies somewhere in between these two. In particular, $ab|cd$ is a quarnet-split of a semi-directed network $\mathcal{N}$ if and only if $\mathcal{N}$ has exactly one displayed quartet on leaf set $\{a, b, c, d\}$ and this quartet has the split $ab|cd$ [34, Lem. 5.1]. Therefore, the quarnet-splits of a semi-directed network can be deduced from its displayed quartets. Although the quarnet-splits might contain less information, they are possibly easier to infer from data [29]. Since there are at most three times as many displayed quartets compared to quarnet-splits, our asymptotic complexity results of the two algorithms that rely on quarnet-splits (Algorithms 1 and 2) also hold when using displayed quartets instead. Lastly, whenever we say that an algorithm 'uses' m quarnet-splits of $\mathcal{N}$, this means the algorithm examines m quarnets to determine if they have a non-trivial split, and, if so, identifies the specific split. Thus, the count m accounts for all quarnets analyzed for the presence of a split, even those turning out not to have a non-trivial split. We use this terminology to emphasize that only the (possible absence of a) non-trivial split of a quarnet is used, and not the complete topology of the quarnet.

3. Determining splits of semi-directed networks with quarnets

This section is focused on a result that connects the splits in a semi-directed network to the quarnet-splits of that network, which will be useful in the algorithms in later sections. Recall that by Observation 1 the quarnet(-split) of a subnetwork is equal to the quarnet(-split) of the whole network. Before proving the main result, we will first present Lemma 1, which is a slightly stronger version of a result by Huber et al. [27]. They show that $A|B$ is a split of a semi-directed phylogenetic network if and only if $a_1a_2|b_1b_2$ is a quarnet-split for all $a_1, a_2 \in A$ and $b_1, b_2 \in B$, whereas we allow the leaves a_1 and b_1 to be fixed. The proof is a mere adjusted version of the proof by Huber et al. [27] and is thus deferred to the appendix.

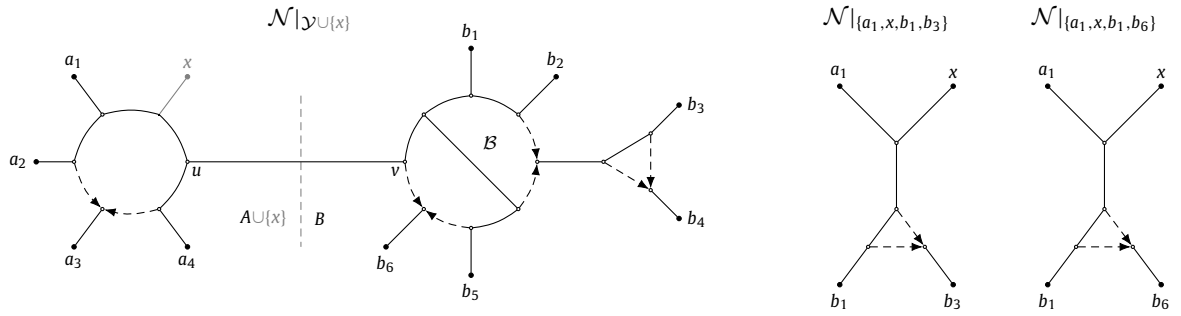

Fig. 3. *Left:* The subnetwork of a semi-directed level-2 network $\mathcal{N}$ induced by the leaves $\mathcal{Y} = \{a_1, \ldots, a_4, b_1, \ldots, b_6\}$ (excluding the grey edge and the grey leaf x) and induced by the leaves $\mathcal{Y} \cup \{x\}$ (including the grey edge and the grey leaf x). The cut-edge uv of $\mathcal{N}|_{\mathcal{Y}}$ induces the split $A|B$ (with $A = \{a_1, \ldots, a_4\}$ and $B = \{b_1, \ldots, b_6\}$) and v is part of the blob $\mathcal{B}$. *Right:* Two quarnets of $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ with splits $a_1x|b_1b_3$ and $a_1x|b_1b_6$, respectively. By Theorem 2 and since $\{a_1, b_1\} \cup B'$ with $B' = \{b_3, b_6\}$ is a distinguishing set of $\mathcal{B}$ in $\mathcal{N}|_{\mathcal{Y}}$, the splits of these two quarnets are enough to determine that $A \cup \{x\}|B$ is a split of $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$, without knowing anything else about x.

Lemma 1. *Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$, a non-trivial partition $A|B$ of $\mathcal{X}$ and any $a_1 \in A$, $b_1 \in B$, the following are equivalent:*

(i) $A|B$ is a split in $\mathcal{N}$;

(ii) $a_1a_2|b_1b_2$ is a quarnet-split of $\mathcal{N}$ for all $a_2 \in A \setminus \{a_1\}$, $b_2 \in B \setminus \{b_1\}$.

Since an undirected phylogenetic tree is uniquely determined by its splits, this lemma also implies that the tree-of-blobs of a semi-directed network $\mathcal{N}$ can be distinguished by the quarnet-splits of $\mathcal{N}$. In other words, two networks with the same set of quarnet-splits must have the same tree-of-blobs. Consequently, we immediately get a slow $\mathcal{O}(2^n \cdot n^2)$ algorithm for free that reconstructs the tree-of-blobs from the quarnet-splits of $\mathcal{N}$: using Lemma 1, we check for all $\mathcal{O}(2^n)$ partitions of $\mathcal{X}$ whether it is a split of $\mathcal{N}$ (and thus of $\mathcal{T}(\mathcal{N})$) in $\mathcal{O}(n^2)$ time. From these splits we can then build the tree-of-blobs.

Towards the next theorem we introduce a few notions, some of which are generalized from [12]. Let $\mathcal{B}$ be an internal blob of a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and let v be the internal vertex of $\mathcal{T}(\mathcal{N})$ that represents it. Let $u_1w_1, \ldots, u_sw_s$ be the cut-edges of $\mathcal{N}$ incident to $\mathcal{B}$ (with the u_i in $\mathcal{B}$). The *partition induced by* $\mathcal{B}$ (or by v) is the partition $X_1|\ldots|X_s$ of $\mathcal{X}$ such that $x \in X_i$ if and only if x is separated from $\mathcal{B}$ by u_iw_i. Given a reticulation vertex r of $\mathcal{B}$, a set X_i is *below the reticulation* r if there exists a partially directed path from r to w_i and we then also say that x is *below the reticulation* r for any $x \in X_i$. A set $D \subseteq \mathcal{X}$ is a *distinguishing set of* $\mathcal{B}$ if it contains at least one leaf below each reticulation of $\mathcal{B}$ and it contains at least one leaf of three different sets X_i. As an example, consider the blob $\mathcal{B}$ of the semi-directed network $\mathcal{N}|_{\mathcal{Y}}$ from Fig. 3 (excluding the grey edge and the grey leaf x). The partition induced by this blob is $\{a_1, a_2, a_3, a_4\}|\{b_1\}|\{b_2\}|\{b_3, b_4\}|\{b_5\}|\{b_6\}$, while b_3 and b_4 are below one reticulation of $\mathcal{B}$ and b_6 is below the other reticulation. The sets $\{a_1, b_3, b_6\}$ and $\{b_1, b_4, b_6\}$ are examples of distinguishing sets of this blob.

We are now ready to prove the theorem lying at the heart of the algorithms in this paper. In particular, it suggests which quarnet-splits are important when we want to add a new leaf to the tree-of-blobs of a subnetwork (see again Fig. 3 for an illustration of the theorem). Note that we can always set B' equal to $B \setminus \{b_1\}$ in this theorem if we have no further information on the reticulations of the network. Furthermore, in a semi-directed level-ℓ network, there is always a distinguishing set of size at most $\max\{3, \ell\}$ for any blob. This will be crucial for our level-1 reconstruction algorithm in Section 5. Recall that in the next theorem, by definition, the vertex v is on the side B of the split $A|B$.

Theorem 2. *Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$, let $\mathcal{Y} \subset \mathcal{X}$ and $x \in \mathcal{X} \setminus \mathcal{Y}$. Let $A|B$ be a split in $\mathcal{N}|_{\mathcal{Y}}$ induced by the cut-edge uv such that $|B| \geq 2$ and v is in a blob $\mathcal{B}$ of $\mathcal{N}|_{\mathcal{Y}}$. Let $a_1 \in A$, $b_1 \in B$ be arbitrary and let $B' \subseteq B \setminus \{b_1\}$ be such that $\{a_1, b_1\} \cup B'$ is a distinguishing set of $\mathcal{B}$, then the following are equivalent*

(i) $A \cup \{x\}|B$ is a split in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$;

(ii) $a_1x|b_1b_2$ is a quarnet-split of $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ for all $b_2 \in B'$.

Proof. That (i) implies (ii) follows readily from Lemma 1 and Observation 1. For the other direction, let $A|B_1|\ldots|B_s$ be the partition induced by $\mathcal{B}$ (with $a_1 \in A$ and $b_1 \in B_1$). Since $\mathcal{N}$ has no 2-blobs, we have that $s \geq 2$ and thus a distinguishing set of $\mathcal{B}$ exists. Let $B' = \{b_2, \ldots, b_t\}$ be such that $D = \{a_1, b_1\} \cup B'$ contains as few leaves as possible, while still being a distinguishing set of $\mathcal{B}$. This implies that every $b_i \in B'$ is in a different B_j. Without loss of generality, we can then assume that $b_i \in B_i$ for all $1 \leq i \leq t \leq s$. Now suppose that $a_1x|b_1b_i$ is a split in $\mathcal{N}|_{\{a_1,b_1,b_i,x\}}$ for all $b_i \in B'$. We will prove that $a_1x|b_1b_*$ is also a quarnet-split of $\mathcal{N}$ for all $b_* \in B \setminus \{b_1, \ldots, b_t\}$. To this end, let $b_* \in B \setminus \{b_1, \ldots, b_t\}$ be arbitrary. We denote by $\mathcal{B}'$ the blob of $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ corresponding to the (possibly smaller) blob $\mathcal{B}$ in $\mathcal{N}|_{\mathcal{Y}}$ and let R_* be the (possibly empty) set of reticulations in the blob $\mathcal{B}'$ such that b_* is below them in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$. We first prove the following claim and then continue the proof of the theorem by considering four cases.

Claim. *There is some $d_* \in D \cup \{x\}$ with the property that d_* is below all the reticulations of R_* in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$.*

Proof of Claim. If $|R_*| = 0$, the claim is trivial, so we assume that $|R_*| > 0$. As with leaves, we say that a reticulation r_2 is *below* another reticulation r_1 if there exists a partially directed path from r_1 to r_2. It suffices to show that there is some $r \in R_*$ below all reticulations in $R_* \setminus \{r\}$, since we can then simply let $d_* \in D \cup \{x\}$ be the leaf below r (which exists because D is a distinguishing set of $\mathcal{B}$, so $D \cup \{x\}$ is a distinguishing set of $\mathcal{B}'$). Towards a contradiction assume that there is no reticulation r below all other reticulations in $R_* \setminus \{r\}$. Then, there are at least two reticulations r_1 and r_2 with no reticulation from R_* below them. But for b_* to be below both r_1 and r_2, a third reticulation has to be below both r_1 and r_2: a contradiction. □

Case 1: $b_* \in B_1 \setminus \{b_1\}$. By our assumption, we know that $a_1x|b_1b_2$ is a split in $\mathcal{N}|_{\{a_1,b_1,b_2,x\}}$. But since b_1 and b_* are both part of the set B_1, we have that $a_1x|b_2b_*$ is then also a split in $\mathcal{N}|_{\{a_1,b_2,b_*,x\}}$. Using Lemma 1, these two quarnet-splits show that $a_1x|b_1b_2b_*$ is a split in $\mathcal{N}|_{\{a_1,b_1,b_2,b_*,x\}}$, which means that $a_1x|b_1b_*$ is a split in $\mathcal{N}|_{\{a_1,b_1,b_*x\}}$.

Case 2: $b_* \in B_i \setminus \{b_i\}$ *with* $2 \leq i \leq t$. By our assumption, we know that $a_1x|b_1b_i$ is a split in $\mathcal{N}|_{\{a_1,b_1,b_i,x\}}$. Clearly, since $\{b_*, b_i\} \subseteq B_i$, the quarnet $\mathcal{N}|_{\{a_1,b_1,b_*,x\}}$ has $a_1x|b_1b_*$ as its split.

Case 3: $b_* \in B_i$ *with* $2 \leq t < i \leq s$ *and* $d_* \in B' \subseteq D \cup \{x\}$. Let $b' = d_*$ and recall that by our assumption, $a_1x|b_1b'$ is a split in $\mathcal{N}|_{\{a_1,b_1,b',x\}}$. By definition, this means that there is some non-empty set of edges E in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ that are part of every up-down path between leaves from $\{a_1, x\}$ and $\{b_1, b'\}$. Towards a contradiction, suppose that $a_1x|b_1b_*$ is not a split in $\mathcal{N}|_{\{a_1,b_1,b_*,x\}}$, which by Lemma 1 means that $a_1x|b_1b'b_*$ is also not a split in $\mathcal{N}|_{\{a_1,b_1,b',b_*,x\}}$. Thus, the addition of b_* to $\mathcal{N}|_{\{a_1,b_1,b',x\}}$ allows for bypassing the edges in E when using up-down paths to connect the leaves in $\{a_1, x\}$ with those in $\{b_1, b'\}$. The only new ways to connect leaves $\{a_1, x\}$ with $\{b_1, b'\}$ when taking the subnetwork induced by $\{a_1, b_1, b', b_*, x\}$ (instead of $\{a_1, b_1, b', x\}$) is that in $\mathcal{N}|_{\{a_1,b_1,b',b_*,x\}}$ we can first take an up-down path from $\{a_1, x\}$ to b_* and then one from b_* to $\{b_1, b'\}$, whereas in $\mathcal{N}|_{\{a_1,b_1,b',x\}}$ this bypass is not possible and we can only take up-down paths directly from $\{a_1, x\}$ to $\{b_1, b'\}$. Now note that up-down paths were also characterized by Xu and Ané [8] as paths without *v-structures*: path segments containing both the reticulation edges corresponding to a reticulation vertex. From this characterization it becomes clear that the previously mentioned 'bypass' of the edges in E is possible in $\mathcal{N}|_{\{a_1,b_1,b',b_*,x\}}$ and not in $\mathcal{N}|_{\{a_1,b_1,b',x\}}$, because in $\mathcal{N}|_{\{a_1,b_1,b',x\}}$ the 'bypass' consists of one path directly from $\{a_1, x\}$ to $\{b_1, b'\}$ with a v-structure on it (and we can not first go to b_*). Hence, b_* is below the reticulation corresponding to the v-structure in $\mathcal{N}|_{\{a_1,x,b_1,b',b_*\}}$, while none of $\{a_1, x, b_1, b'\}$ is below this reticulation. It follows that b_* is also below some reticulation $r \in R_*$ in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$, while none of $\{a_1, x, b_1, b'\}$ is below r: a contradiction with the fact that $b' = d_*$ is below all reticulations in R_* (see the claim). Therefore, we have that $a_1x|b_1b_*$ is a split in $\mathcal{N}|_{\{a_1,x,b_1,b_*\}}$.

Case 4: $b_* \in B_i$ *with* $2 \leq t < i \leq s$ *and* $d_* \in \{a_1, b_1, x\} \subseteq D \cup \{x\}$. In this case, pick $b' = b_2 \in D$. We again have that $a_1x|b_1b'$ is a split in $\mathcal{N}|_{\{a_1,b_1,b',x\}}$. As in the previous case, assume towards a contradiction that $a_1x|b_1b_*$ is not a split in $\mathcal{N}|_{\{a_1,b_1,b_*,x\}}$. The exact same reasoning then shows that b_* is below some reticulation r of R_* in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$, while none of $\{a_1, x, b_1, b'\}$ is below r. This is again a contradiction because $d_* \in \{a_1, b_1, x\}$ is below all reticulations in R_* (see the claim). Therefore, we again have that $a_1x|b_1b_*$ is a split in $\mathcal{N}|_{\{a_1,x,b_1,b_*\}}$.

All in all, the four cases show that $a_1x|b_1b_*$ is a split in $\mathcal{N}|_{\{a_1,b_1,b_*,x\}}$ for all $b_* \in B \setminus \{b_1\}$. Since $A|B$ is a split in $\mathcal{N}|_{\mathcal{Y}}$, Lemma 1 tells us that $a_1a_2|b_1b_*$ is a split in $\mathcal{N}|_{\{a_1,a_2,b_1,b_*\}}$ for all $a_2 \in A \setminus \{a_1\}$, $b_* \in B \setminus \{b_1\}$. Together, we thus get that $a_1a_*|b_1b_*$ is a split in $\mathcal{N}|_{\{a_1,a_*,b_1,b_*\}}$ for all $a_* \in (A \cup \{x\}) \setminus \{a_1\}$ and $b_* \in B \setminus \{b_1\}$. So, again by Lemma 1, $A \cup \{x\}|B$ is a split in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$. This proves the backward direction for any B' such that D has as few leaves as possible. It follows trivially that it also holds for any other allowed set B'. □

4. Reconstructing a tree-of-blobs from quarnet-splits

In this section we will describe an algorithm that reconstructs the tree-of-blobs of a semi-directed network $\mathcal{N}$ on $\mathcal{X}$, using only its quarnet-splits. Our approach to achieve this will be to 'grow' our tree-of-blobs one leaf at a time. In particular, we will maintain the property that every intermediate tree on leaf set $\mathcal{Y} \subseteq \mathcal{X}$ will be the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$. As mentioned in Section 2, this tree can be more refined than $\mathcal{T}(\mathcal{N})$. Thus, we might need to contract parts of the tree before attaching a new leaf. Where to attach this leaf and what parts to contract will be based on the splits of the tree-of-blobs, which can be determined with Theorem 2 and the quarnet-splits. To this end, we need the following concepts.

Definition 3. Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and $\mathcal{Y} \subset \mathcal{X}$, let uv be an edge in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ inducing the split $A|B$ and $x \in \mathcal{X} \setminus \mathcal{Y}$. The edge uv is one of the following

- a *strong stem edge (for x)*, if $A|B \cup \{x\}$ and $A \cup \{x\}|B$ are splits in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$;
- a *weak stem edge (for x)*, if neither $A|B \cup \{x\}$ nor $A \cup \{x\}|B$ are splits in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$;
- a *pointing edge (for x) with orientation uv*, if $A|B \cup \{x\}$ is a split in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ but $A \cup \{x\}|B$ is not;
- a *pointing edge (for x) with orientation vu*, if $A|B \cup \{x\}$ is not a split in $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ but $A \cup \{x\}|B$ is.

An internal vertex v of $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ is a *stem vertex (for x)* if all its incident edges uv in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ are pointing edges for x with orientation uv. A subtree of $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ is a *weak stem subtree (for x)* if it is a maximal subtree containing only weak stem edges for x.

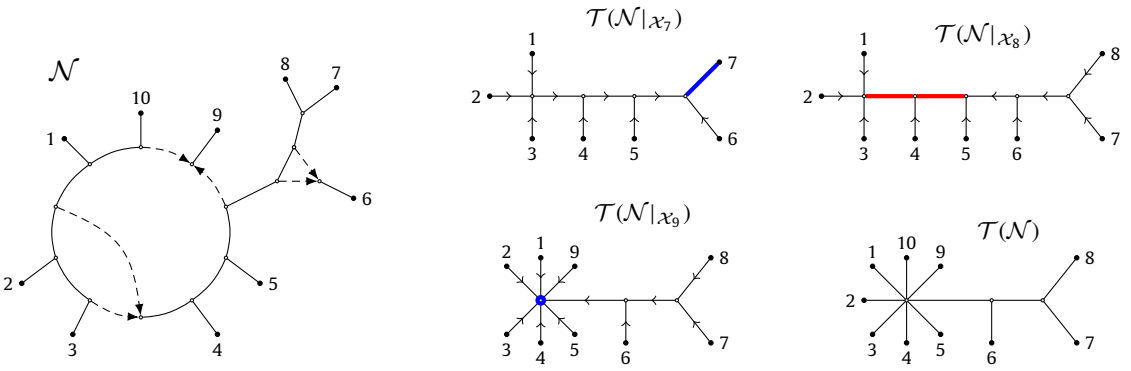

Fig. 4. A semi-directed level-2 network $\mathcal{N}$ on $\mathcal{X} = \{1, \ldots, 10\}$. On the right we have the tree-of-blobs $\mathcal{T}(\mathcal{N})$ of the whole network and three trees-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{X}_i})$, with $\mathcal{X}_i$ the set of the first i leaves of $\mathcal{X}$. In each of the three intermediate trees-of-blobs, the oriented edges indicate pointing edges. The stems of the trees-of-blobs are highlighted and are in order: a strong stem edge in thick blue, a weak stem subtree (in this case a path) in thick red, and a stem vertex in thick blue. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

If the leaf x is clear from the context, we often omit 'for x' from these descriptions. We use the terms 'pointing' and 'orientation' to emphasize that the pointing edges are not actual directed edges in the tree-of-blobs. To illustrate these different types of edges and vertices we refer to Fig. 4. One can check that the edges and vertices in the depicted trees-of-blobs exactly follow the previous definition. What stands out from the figure is that the highlighted edges and vertex seem to be the focal points when obtaining $\mathcal{T}(\mathcal{N}|_{\mathcal{X}_{i+1}})$ from $\mathcal{T}(\mathcal{N}|_{\mathcal{X}_i})$. This motivates the next definition of a *stem* (not to be confused with its unrelated namesake from [23]).

Definition 4 *(Stem).* Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$ and $\mathcal{Y} \subset \mathcal{X}$ with $x \in \mathcal{X} \setminus \mathcal{Y}$, a *stem (for x)* of $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ is a strong stem edge for x, stem vertex for x, or weak stem subtree for x.

Fig. 4 suggests that there is a unique stem and that the pointing edges are oriented towards this stem. We prove this in the next lemma and therefore we thus refer to *the* stem from now on. Furthermore, as the name suggests, the unique stem is the location where we attach the new leaf x to the tree-of-blobs. Exactly how this needs to be done is also outlined in the next lemma. It is enough to consider the following two attachments for a given leaf x. By *attaching x to an edge uv*, we mean subdividing uv with a vertex y to create two new edges uy and yv, and then adding a new edge yx. Similarly, *attaching x to a vertex v* means adding the edge vx.

Lemma 3. *Let $\mathcal{N}$ be a semi-directed network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$. Given the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ and some $x \in \mathcal{X} \setminus \mathcal{Y}$, there is a unique stem and the orientation of any pointing edge in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ is towards this stem. Furthermore,*

(a) if the stem is a strong stem edge, then the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ can be obtained from $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ by attaching x to this edge;
(b) if the stem is a weak stem subtree, then the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ can be obtained from $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ by contracting this subtree to a single vertex and attaching x to it;
(c) if the stem is a stem vertex, the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ can be obtained from $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ by attaching x to this vertex.

Proof. We let the blob $\mathcal{B}$ of $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ be the blob adjacent to x, i.e. the blob with whom x shares an incident edge. $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ contains no 2-blobs, so when taking the subnetwork of $\mathcal{N}|_{\mathcal{Y}\cup\{x\}}$ induced by $\mathcal{Y}$ we can just delete the leaf x. Then, we only need to do some operations on the blob $\mathcal{B}$. The other parts of the network remain the same. We distinguish three mutually exclusive cases covering what happens with $\mathcal{B}$.

(a): $\mathcal{B}$ completely disappears. This means that the deletion of x made $\mathcal{B}$ into a 2-blob, which was then subsequently suppressed. Then, there is a cut-edge e in $\mathcal{N}|_{\mathcal{Y}}$ that is the result of this suppression. Since all other parts of the network remain the same, we can obtain $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ from $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ by attaching x to the unique edge uv that induces the same split $A|B$ as the cut-edge e. Then, $A|B \cup \{x\}$ and $A \cup \{x\}|B$ are splits in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$, so uv is a strong stem edge. Since the rest of the tree $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ remains the same as in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$, it is easy to check that all other edges in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ are pointing edges oriented towards uv.

(b): $\mathcal{B}$ splits up into multiple blobs $\mathcal{B}'_i$. Then, the blobs $\mathcal{B}'_i$ are connected to each other. So, the vertices in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ that represent these blobs form a subtree T. Again, since all other parts of the network remain the same, we can obtain $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ from $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ by contracting the edges in T and then attaching x to the created vertex. Furthermore, if uv is an edge in T that induces the split $A|B$, then neither $A|B \cup \{x\}$ nor $A \cup \{x\}|B$ can be a split in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$. Thus, every such edge uv is a weak stem edge and so T is a weak stem subtree. By similar reasoning as before, all other edges in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ are pointing edges oriented towards T.

(c): $\mathcal{B}$ becomes a single blob $\mathcal{B}'$. Then, we can obtain $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}})$ from $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ by attaching x to the vertex v representing $\mathcal{B}'$. Again, since all other parts of the network (and thus of the tree-of-blobs) remain the same, all edges in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ are pointing edges oriented towards v, showing that v is a stem vertex. □

With Lemma 3 we now know how to attach a leaf to a stem (see Fig. 4 for examples). Thus, we have reduced the problem of reconstructing a tree-of-blobs from quarnet-splits to finding the stem in a tree-of-blobs. The following corollary of Theorem 2 shows how this is done in quadratic time.

Corollary 4. *Let $\mathcal{N}$ be a semi-directed network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$ contain k leaves. Given the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$, some $x \in \mathcal{X} \setminus \mathcal{Y}$ and the quarnet-splits of $\mathcal{N}$, one can find the stem for x of $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ in $\mathcal{O}(k^2)$ time using $\mathcal{O}(k^2)$ quarnet-splits of $\mathcal{N}$.*

Proof. We know from the previous lemma that there is a unique stem in $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$. To find this stem, it is enough to check for each of the at most $\mathcal{O}(k)$ edges in the tree-of-blobs whether it is a strong stem edge, weak stem edge, or pointing edge. If such an edge induces a split $A|B$ with $|B| = 1$, this is trivial since then $A \cup \{x\}|B$ is always a split. Otherwise, if $|B| \geq 2$, we can use Theorem 2 to determine whether $A \cup \{x\}|B$ is a split in $\mathcal{O}(k)$ time, using $\mathcal{O}(k)$ quarnet-splits of $\mathcal{N}$. In particular, we arbitrarily pick $a_1 \in A$, $b_1 \in B$ and then check if $a_1x|b_1b_2$ is a quarnet-split for all $b_2 \in B' = B \setminus \{b_1\}$. So, we can always determine whether $A \cup \{x\}|B$ is a split. Hence, we know in $\mathcal{O}(k)$ time (using $\mathcal{O}(k)$ quarnet-splits) whether the edge is a strong stem edge, weak stem edge, or pointing edge, which proves the result. □

In Algorithm 1, we can now present a concise formulation of how to reconstruct the tree-of-blobs of a semi-directed network. Fig. 4 again serves as an example of a few iterations of this algorithm. The time complexity of the algorithm is stated in Theorem 5, whose proof follows directly from the two previous results. As mentioned in Section 2, the quarnet-splits that are used as input for Algorithm 1 can also be obtained from the displayed quartets of a semi-directed network.

Algorithm 1: Reconstruction of the tree-of-blobs of a semi-directed network.

Input: quarnet-splits of a semi-directed network $\mathcal{N}$ on $\mathcal{X} = \{x_1, \ldots, x_n\}$
Output: tree-of-blobs of $\mathcal{N}$

```
X_i ← {x_1, ..., x_i} for all i ∈ {1, ..., n}
T(N|_{X_2}) ← single edge {x_1, x_2}
for i ∈ {3, ..., n} do                                   // if n = 2, {3, ..., n} = ∅
    find the stem of T(N|_{X_{i-1}}), using Corollary 4 and the quarnet-splits of N
    T(N|_{X_i}) is constructed from T(N|_{X_{i-1}}) as described in Lemma 3
return T(N|_{X_n})
```

Theorem 5. *Given the quarnet-splits of a semi-directed network $\mathcal{N}$ on $\mathcal{X} = \{x_1, \ldots, x_n\}$, Algorithm 1 reconstructs the tree-of-blobs of $\mathcal{N}$ in $\mathcal{O}(n^3)$ time using $\mathcal{O}(n^3)$ quarnet-splits of $\mathcal{N}$.*

We refer to Algorithm 1 as a *tree-of-blobs reconstruction algorithm*: an algorithm that reconstructs the tree-of-blobs of a given semi-directed network, using only its quarnet-splits. Any tree-of-blobs reconstruction algorithm needs to use $\Omega(n)$ quarnet-splits since it needs to consider every leaf at least once. An interesting question that arises, is whether any tree-of-blobs reconstruction algorithm needs $\Omega(n^3)$ quarnet-splits, which would show that Algorithm 1 is optimal. Although we are not able to answer this question affirmatively, we do provide a worst-case lower bound of $\Omega(n \log n + \ell \cdot n)$ quarnet-splits for level-ℓ networks. This simplifies to $\Omega(n^2)$ quarnet-splits for networks of unbounded level (because then ℓ could be $\Omega(n)$, see e.g. Fig. 5), narrowing the gap to $\Theta(n)$.

Proposition 6. *Given the quarnet-splits of a semi-directed level-ℓ network $\mathcal{N}$ on $\mathcal{X} = \{x_1, \ldots, x_n\}$, any algorithm using only quarnet-splits to reconstruct the tree-of-blobs of $\mathcal{N}$ needs to use $\Omega(n \log n + \ell \cdot n)$ quarnet-splits.*

Proof. The first part of the bound is based on reasoning from [39], who provide a lower bound on the number of triplets (3-leaf subtrees) that need to be used for tree reconstruction. They show that there are $2^{\Omega(n \log n)}$ possible binary phylogenetic trees on n leaves. This is also a lower bound on the number of trees-of-blobs of all n-leaf semi-directed strict level-ℓ networks. Clearly, there are only four possible quarnet-splits for a given set of four leaves (including the possibility that a quarnet has no non-trivial split). Then, by an information-theoretic argument, any tree-of-blobs reconstruction algorithm must use $\log_4(2^{\Omega(n \log n)}) = \Omega(n \log n)$ quarnet-splits in the worst case. This shows the first part of the bound for any ℓ.

Consider the semi-directed strict level-ℓ network $\mathcal{N}$ with $p \geq 2$ internal blobs from Fig. 5. For all $k \in \{1, \ldots, p\}$ and $i, j \in \{1, \ldots, \ell\}$ with $i \neq j$, let $\mathcal{N}_{i,j,k}$ be the second network from Fig. 5 which is also strict level-ℓ for $p \geq 2$. Towards the second bound, suppose that $\ell \geq 2$. For simplicity, we also assume that $n \geq 4$ is a multiple of ℓ such that $n = p \cdot \ell$. For values of n which are not multiples of ℓ the following exposition can trivially be adapted by letting one of the blobs be bigger (but

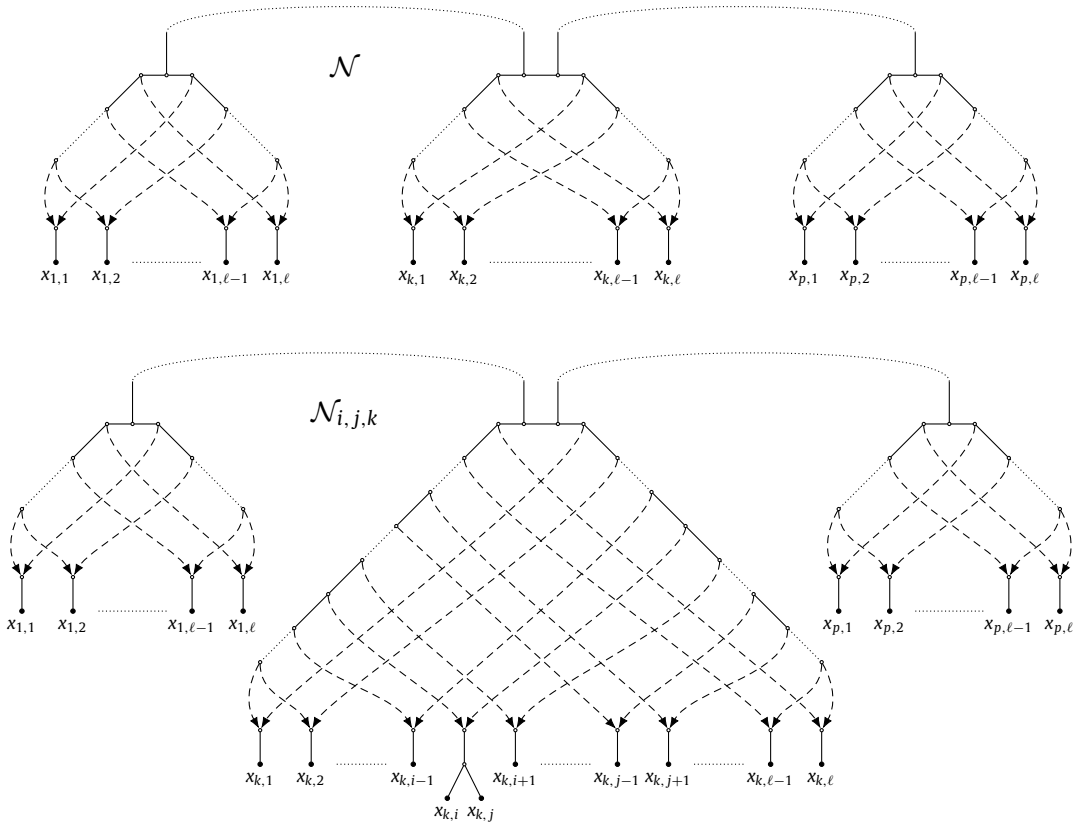

Fig. 5. A semi-directed strict level-ℓ network $\mathcal{N}$ with p internal blobs and a set of semi-directed strict level-ℓ networks $\mathcal{N}_{i,j,k}$ on the same leaf set with $p+1$ internal blobs, where $i, j \in \{1, \ldots, \ell\}$ such that $i \neq j$ and $k \in \{1, \ldots, p\}$. All networks have $n = \ell \cdot p$ leaves, where $\ell \geq 2$ and $p \geq 2$.

with a similar structure) to accommodate for the remainder of the leaves. Every quarnet-split that does not contain both $x_{k,i}$ and $x_{k,j}$ is the exact same in $\mathcal{N}$ and $\mathcal{N}_{i,j,k}$. (Note that the quarnets might differ since i is not necessarily $j-1$, but the quarnet-splits do not.) Since $\mathcal{N}$ has a different tree-of-blobs than the networks $\mathcal{N}_{i,j,k}$, every tree-of-blobs reconstruction algorithm must consider at least one quarnet-split for every combination of $k \in \{1, \ldots, p\}$ and $i \neq j \in \{1, \ldots, \ell\}$. Otherwise, such an algorithm can never unambiguously construct the tree-of-blobs of $\mathcal{N}$. This means that such an algorithm needs to consider at least $\frac{1}{2} \cdot p \cdot \ell \cdot (\ell - 1) = \frac{1}{2} \cdot n \cdot (\ell - 1)$ quarnet-splits of $\mathcal{N}$. The bound then immediately follows. □

5. Reconstructing a level-1 network from quarnets

Whereas the previous section was aimed at reconstructing the tree-of-blobs of a general semi-directed network, this section is solely devoted to semi-directed level-1 networks. With this restriction we aim for more than in the previous section: the reconstruction of the full network. We first show how to construct a canonical form of such a network from $\mathcal{O}(n \log n)$ of its quarnet-splits, after which we use the full quarnets to reconstruct the whole network.

5.1. Canonical level-1 networks

Recall from Section 2 that every internal blob in a semi-directed level-1 network is either a single vertex or a k-cycle. We say that such a cycle is a *large cycle* if it is neither a triangle nor a 4-cycle. An edge between two adjacent non-reticulation vertices u and v of a cycle in a (possibly non-binary) semi-directed level-1 network is a *cycle contraction edge* if u or v is adjacent to the reticulation vertex of the cycle and if u and v are both degree-3 vertices. Clearly, every 4-cycle and every large cycle of a binary semi-directed level-1 network has two distinct cycle contraction edges: one on each side of the reticulation. We are now ready to define the canonical form that is of interest in this section and we refer to the top part of Fig. 6 for an example.

Definition 5 *(Canonical form).* Given a semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X}$, the *canonical form* $\mathcal{N}^c$ of $\mathcal{N}$ is obtained by contracting every triangle and 4-cycle to a single vertex; and by contracting the cycle contraction edges of every large cycle.

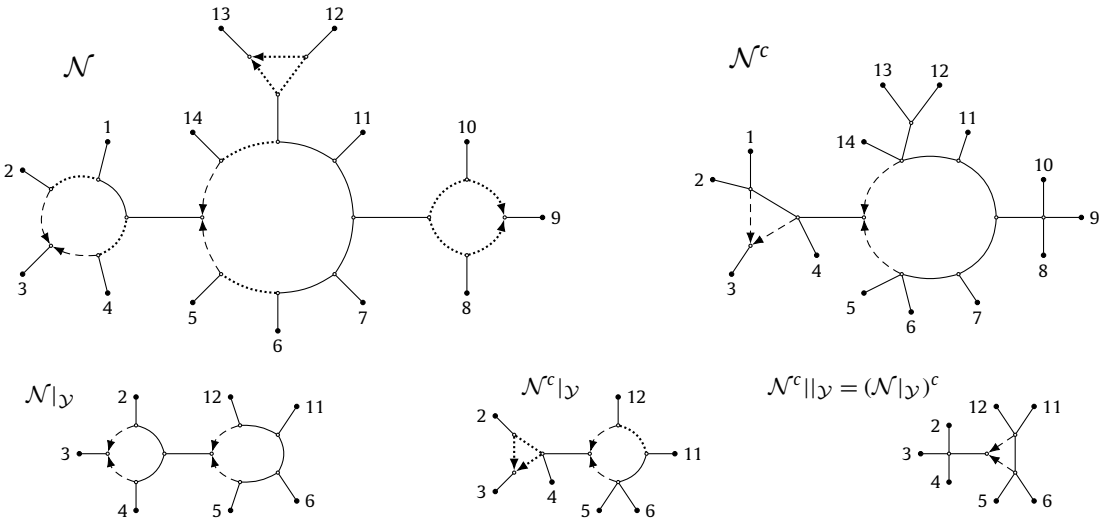

Fig. 6. *Top:* A semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X} = \{1, \ldots, 14\}$ and its canonical form $\mathcal{N}^c$ obtained by contracting all dotted edges. *Bottom:* The subnetwork of $\mathcal{N}$ induced by $\mathcal{Y} = \{2, 3, 4, 5, 6, 11, 12\}$, the subnetwork of $\mathcal{N}^c$ induced by $\mathcal{Y}$, and the canonical subnetwork of $\mathcal{N}^c$ induced by $\mathcal{Y}$ obtained from $\mathcal{N}^c|_\mathcal{Y}$ by contracting all dotted edges, which either form a 4-blob, or they are a cycle contraction edge between two degree-3 vertices. Note that this last canonical network is also the canonical form of $\mathcal{N}|_\mathcal{Y}$.

Sometimes we say that $\mathcal{N}^c$ is a *canonical (level-1) network* when we mean that it is the canonical form of a semi-directed level-1 network $\mathcal{N}$. Note that a canonical network is also a semi-directed level-1 network, albeit with a maximum degree of four, instead of three. Hence, we straightforwardly extend some definitions, such as blobs and trees-of-blobs, in a trivial way to canonical networks. It is easy to see that the canonical form of a semi-directed level-1 network can be viewed as a structure that is more refined than the tree-of-blobs, yet less refined than the original network. Moreover, the tree-of-blobs of the canonical form is equal to the tree-of-blobs of the original network. It will follow from Theorem 13 that, as with the tree-of-blobs, the canonical form $\mathcal{N}^c$ of a network $\mathcal{N}$ can be reconstructed from the quartet-splits of $\mathcal{N}$. As we will see in Section 5.3, it is even the most refined network one can construct from quartet-splits.

In this section, we are mostly concerned with the canonical form of a subnetwork, i.e. $(\mathcal{N}|_\mathcal{Y})^c$ for some $\mathcal{Y} \subseteq \mathcal{X}$. Unfortunately, we might have that $(\mathcal{N}|_\mathcal{Y})^c \neq \mathcal{N}^c|_\mathcal{Y}$ (consider for example the set $\mathcal{Y}$ and the two networks $(\mathcal{N}|_\mathcal{Y})^c$ and $\mathcal{N}^c|_\mathcal{Y}$ in Fig. 6). Thus, it matters whether we first induce a subnetwork, or first create the canonical form. Hence, we define the following procedure for canonical networks which does have the desired property (see Fig. 6 for an example). Note that a 3-blob (resp. 4-blob) in a canonical network need not necessarily be a triangle (resp. 4-cycle) due to the possible degree-4 vertices.

Definition 6 *(Canonical subnetwork).* Given a canonical network $\mathcal{N}^c$ on $\mathcal{X}$ and some $\mathcal{Y} \subseteq \mathcal{X}$ with $|\mathcal{Y}| \geq 2$, the *canonical subnetwork* of $\mathcal{N}^c$ induced by $\mathcal{Y}$ is the canonical network $\mathcal{N}^c||_\mathcal{Y}$ obtained from $\mathcal{N}^c$ by first taking the subnetwork of $\mathcal{N}^c$ induced by $\mathcal{Y}$, followed by contracting every 3-blob and 4-blob to a single vertex, and by contracting every cycle contraction edge.

Indeed, this canonical subnetwork is equivalent to first inducing a subnetwork of the original network, and then creating the canonical form. This follows from the fact that every 3-blob or 4-blob in the canonical subnetwork of a canonical network must have been a triangle or a 4-cycle in the subnetwork of the original network. These blobs should thus be identified by a single vertex. Furthermore, the cycle contraction edges in $\mathcal{N}^c|_\mathcal{Y}$ correspond to cycle contraction edges in the subnetwork $\mathcal{N}|_\mathcal{Y}$ of the original binary network, since - by definition - the cycle contraction edges in $\mathcal{N}^c|_\mathcal{Y}$ are only between degree-3 vertices. We present the formal statement in the following observation, yet omit its rather straightforward proof which follows from the reasoning above. Throughout the rest of this section we will frequently, yet implicitly, use this observation.

Observation 2. Given a semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X}$ and its canonical form $\mathcal{N}^c$, if $\mathcal{Y} \subseteq \mathcal{X}$ with $|\mathcal{Y}| \geq 2$, then $(\mathcal{N}|_\mathcal{Y})^c = \mathcal{N}^c||_\mathcal{Y}$.

5.2. Reconstructing a canonical network

Our strategy to construct a canonical network will be similar to the strategy that was used to construct trees-of-blobs. However, instead of contracting parts of the intermediate trees-of-blobs (which signifies the addition of a reticulation), we

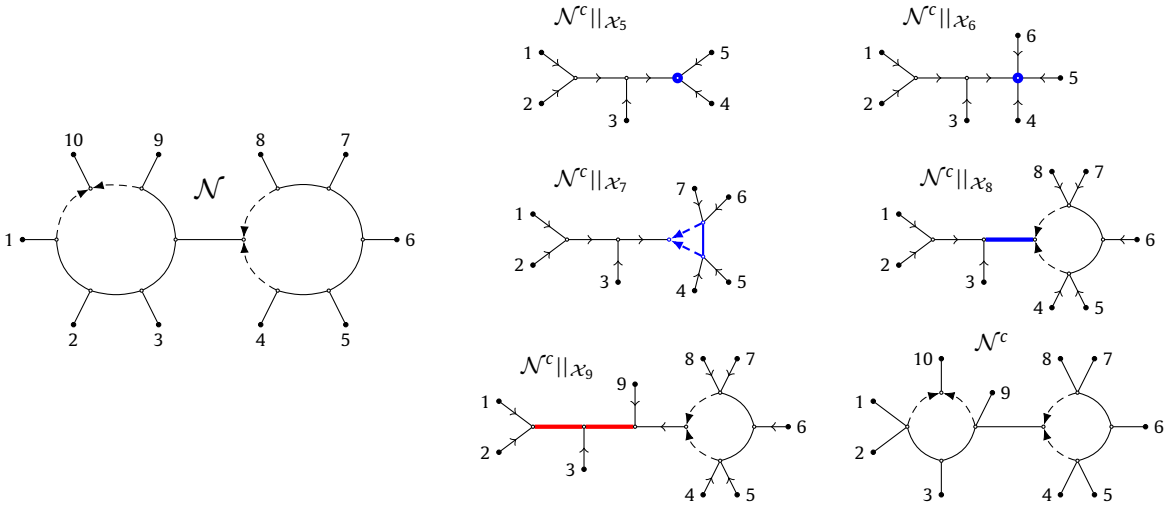

Fig. 7. A semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X} = \{1, \ldots, 10\}$. On the right, we have the canonical network $\mathcal{N}^c$ and five canonical forms of subnetwork $\mathcal{N}^c||_{\mathcal{X}_i}$, with $\mathcal{X}_i$ the set of the first i leaves of $\mathcal{X}$. In each of the five intermediate canonical networks the oriented edges (with the arrowhead in the middle) indicate a pointing edge. The canonical stems of the canonical networks are highlighted and are in order: a stem vertex of degree-3 in thick blue, a stem vertex of degree-4 in thick blue, a stem cycle in thick blue, a strong stem edge in thick blue, and a weak stem path in thick red. Note that the dashed edges with the arrowhead at the end of the edge are reticulation edges and not pointing edges.

add a reticulation. The reduction in the number of used quartet-splits from $\mathcal{O}(n^3)$ to $\mathcal{O}(n \log n)$ has two reasons. On the one hand, we get a reduction of $\Theta(n)$ by using Theorem 2 to our advantage. Specifically, we know where the reticulations are in a canonical network (except for those in triangles and 4-cycles) and we can thus find a constant size distinguishing set. Getting the logarithmic factor requires some more care but is loosely based on recursively picking cut-edges that create 'balanced' cuts in the network. We now start by transferring the concept of a stem to canonical networks.

Since the tree-of-blobs of a canonical network is equal to the tree-of-blobs of the original network, we can generalize the notions from Definition 3 to canonical networks in a straightforward way. To this end, let $\mathcal{N}^c$ be a canonical network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$ with $x \in \mathcal{X} \setminus \mathcal{Y}$. Then, a cut-edge of $\mathcal{N}^c||_{\mathcal{Y}}$ is a *strong stem edge / weak stem edge / pointing edge (for x)* if the unique edge that induces the same split in its tree-of-blobs $\mathcal{T}(\mathcal{N}^c||_{\mathcal{Y}}) = \mathcal{T}(\mathcal{N}|_{\mathcal{Y}})$ is a strong stem edge / weak stem edge / pointing edge for x. Similarly, we let a blob in $\mathcal{N}^c||_{\mathcal{Y}}$ be a *stem blob (for x)* if the vertex that represents it in its tree-of-blobs is a stem vertex for x. Specifically, if such a blob is a single vertex it is a *stem vertex (for x)*, while it is a *stem cycle (for x)* if the blob is a cycle. A *cut-path* is a path that only contains cut-edges of $\mathcal{N}^c||_{\mathcal{Y}}$, and such a cut-path is a *weak stem path (for x)* if the corresponding path in its tree-of-blobs is a weak stem subtree for x. As was the case for trees-of-blobs, we often omit 'for x' if the leaf x is clear from the context. We can now define the analogue of a stem for canonical networks.

Definition 7 *(Canonical stem).* Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$. Given the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ and some $x \in \mathcal{X} \setminus \mathcal{Y}$, a *canonical stem (for x)* of $\mathcal{N}^c||_{\mathcal{Y}}$ is a strong stem edge for x, stem vertex for x, stem cycle for x, or weak stem path for x.

We refer to Fig. 7 for examples of the different types of canonical stems. Analogous to Lemma 3, the next lemma shows that a canonical network has a unique canonical stem and that all pointing edges are oriented towards it. The lemma also details how to attach a leaf x to a stem, for which we need the following new attachment operation. When we *attach x to two different vertices u and v*, we mean that we add a new reticulation vertex w with directed edges uw and vw, and then attach x to w. The different ways to attach the leaves can also be seen in Fig. 7. Note that the lemma does not cover the case when the canonical stem is a stem cycle. Then, we need to attach the leaf somewhere on the cycle, but this cannot be done using $\mathcal{O}(1)$ quartet-splits. Later in this section in Lemma 11 we will solve this issue by deleting the reticulation of the cycle, attaching the leaf, and then putting the reticulation back in.

Lemma 7. *Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$. Given the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ and some $x \in \mathcal{X} \setminus \mathcal{Y}$, there is a unique canonical stem and the orientation of any pointing edge in $\mathcal{N}^c||_{\mathcal{Y}}$ is towards this canonical stem. Furthermore,*

(a) if the canonical stem is a strong stem edge, then the canonical network $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ can be obtained from $\mathcal{N}^c||_{\mathcal{Y}}$ by attaching x to this edge;

(b) if the canonical stem is a weak stem path, then the canonical network $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ can be obtained from $\mathcal{N}^c||_{\mathcal{Y}}$ by attaching x to both end vertices of this path;

(c) if the canonical stem is a stem vertex of degree-3, the canonical network $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ can be obtained from $\mathcal{N}^c||_{\mathcal{Y}}$ by attaching x to this vertex;

(d) if the canonical stem is a stem vertex of degree-4, the canonical network $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ can be obtained from $\mathcal{N}^c||_{\mathcal{Y}}$ in constant time using at most five quarnet-splits of $\mathcal{N}$.

Proof. From Lemma 3 we know that the tree-of-blobs $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}}) = \mathcal{T}(\mathcal{N}^c||_{\mathcal{Y}})$ contains a unique stem and all other edges are pointing edges oriented towards this stem. If the stem in this tree-of-blobs is a strong stem edge, $\mathcal{N}^c||_{\mathcal{Y}}$ has a strong stem edge. Similarly, if this stem is a stem vertex, $\mathcal{N}^c||_{\mathcal{Y}}$ has a stem vertex or a stem cycle. Lastly, if this stem is a weak stem subtree T, we know from Lemma 3b that $\mathcal{T}(\mathcal{N}|_{\mathcal{Y}\cup\{x\}}) = \mathcal{T}(\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}})$ can be obtained by contracting T and then attaching x to it. To not contradict the fact that $\mathcal{N}^c||_{\mathcal{Y}}$ is level-1, all vertices of T must represent trivial blobs in $\mathcal{N}|_{\mathcal{Y}}$ (and thus in $\mathcal{N}^c||_{\mathcal{Y}}$) and they form a cut-path. Therefore, there is a weak stem path in $\mathcal{N}^c||_{\mathcal{Y}}$. This proves the first part of the lemma.

Part (a) (resp. part (c)) of the lemma follows directly from Lemma 3a (resp. Lemma 3c) and the fact that triangles (resp. 4-cycles) are suppressed in canonical networks.

For part (b), we already know by the previous reasoning that all vertices of the weak stem path become part of the same cycle $\mathcal{C}$ when we attach x. Thus, x is the first leaf (with respect to all other leaves in $\mathcal{Y}$) below the reticulation of $\mathcal{C}$. For canonical networks, we do not care about the ordering of the two leaves in a reticulation cycle pair. Clearly, we can then just attach x to both end vertices of the path (and direct the two new edges towards x) to create the cycle $\mathcal{C}$.

For part (d), we let $Y_1|Y_2|Y_3|Y_4$ be the partition of $\mathcal{Y}$ induced by the stem vertex v and we also pick one arbitrary leaf y_i from every Y_i. The stem vertex v represents a 4-cycle in $\mathcal{N}|_{\mathcal{Y}}$ and Lemma 3c tells us that v becomes a 5-blob in $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ with induced partition $Y_1|Y_2|Y_3|Y_4|\{x\}$. Such a 5-blob will be a triangle with five cut-edges incident to it in $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$. Then, there will be exactly one subset of four leaves from $\{y_1, y_2, y_3, y_4, x\}$ that induces a quarnet-split. The remaining fifth leaf then corresponds to the subset of leaves below the reticulation of $\mathcal{C}$, while the quarnet-split is enough to order the other leaves around the triangle. □

The next lemma, which follows from Theorem 2, is crucial for finding out whether a cut-edge is a weak/strong stem edge or a pointing edge. In contrast with the linear amount of quarnet-splits that was needed in the previous section, we need only a constant amount in the case of level-1 networks.

Lemma 8. *Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$ contain k leaves. Given a cut-edge uv in the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ and some $x \in \mathcal{X} \setminus \mathcal{Y}$, one can determine in $\mathcal{O}(k)$ time whether uv is a strong stem edge, a weak stem edge, or a pointing edge, using at most four quarnet-splits of $\mathcal{N}$.*

Proof. If the cut-edge uv induces a split $A|B$ with $|B| = 1$, we know that $A \cup \{x\}|B$ is always a split. If $|B| \geq 2$, we can use Theorem 2 to determine whether $A \cup \{x\}|B$ is a split in $\mathcal{O}(k)$ time, using at most two quarnet-splits of $\mathcal{N}$. To be precise, we can always pick the set B' in $\mathcal{O}(k)$ time such that it has size at most two. This follows from the fact that, except for the triangles and 4-cycles, we always know where the reticulation vertices are in a canonical network. Finally, reversing the roles of A and B allows us to determine what type uv is. □

Whenever we use the previous lemma and find out that a given edge is a strong stem edge, Lemma 7 describes how to add the leaf to the canonical network. If a weak stem edge is found, we first need to find the complete weak stem path before we can use Lemma 7 to attach a new leaf. The following lemma proves that this can be done using a logarithmic number of quarnet-splits, making it the first puzzle piece to get the number of quarnet-splits that our algorithm uses down to $\mathcal{O}(n \log n)$. Fig. 8 is a visualization of part of the algorithm described in the proof of the lemma.

Lemma 9. *Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$ contain k leaves. Given a weak stem edge in the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ and some $x \in \mathcal{X} \setminus \mathcal{Y}$, one can find the weak stem path in $\mathcal{N}^c||_{\mathcal{Y}}$ in $\mathcal{O}(k)$ time using $\mathcal{O}(\log k)$ quarnet-splits of $\mathcal{N}$.*

Proof. Let u_1u_2 be the given weak stem edge. By Lemma 7 there exists a unique cut-path P that is a weak stem path and that contains u_1u_2. By the same lemma, we also know that we can obtain $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ from $\mathcal{N}^c||_{\mathcal{Y}}$ by attaching x to the end vertices of P. Now let T be the unique maximal subtree of $\mathcal{N}^c||_{\mathcal{Y}}$ that contains u_1u_2 and does not contain degree-4 vertices, leaves, or vertices that are part of a cycle. Then, T is a binary tree and P is completely in T. Otherwise, attaching x to the end vertices of P would not result in a canonical network. Fig. 8 shows an example of such a tree T for a given weak stem edge. Note that finding T can be done in $\mathcal{O}(k)$ time. We also store for each edge of T what its corresponding split is in $\mathcal{N}^c||_{\mathcal{Y}}$. This allows us to say that an edge in T (or in any of its subtrees) is a strong stem edge / weak stem edge / pointing edge, when we mean that the corresponding edge in $\mathcal{N}^c||_{\mathcal{Y}}$ is a strong stem edge / weak stem edge / pointing edge.

The algorithm now continues as follows. We delete the edge u_1u_2 from T to create two smaller subtrees T_1 and T_2 with u_1 in T_1 and u_2 in T_2. Clearly, one end vertex p_1 of P must be in T_1, and the other end vertex p_2 in T_2. We will now show that one can recursively locate the vertex p_i in the tree T_i in $\mathcal{O}(k_i)$ time using $\mathcal{O}(\log k_i)$ quarnet-splits, where k_i is the number of vertices in T_i. Since knowing p_1 and p_2 is enough to find P in linear time, the result will then follow.

Whenever $k_i = 1$, we trivially know that the only vertex in T_i is p_i and we are done. Otherwise, if $k_i \geq 2$, we first find a *centroid* v of T_i: a vertex such that every component of $T_i - \{v\}$ contains at most $\frac{2}{3}k_i$ vertices. A centroid always exists

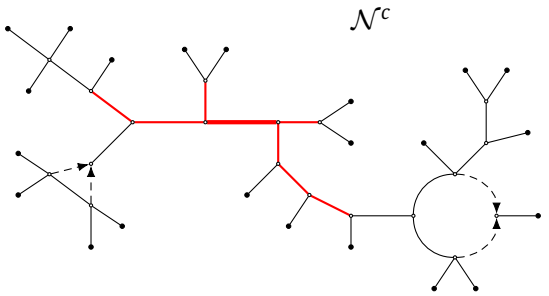

Fig. 8. A canonical network $\mathcal{N}^c$ with unlabeled leaves (indicated by the filled black vertices) and a given weak stem edge in thick red. Together with the other red edges, they induce the binary subtree T that has to contain the weak stem path. This tree is used in the algorithm from Lemma 9.

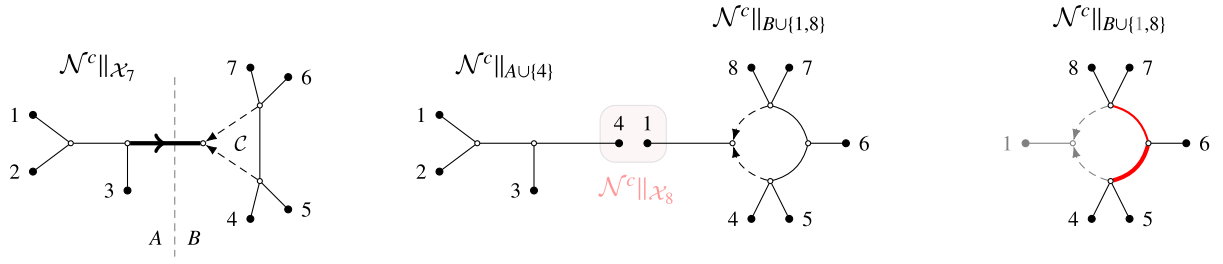

Fig. 9. *Left:* The canonical subnetwork of the canonical network $\mathcal{N}^c$ from Fig. 7 induced by its first seven leaves $\mathcal{X}_7$. The thick cut-edge is a pointing edge for leaf 8 and induces the split $A|B$ with $A = \{1, 2, 3\}$ and $B = \{4, 5, 6, 7\}$. *Middle:* The canonical subnetworks $\mathcal{N}^c||_{A\cup\{4\}}$ and $\mathcal{N}^c||_{B\cup\{1,8\}}$. Glueing $\mathcal{N}^c||_{A\cup\{4\}}$ at 4 to $\mathcal{N}^c||_{B\cup\{1,8\}}$ at 1 is done by identifying the leaves in the shaded box and then suppressing the resulting degree-2 vertex. This results in the canonical subnetwork of $\mathcal{N}^c$ induced by its first eight leaves $\mathcal{X}_8$. *Right:* The canonical subnetwork $\mathcal{N}^c||_{B\cup\{8\}}$ (excluding the grey edges and vertices) with the weak stem path for leaf 1 in red. Including the grey edges and vertices produces $\mathcal{N}^c||_{B\cup\{1,8\}}$. Note that the thick red cut-edge in $\mathcal{N}^c||_{B\cup\{8\}}$ corresponds to the single non-reticulation edge of the cycle $\mathcal{C}$ in $\mathcal{N}^c||_{\mathcal{X}_7}$, while the thinner red cut-edge is one of the four cut-edges incident to the end-points of the thick red cut-edge.

and can be found in linear time (see e.g. [39,40]). Since T_i is binary and we are in the case with $k_i \geq 2$, there is one edge vw incident to v such that both components of $T_i - \{vw\}$ have at most $\frac{2}{3}k_i$ vertices. We will now describe how to find out which of the two components contains p_i and which component we should thus recurse on. This will then prove correctness since both components contain at most $\lfloor \frac{2}{3}k_i \rfloor < k_i$ vertices. To this end, we test with Lemma 8 whether vw is a weak stem edge or a pointing edge in linear time using at most four quartet-splits. Note that by the uniqueness of the stem (see Lemma 7) vw cannot be a strong stem edge. If vw is a pointing edge, we know from Lemma 7 that its corresponding edge in $\mathcal{N}^c||_{\mathcal{Y}}$ is oriented towards P (and thus p_i). Thus, p_i is in the component of $T_i - \{vw\}$ to which vw is pointing. On the other hand, if vw is a weak stem edge, p_i can only be in the component of $T_i - \{vw\}$ that does not contain u_i (or, if we have already recursed and u_i is not in the tree any more, the component furthest away from u_i in the original tree T).

We now let $D(k')$ be the recursion depth of this algorithm when applied to a tree with k' vertices. From the above analysis it follows that $D(k') \leq D\left(\lfloor \frac{2}{3}k' \rfloor\right) + 1$ with $D(2) = 1$, which results in $D(k_i) = \mathcal{O}(\log k_i)$. Since we use $\mathcal{O}(k_i)$ time and at most four quartet-splits per recursive level, our algorithm to find p_i takes $\mathcal{O}(k_i)$ time and $\mathcal{O}(\log k_i)$ quartet-splits. □

The following result is a consequence of Lemma 7, which states that pointing edges are always oriented towards the canonical stem. Given two canonical networks $\mathcal{N}_1^c$ and $\mathcal{N}_2^c$ on leaf sets $\mathcal{X}_1$ and $\mathcal{X}_2$ with $x_1 \in \mathcal{X}_1$ and $x_2 \in \mathcal{X}_2$ such that $\mathcal{X}_1 \cap \mathcal{X}_2 \subseteq \{x_1, x_2\}$, we first define the following operation. The graph obtained by *glueing* $\mathcal{N}_1^c$ *at* x_1 *to* $\mathcal{N}_2^c$ *at* x_2 is the union of $\mathcal{N}_1^c$ and $\mathcal{N}_2^c$, where x_1 and x_2 are identified as a single vertex and the resulting degree-2 vertex is suppressed. We refer to the left and middle part of Fig. 9 to get some intuition for the glueing operation and the lemma.

Lemma 10. *Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X}$ and let $\mathcal{Y} \subset \mathcal{X}$. Let uv be a cut-edge in the canonical network inducing the split $A|B$ and let $x \in \mathcal{X} \setminus \mathcal{Y}$, $a \in A$, and $b \in B$. If uv is a pointing edge for x with orientation uv, then $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ can be constructed by glueing $\mathcal{N}^c||_{A\cup\{b\}}$ at b to $\mathcal{N}^c||_{B\cup\{a,x\}}$ at a.*

Observe that to get $\mathcal{N}^c||_{B\cup\{x,a\}}$ in the lemma, we only need to attach x to $\mathcal{N}^c||_{B\cup\{a\}}$, which in turn can be obtained from $\mathcal{N}^c||_{\mathcal{Y}}$ by identifying all vertices on the 'A-side' of the cut-edge by the single leaf a. Thus, the lemma allows us to first cut off part of the network, then attach the new leaf to the remaining part, and afterwards glue the two parts together again.

In the case where uv points directly towards a reticulation v of a cycle $\mathcal{C}$ (as is the case in Fig. 9), we can even get rid of the leaf a. That is, instead of attaching x to $\mathcal{N}^c||_{B\cup\{a\}}$, we can first attach x to $\mathcal{N}^c||_B$ and afterwards attach a to its stem again to obtain $\mathcal{N}^c||_{B\cup\{x,a\}}$ (see e.g. the right part of Fig. 9). Note that the cycle $\mathcal{C}$ does not appear any more in $\mathcal{N}^c||_B$ since

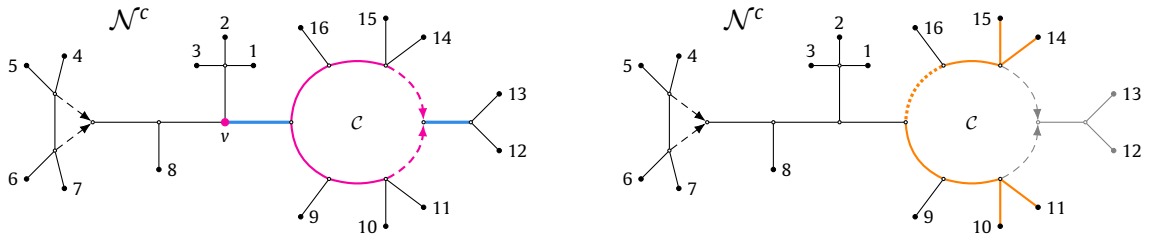

Fig. 10. A canonical network $\mathcal{N}^c$ on $\mathcal{X} = \{1, \ldots, 16\}$. *Left:* The two central blobs of $\mathcal{N}^c$: a central vertex v in pink and a central cycle $\mathcal{C}$ in pink. The left thick blue edge is a nice central edge incident to v, and the right thick blue edge is a nice central edge incident to $\mathcal{C}$. *Right:* The spine edges of the cycle $\mathcal{C}$ in orange with the dotted orange edge being a nice spine edge of $\mathcal{C}$. The filled black leaves are the leaves $\mathcal{Y} = \{1, \ldots, 11, 14, 15, 16\}$ not below the reticulation of $\mathcal{C}$. Deleting the grey edges and vertices forms the canonical subnetwork $\mathcal{N}^c||_{\mathcal{Y}}$, in which the nice spine edge is a cut-edge.

none of the leaves below its reticulation are in B. Thus, in some sense, we are 'unfolding' the cycle $\mathcal{C}$. This also solves the problem of attaching a leaf to a stem cycle mentioned in the discussion before Lemma 7. We prove that this process can be done efficiently in the following lemma and refer to Fig. 9 for an illustration of parts of the proof.

Lemma 11. *Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X}$ and let $\mathcal{Y} \subseteq \mathcal{X}$ contain k leaves. Let uv be a cut-edge in the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ inducing the split $A|B$ and let $x \in \mathcal{X} \setminus \mathcal{Y}$, $a \in A$ and $b \in B$. If uv is a pointing edge for x with orientation uv and v is the reticulation of a cycle, then $\mathcal{N}^c||_{B\cup\{a,x\}}$ can be constructed from $\mathcal{N}^c||_{B\cup\{x\}}$ in $\mathcal{O}(k)$ time using $\mathcal{O}(1)$ quartet-splits of $\mathcal{N}$.*

Proof. Let $\mathcal{C}$ be the c-cycle (with $c \geq 3$) in $\mathcal{N}^c||_{\mathcal{Y}}$ of which v is the reticulation and note that $\mathcal{C}$ (and its reticulation v) also exist in $\mathcal{N}^c||_{B\cup\{a\}}$ (which is just a canonical subnetwork of $\mathcal{N}^c||_{\mathcal{Y}}$). Since uv is a pointing edge for x in $\mathcal{N}^c||_{\mathcal{Y}}$, the leaf x is not attached somewhere below the reticulation v when constructing $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ from $\mathcal{N}^c||_{\mathcal{Y}}$ (see Lemma 7). Therefore, to construct $\mathcal{N}^c||_{B\cup\{a,x\}}$ from $\mathcal{N}^c||_{B\cup\{x\}}$ we need to attach a such that it creates a cycle with a being the only leaf below its reticulation. The resulting cycle will then be a c-cycle or $(c+1)$-cycle (depending on whether x is attached to the cycle or somewhere else). By Lemma 7, this means that we need to attach a to a weak stem path for a in $\mathcal{N}^c||_{B\cup\{x\}}$. In particular, this weak stem path has to be a path of $c-1$ or c vertices for the new cycle to be a c-cycle or $(c+1)$-cycle. We will now describe how to find this weak stem path in $\mathcal{O}(k)$ time using $\mathcal{O}(1)$ quartet-splits, which will then prove the lemma.

Note that every non-reticulation edge of $\mathcal{C}$ in $\mathcal{N}^c||_{B\cup\{a\}}$ (or equivalently in $\mathcal{N}^c||_{\mathcal{Y}}$) corresponds to a unique cut-edge in $\mathcal{N}^c||_B$ (see e.g. Fig. 9). In $\mathcal{N}^c||_{B\cup\{x\}}$ these cut-edges still exist, although one of them could have been subdivided in the case that x was attached to it. Clearly, for the resulting cycle in $\mathcal{N}^c||_{B\cup\{a,x\}}$ to be compatible with the cycle $\mathcal{C}$ in $\mathcal{N}^c||_{B\cup\{a\}}$, all of these cut-edges have to be part of the weak stem path for a. Now note that these cut-edges form a path in $\mathcal{N}^c||_{B\cup\{x\}}$ and that there are either $c-2$ or $c-1$ of them (depending on whether one of them was subdivided by an attachment of x). Therefore, we have already found a path of $c-1$ or c vertices containing only weak stem edges. Recall that we know that the weak stem path for a has length at most c. Therefore, it is sufficient to check whether the four cut-edges incident to the end-vertices of the path are weak stem edges. By Lemma 8 this takes $\mathcal{O}(k)$ time and $\mathcal{O}(1)$ quartet-splits for each of the four cut-edges. □

In light of Lemma 10 we ideally want to find cut-edges that split the leaf set in a 'balanced' way, thus cutting off roughly half of the leaves each time we recurse. This would allow us to get the desired logarithm in the number of quartet-splits that are used. Unfortunately, such an edge does not always exist (consider e.g. a canonical network with one very large cycle as its only internal blob). However, if we choose our cut-edges smartly, we can use Lemma 11 to 'unfold' such a cycle and find a 'balanced' cut-edge within the unfolded cycle. To formalize this we will introduce some new 'central' structures in canonical networks.

Given a canonical network $\mathcal{N}^c$ on $\mathcal{X}$, an internal blob $\mathcal{B}$ is a *central blob* if every component of $\mathcal{N}^c - \mathcal{B}$ contains at most half of the leaves of $\mathcal{X}$. Since a canonical network is obtained from a level-1 network, a central blob is either a *central vertex* or a *central cycle*. Given a central blob, we let a cut-edge uv incident to it be a *central edge*. Clearly, if uv is a central edge inducing the split $A|B$ (where u is in the blob and so A is on the side of the blob) then $|\mathcal{X}| \geq 3$, and so $|A| \geq 2$ and $|B| \leq \frac{1}{2}|\mathcal{X}|$. A central edge uv incident to a central blob $\mathcal{B}$ (where u is in $\mathcal{B}$) of a canonical network is *nice* if uv satisfies one of the following two statements: (i) $\mathcal{B}$ is a central vertex, $|A| \leq \frac{3}{4}|\mathcal{X}|$ and $|B| \geq 2$; (ii) $\mathcal{B}$ is a central cycle and uv is the cut-edge incident to the reticulation of this cycle. See the left part of Fig. 10 for an illustration of central blobs and central edges. Note that a canonical network does not always have both a central vertex and a central cycle.

By definition, a cycle $\mathcal{C}$ in a canonical network $\mathcal{N}^c$ will always contain two degree-4 vertices. We call the four cut-edges incident to the two degree-4 vertices the *outer spine edges* of the cycle and the non-reticulation edges within the cycle the *inner spine edges*. Together, they form the *spine edges* of $\mathcal{C}$ (see Fig. 10). If we let $\mathcal{Y} \subset \mathcal{X}$ be the set of leaves not below the reticulation of $\mathcal{C}$, then every spine edge of $\mathcal{C}$ *corresponds* to a unique cut-edge in $\mathcal{N}^c||_{\mathcal{Y}}$. The *spine split* $A'|B'$ induced by a spine edge of $\mathcal{C}$ is the split induced by the corresponding cut-edge in $\mathcal{N}^c||_{\mathcal{Y}}$. Thus, $A'|B'$ is a partition of $\mathcal{Y}$ and not of $\mathcal{X}$. A spine edge of a central cycle is *nice* if it has spine split $A'|B'$ such that $2 \leq |A'| \leq \frac{3}{4}|\mathcal{X}|$ and $2 \leq |B'| \leq \frac{3}{4}|\mathcal{X}|$ (see Fig. 10).

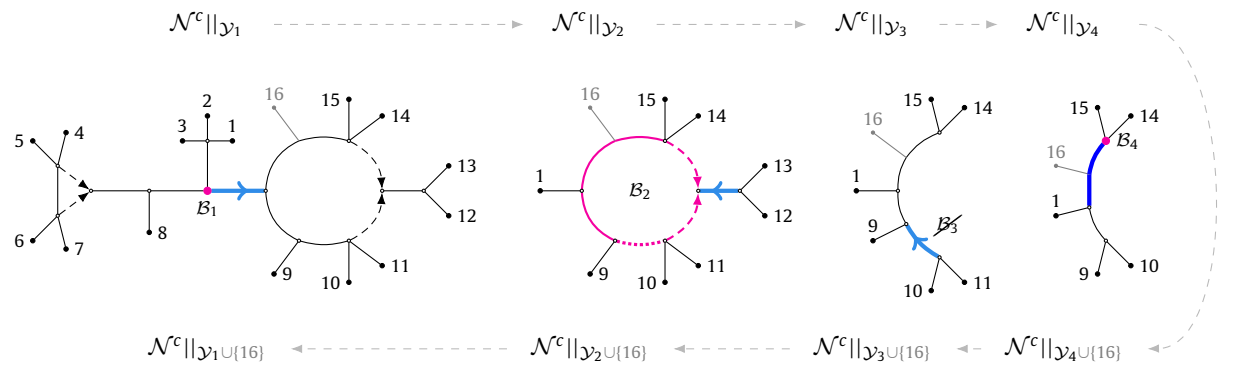

Fig. 11. Illustration of Algorithm 2 when used to attach leaf 16 to $\mathcal{N}^c||_{\mathcal{Y}_1}$, where $\mathcal{N}^c$ is the canonical network from Fig. 10 and $\mathcal{Y}_1 = \{1, \dots, 15\}$ contains its first fifteen leaves. The algorithm recurses (from left to right, excluding the grey leaf 16) on the leaf sets $\mathcal{Y}_1$, $\mathcal{Y}_2 = \{1, 9, \dots, 15\}$, $\mathcal{Y}_3 = \{1, 9, 10, 11, 14, 15\}$ and $\mathcal{Y}_4 = \{1, 9, 10, 14, 15\}$. In the recursive steps $i \in \{1, 2, 4\}$, a central blob $\mathcal{B}_i$ (in pink) and an incident nice central edge $u_i v_i$ (the thick light/dark blue edge) are found. In recursive step 3, the central blob $\mathcal{B}_3$ is undefined and the thick light blue edge $u_3 v_3$ instead originates from a nice spine edge (in dotted pink) of the previous central cycle $\mathcal{B}_2$. In the recursive steps $i \in \{1, 2, 3\}$, the light blue edge $u_i v_i$ is a pointing edge for leaf 16 (with the arrowhead showing the orientation) which is used to cut off part of the network. In the last recursive step, the dark blue edge $u_4 v_4$ is a strong stem edge used to attach leaf 16 (in grey). The algorithm subsequently moves back up to the top level of the recursion, each time gluing the relevant parts back to the canonical subnetworks of the network (from right to left, now including the grey leaf 16).

That means that a nice spine edge is 'balanced' with respect to the leaves in $\mathcal{X}$ (and not necessarily to those in $\mathcal{Y} \subset \mathcal{X}$). Lemma 12 shows that we can efficiently find the different types of nice edges.

Lemma 12. *Given a canonical level-1 network $\mathcal{N}^c$ on $\mathcal{X}$ with $n \geq 5$ leaves, one can find a central blob $\mathcal{B}$ and a nice central edge incident to $\mathcal{B}$ in $\mathcal{O}(n)$ time. Furthermore, if $\mathcal{B}$ is a central cycle, one can find a nice spine edge of $\mathcal{B}$ in $\mathcal{O}(n)$ time.*

Proof. The following algorithm shows the existence of a central blob and a nice central edge and how to find them. First, create the tree-of-blobs $\mathcal{T}(\mathcal{N}^c)$ in $\mathcal{O}(n)$ time. An internal vertex v of $\mathcal{T}(\mathcal{N}^c)$ is *leaf-centroid* if every component of $\mathcal{T}(\mathcal{N}^c) - v$ contains at most $\frac{n}{2}$ leaves. A straightforward greedy-type algorithm from [40, Lem. 5] shows that a leaf-centroid always exists and how to find one in linear time. Clearly, an internal vertex of $\mathcal{T}(\mathcal{N}^c)$ is a leaf-centroid if and only if the blob it represents is a central blob. Thus, we have found a central blob $\mathcal{B}$. If $\mathcal{B}$ is a central cycle with reticulation r, we can pick the cut-edge incident to r as our nice central edge. Otherwise, if $\mathcal{B}$ is a central vertex v, we pick a cut-edge vw (inducing the split $A|B$) that cuts off most of the leaves from v. That is, $|B|$ is as large as possible. Since v has at most four incident cut-edges, we have $|A| \leq \frac{3}{4}n$. Furthermore, since $n \geq 5$ and vw cuts off most of the leaves, we must also have $|B| \geq 2$. Thus, vw is a nice central edge.

Now assume that $\mathcal{B}$ is a central cycle and let $Y_1|Y_2|\dots|Y_{k-1}|Y_k|Z$ be the partition of $\mathcal{X}$ induced by $\mathcal{B}$. Assume that Z is the set of leaves below the reticulation, and that the Y_i follow the order of the cycle. Note that Y_1, Y_2, Y_{k-1} and Y_k correspond to the outer spine edges of the cycle and that the order of Y_1 and Y_2 (resp. Y_{k-1} and Y_k) is arbitrary. Then, $\mathcal{Y} = \bigcup_{i=1}^{k} Y_i$ is the set of leaves not below the reticulation. Note that for every $\ell \in \{1, \dots, k-1\}$ there is a cut-edge in $\mathcal{N}^c||_{\mathcal{Y}}$ inducing the split $Y_1 \cup \dots \cup Y_\ell | Y_{\ell+1} \cup \dots \cup Y_k$. In particular, these are all edges corresponding to spine edges of $\mathcal{B}$ (see Fig. 10).

Finding a nice spine edge of $\mathcal{B}$ is now fairly simple. Since $\mathcal{B}$ was a central cycle, we know that $1 \leq |Y_i| \leq \frac{n}{2}$ for each i. Our first candidate is the cut-edge that cuts off Y_1. If $|Y_1| \geq \frac{n}{4}$, that will be our nice spine edge. Otherwise, we move to the cut-edge that cuts off Y_1 and Y_2. Again, if $|Y_1 \cup Y_2| \geq \frac{n}{4}$ we stop, and if $|Y_1 \cup Y_2| < \frac{n}{4}$ we move to the next cut-edge. We keep continuing this procedure until $|Y_1 \cup \dots \cup Y_\ell| \geq \frac{n}{4}$ for some $\ell < k$. Then, the cut-edge e inducing the split $A'|B'$ (where $A' = Y_1 \cup \dots \cup Y_\ell$ and $B' = Y_{\ell+1} \cup \dots \cup Y_k$) has $|A'|, |B'| \leq \frac{3}{4}n$. Furthermore, since $n \geq 5$, this also forces $|A'|, |B'| \geq 2$. Thus, e is a nice spine edge of $\mathcal{B}$. □

Finally, we are ready to devise Algorithm 2 which recursively adds a leaf to a canonical network. In the spirit of Algorithm 1, our final reconstruction algorithm will then repeatedly apply Algorithm 2 to attach n leaves (in any arbitrary order) to a canonical network in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n \log n)$ quartet-splits. Before proving this in Theorem 13, we first roughly sketch how Algorithm 2 works. We refer to Fig. 11 for an illustration of the recursive part of the algorithm. Apart from the quartet-splits, the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ and the new leaf x, the algorithm takes as optional input a cut-edge uv. At the top level of the recursion this cut-edge is not provided, so we will leave this variation to the end of our discussion.

Unless we are in the base case, we first find a central blob $\mathcal{B}$ and incident central edge uv, after which we check whether it is a strong stem edge, weak stem edge or pointing edge. If uv turns out to be a strong stem edge, we can attach x to it (see Lemma 7 and Fig. 7). In the case that uv is a weak stem edge, we first find the weak stem path, after which we attach x to it (again see Lemma 7 and Fig. 7).

The final case is when uv (with induced split $A|B$) is a pointing edge. Then, it is enough to construct $\mathcal{N}^c||_{B\cup\{x,a\}}$, which can subsequently be glued to $\mathcal{N}^c||_{A\cup\{b\}}$ to create $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ on line 23 (see also the middle part of Fig. 9). On the one hand, if uv is oriented away from the blob $\mathcal{B}$ (i.e. u is in $\mathcal{B}$) or $\mathcal{B}$ is a central vertex, we simply attach x to $\mathcal{N}^c||_{B\cup\{a\}}$ by recursion. Since uv is a nice central edge this will always cut off some leaves. In the other case, $\mathcal{B}$ is a central cycle and uv is oriented towards $\mathcal{B}$ with v the reticulation of $\mathcal{B}$. We cannot recurse directly on $\mathcal{N}^c||_{B\cup\{a\}}$ since it might not cut off anything (consider again a network with one large cycle). So, we recurse on $\mathcal{N}^c||_B$ and attach x. Here, we also provide a cut-edge st which will take over the role of uv in the next level of recursion. Note that we do not provide an incident blob, so in the next level of recursion $\mathcal{B}$ is undefined. After $\mathcal{N}^c||_{B\cup\{x\}}$ is created, we attach the leaf a again (see Lemma 11 and the right part of Fig. 9).

Algorithm 2: Recursive procedure to attach a leaf to a canonical network.

Input: quarnet-splits of a semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X}$; canonical network $\mathcal{N}^c||_{\mathcal{Y}}$ on $\mathcal{Y}\subset\mathcal{X}$; leaf $x\in\mathcal{X}\setminus\mathcal{Y}$ to be attached; *optional:* a non-trivial cut-edge uv corresponding to a nice spine edge of the last recursive call
Output: canonical network $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$

1 **if** $|\mathcal{Y}|\in\{2,3,4\}$ **then**
2 find the canonical stem of $\mathcal{N}^c||_{\mathcal{Y}}$ by determining for each edge of $\mathcal{N}^c||_{\mathcal{Y}}$ whether it is a strong stem edge, weak stem edge or pointing edge, using the quarnet-splits of $\mathcal{N}$ and Lemma 8
3 $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ is constructed from $\mathcal{N}^c||_{\mathcal{Y}}$ as described in Lemma 7
4 **return** $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$
5 **if** *no cut-edge uv is provided* **then**
6 $\mathcal{B}\leftarrow$ central blob of $\mathcal{N}^c||_{\mathcal{Y}}$, using Lemma 12
7 $uv\leftarrow$ nice central edge of $\mathcal{N}^c||_{\mathcal{Y}}$ incident to $\mathcal{B}$, using Lemma 12
8 determine whether uv is a strong stem edge, weak stem edge or pointing edge, using the quarnet-splits of $\mathcal{N}$ and Lemma 8
9 **if** *uv is a strong stem edge* **then**
10 $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ is constructed from $\mathcal{N}^c||_{\mathcal{Y}}$ by attaching x to uv ; // see Figs. 7 and 11
11 **else if** *uv is a weak stem edge* **then**
12 determine the weak stem path in $\mathcal{N}^c||_{\mathcal{Y}}$ containing uv, using the quarnet-splits of $\mathcal{N}$ and Lemma 9
13 $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ is constructed from $\mathcal{N}^c||_{\mathcal{Y}}$ by attaching x to the end vertices of the cut-path ; // see Fig. 7
14 **else if** *uv is a pointing edge with orientation uv* **then** // otherwise, reverse roles of u, v
15 $A|B\leftarrow$ split induced by uv ; // this implies that A is on the side of u
16 $a\leftarrow$ arbitrary leaf from A; $b\leftarrow$ arbitrary leaf from B
17 **if** *$\mathcal{B}$ is undefined* **or** *u is in $\mathcal{B}$* **or** *$\mathcal{B}$ is a central vertex* **then**
18 $\mathcal{N}^c||_{B\cup\{x,a\}}$ is constructed from $\mathcal{N}^c||_{B\cup\{a\}}$ by attaching x with (recursion on) Algorithm 2
19 **else**
20 $st\leftarrow$ non-trivial cut-edge of $\mathcal{N}^c||_B$ corresponding to a nice spine edge of $\mathcal{B}$ in $\mathcal{N}^c||_{\mathcal{Y}}$, using Lemma 12
21 $\mathcal{N}^c||_{B\cup\{x\}}$ is constructed from $\mathcal{N}^c||_B$ by attaching x with (recursion on) Algorithm 2 and providing st as optional argument
22 $\mathcal{N}^c||_{B\cup\{x,a\}}$ is constructed from $\mathcal{N}^c||_{B\cup\{x\}}$ by attaching a, using the quarnet-splits of $\mathcal{N}$ and Lemma 11
23 $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ is constructed by glueing $\mathcal{N}^c||_{A\cup\{b\}}$ at b to $\mathcal{N}^c||_{B\cup\{a,x\}}$ at a ; // see Fig. 9
24 **return** $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$

Theorem 13. *Given the quarnet-splits of a semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X}=\{x_1,\dots,x_n\}$, there exists an algorithm that constructs the canonical form of $\mathcal{N}$ in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n\log n)$ quarnet-splits of $\mathcal{N}$.*

Proof. *Correctness:* The algorithm starts with a single edge between two arbitrary leaves and then repeatedly attaches the remaining $\mathcal{O}(n)$ leaves in arbitrary order with Algorithm 2. Note that we do not provide the optional cut-edge in the top level recursive calls of Algorithm 2. To prove correctness it will be enough to prove the following claim by strong induction on k: given a set $\mathcal{Y}\subset\mathcal{X}$ of k leaves, the canonical network $\mathcal{N}^c||_{\mathcal{Y}}$, some leaf $x\in\mathcal{X}\setminus\mathcal{Y}$, and optionally a non-trivial cut-edge uv, Algorithm 2 constructs the canonical network $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$. In the base cases where $k\in\{2,3,4\}$, the claim follows directly from Lemmas 7 and 8. This is true since the canonical stem can never be a stem cycle in a canonical network with at most four leaves, so Lemma 7 covers all cases.

Now let $k\geq 5$ be arbitrary and assume that the claim holds for all values k' with $2\leq k'<k$. If no cut-edge uv is provided, Lemma 12 will always find a central blob $\mathcal{B}$ and incident nice central edge uv. Lemma 8 will correctly determine whether uv is a strong stem edge, weak stem edge or pointing edge. If uv is a strong or weak stem edge, Lemmas 7 and 9 prove correctness.

The if-statement on line 14 is entered when uv is a pointing edge with orientation uv. The glueing operation to obtain $\mathcal{N}^c||_{\mathcal{Y}\cup\{x\}}$ on line 23 is then correct by Lemma 10. It remains to show that the creation of $\mathcal{N}^c||_{B\cup\{x,a\}}$ is correct. First, consider the case where $\mathcal{B}$ is undefined, u is in $\mathcal{B}$ and/or $\mathcal{B}$ is a central vertex. If $\mathcal{B}$ is undefined, a non-trivial cut-edge was provided and we have $2\leq|B\cup\{a\}|\leq k-1<k$. Otherwise, if u is in $\mathcal{B}$ and/or $\mathcal{B}$ is a central vertex, uv is a nice central edge. Specifically, if u is in $\mathcal{B}$ (so uv points away from $\mathcal{B}$), we have by the definition of a central edge that $|B\cup\{a\}|\leq\frac{1}{2}k+1<k$. On the other hand, if v is in $\mathcal{B}$ and $\mathcal{B}$ is a central vertex, then $|B\cup\{a\}|\leq\frac{3}{4}k+1<k$ by the definition of a nice central

edge. In all cases we recurse on a canonical network with strictly fewer leaves (and more than two leaves), which shows correctness by the induction hypothesis.

For the other case, note that since $\mathcal{B}$ is not undefined, no cut-edge uv could have been provided as an argument. Thus, because $\mathcal{B}$ is also not a central vertex, it is a central cycle and uv is a nice central edge incident to it such that v is the reticulation of $\mathcal{B}$. By Lemma 12, we can indeed find a nice spine edge of the cycle $\mathcal{B}$, which in turn corresponds to a non-trivial cut-edge st in $\mathcal{N}^c||_B$ (see e.g. Fig. 10). Thus, $2 \leq |B| < k$ and we recurse on a strictly smaller network (with at least two leaves), proving that $\mathcal{N}^c||_{B\cup\{x\}}$ is constructed correctly by the induction hypothesis. Since $\mathcal{B}$ is a central cycle and v is the reticulation of $\mathcal{B}$, Lemma 11 then proves that $\mathcal{N}^c||_{B\cup\{x,a\}}$ is constructed correctly.

Complexity: Recall that the main algorithm repeatedly invokes Algorithm 2 (without providing a cut-edge at the top level of recursion) to attach all the $\mathcal{O}(n)$ leaves. Thus, for the theorem it will be enough to show that Algorithm 2 takes $\mathcal{O}(k)$ time and uses $\mathcal{O}(\log k)$ quarnet-splits on a canonical network of k leaves (when no cut-edge is provided at the top level of recursion).

We will first consider the maximum recursion depth $D(k)$ of Algorithm 2 applied to a canonical network of k leaves. At the top level of recursion no cut-edge is provided and $\mathcal{B}$ is defined. Thus, when the algorithm recurses and u is in $\mathcal{B}$ or $\mathcal{B}$ is a central vertex, it recurses on a canonical network with at most $\frac{3}{4}k+1$ leaves (see the correctness proof). The other case when the algorithm recurses is if v is in $\mathcal{B}$ and $\mathcal{B}$ is a central cycle. Then, again by the previous correctness proof, it recurses on at most $k-1$ leaves. However, a cut-edge derived from a nice spine edge with spine split $A'|B'$ is also provided for the next level of recursion. We then know that $2 \leq |A'| \leq \frac{3}{4}k$ and $2 \leq |B'| \leq \frac{3}{4}k$. But this means that the recursion on the next level is on a leaf set with at most $\frac{3}{4}k+1$ leaves. Hence, we obtain the recurrence formula $D(k) \leq D\left(\frac{3}{4}k+1\right)+2$ for $k \geq 5$ with $D(5) \leq 1$. Whenever $k \geq 5$ we have $\frac{3}{4}k+1 < \frac{19}{20}k$ and therefore the formula simplifies to $D(k) \leq D\left(\frac{19}{20}k\right)+2$. It then easily follows that the recursion depth is $\mathcal{O}(\log k)$.

For the final complexity, note that the glueing operations, attaching leaves, finding splits and (implicitly) inducing canonical subnetworks all take no more than $\mathcal{O}(k)$ time per recursive level. This holds because the number of vertices and edges in a canonical network is a linear function of its leaves. All the other non-recursive operations also take $\mathcal{O}(k)$ time as outlined in Lemmas 8, 9, 11 and 12. Together with the recursion depth of $\mathcal{O}(\log k)$ this results in a total time complexity of $\mathcal{O}(k)$ for Algorithm 2. The operations from Lemmas 8, 11 and 12 all use $\mathcal{O}(1)$ quarnet-splits, whereas Lemma 9 uses $\mathcal{O}(\log k)$ quarnet-splits. However, the method from Lemma 9 is only used when the algorithm does not recurse any further. Hence, it follows from the recursion depth that the total number of quarnet-splits used by Algorithm 2 is $\mathcal{O}(\log k)$. □

5.3. Reconstructing a level-1 network from its canonical form

The previous subsection was concluded with an algorithm that constructs the canonical form of a semi-directed level-1 network from its quarnet-splits. If more information is available, i.e. the full quarnets are known, we can create the complete network from its canonical form. This is described in the following theorem. Assuming that the network is triangle-free even allows us to construct the complete network while disregarding any triangles in the quarnets. As explained in the introductory section, this aligns nicely with recent work showing that the triangles in quarnets are hard to locate in practice [21,28,29]. Furthermore, we also describe a variant of our algorithm that constructs most of the semi-directed level-1 network from its displayed quartets instead, aligning with the identifiability results from [6].

Theorem 14. *Let $\mathcal{N}$ be a semi-directed level-1 network on $\mathcal{X} = \{x_1, \ldots, x_n\}$ with $n \geq 4$.*

(a) Given the quarnets of $\mathcal{N}$, there exists an algorithm that reconstructs $\mathcal{N}$ in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n \log n)$ quarnets of $\mathcal{N}$.

(b) Given the quarnet-splits and the four-cycle quarnets of $\mathcal{N}$, there exists an algorithm that reconstructs $\mathcal{N}$ up to placing the reticulations in its triangles and up to collapsing its triangles in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n \log n)$ quarnet-splits and $\mathcal{O}(n)$ four-cycle quarnets of $\mathcal{N}$.

(c) Given the displayed quartets of $\mathcal{N}$, there exists an algorithm that reconstructs $\mathcal{N}$ up to placing the reticulations in its triangles and 4-cycles and up to collapsing its triangles in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n \log n)$ displayed quartets of $\mathcal{N}$.

Proof. We first prove part (a) of the theorem and treat parts (b) and (c) separately at the end of the proof. Theorem 13 proves that we can construct the canonical network $\mathcal{N}^c$ in quadratic time from $\mathcal{O}(n \log n)$ quarnet-splits of $\mathcal{N}$, which can be formed from the quarnets of $\mathcal{N}$. It remains to show that we can transform every blob of $\mathcal{N}^c$ into the correct blob of $\mathcal{N}$, using the following three steps. The complexity bounds will then follow directly from the fact that $\mathcal{N}$ has $\mathcal{O}(n)$ blobs and that the described methods take no more than $\mathcal{O}(n)$ time (and $\mathcal{O}(1)$ quarnets) per blob. Given a cycle $\mathcal{C}$, we will refer to the circular order of the subnetworks around $\mathcal{C}$ and the placement of the reticulation as the *orientation* of $\mathcal{C}$.

Step 1. Every blob of $\mathcal{N}^c$ that consists of a single degree-4 vertex v is a 4-cycle $\mathcal{C}$ in $\mathcal{N}$. Suppose the partition of $\mathcal{X}$ induced by v is $Y_1|Y_2|Y_3|Y_4$ and choose one leaf y_i from every Y_i. Then the quarnet $\mathcal{N}|_{\{y_1,y_2,y_3,y_4\}}$ will be a four-cycle with one reticulation vertex and its orientation tells us exactly what the orientation of $\mathcal{C}$ must be in $\mathcal{N}$.

Step 2. Similarly, every blob of $\mathcal{N}^c$ that consists of a single degree-3 vertex v is either a single vertex v' or a triangle $\mathcal{C}$ in $\mathcal{N}$. Suppose the partition of $\mathcal{X}$ induced by v is $Y_1|Y_2|Y_3$ such that $|Y_3| \geq 2$ (this exists since $n \geq 4$). Choose one leaf y_i from every Y_i and choose y_4 as a different leaf from Y_3. The quarnet $\mathcal{N}|_{\{y_1,y_2,y_3,y_4\}}$ will then be a quartet tree, a single

triangle, or a double triangle (see e.g. Fig. 2). If we take the subnetwork of this quarnet induced by $\{y_1, y_2, y_3\}$, we either get a triangle or a 3-star. If this *trinet* is a triangle, we know the exact orientation of the triangle $\mathcal{C}$ in $\mathcal{N}$. If the trinet is a 3-star, we know that v is a single vertex v' in $\mathcal{N}$.

Step 3. Lastly, in every large cycle $\mathcal{C}$ of $\mathcal{N}$ the two cycle contraction edges are contracted when obtaining $\mathcal{N}^c$. A similar strategy as above allows us to determine how to undo this operation. Specifically, suppose that $Y_1|Y_2|\dots|Y_{k-1}|Y_k|Z$ is the partition of $\mathcal{X}$ induced by $\mathcal{C}$ with Z below the reticulation, and Y_1, Y_2 and Y_{k-1}, Y_k the leaf sets corresponding to the two cycle contraction edges. If we let $y_i \in Y_i$ and $z \in Z$ be arbitrary, then the two four-cycle quarnets $\mathcal{N}|_{\{z,y_1,y_2,y_{k-1}\}}$ and $\mathcal{N}|_{\{z,y_{k-1},y_k,y_1\}}$ directly show what the orientation of $\mathcal{C}$ must be in $\mathcal{N}$.

For part (b) of the theorem it is enough to note that Step 2 is redundant if we want to reconstruct $\mathcal{N}$ up to collapsing its triangles and up to placing reticulations in them. For part (c), we use that from the displayed quartets of a semi-directed network we can obtain the quarnet-splits and the circular ordering of the four-cycle quarnets without the placement of the reticulation (see [34, Lem. 5.1]). To see the result, note that we can again skip Step 2, whereas we can perform most of Step 1 except for placing the reticulation in the 4-cycles of the network. Finally, note that the number of displayed quartets used is at most three times the corresponding number of quarnets used. □

The proof of the previous result also implicitly shows why the canonical network is the most refined network one can unambiguously construct from quarnet-splits. In particular, it is impossible to locate triangles using quarnet-splits, to infer the circular order of subnetworks and the reticulation vertex of 4-cycles, and to determine how the two cut-edge pairs next to a reticulation vertex of a large cycle are ordered. Therefore, quarnet-splits do not uniquely determine a complete semi-directed level-1 network.

Recall that Proposition 6 showed that any algorithm using only quarnet-splits to reconstruct the tree-of-blobs of a semi-directed level-ℓ network needs to use $\Omega(n \log n + \ell \cdot n)$ quarnet-splits. Since one can easily create the tree-of-blobs of a level-1 network from its canonical form, Theorem 13 implies that one can reconstruct the tree-of-blobs of a level-1 network from $\mathcal{O}(n \log n)$ quarnet-splits. Thus, the lower bound from Proposition 6 is tight for level-1 networks. Furthermore, a similar information-theoretic argument as in Proposition 6 shows that the algorithm from Theorem 14 is optimal in terms of the number of quarnets it uses. That is, no algorithm exists that uses asymptotically less than $\mathcal{O}(n \log n)$ quarnets to reconstruct a semi-directed level-1 network. We present the formal statement in the following proposition. The argument can trivially be adapted to show that the number of displayed quartets used in part (c) of the previous theorem is also asymptotically optimal.

Proposition 15. *Given the quarnets of a semi-directed level-1 network $\mathcal{N}$ on $\mathcal{X} = \{x_1, \dots, x_n\}$, any algorithm using only quarnets to reconstruct $\mathcal{N}$ needs to use $\Omega(n \log n)$ quarnets.*

Proof. In [39] it is shown that there are $2^{\Omega(n \log n)}$ possible binary phylogenetic trees on n leaves, which is thus surely a lower bound on the number of n-leaf semi-directed level-1 networks. Up to labeling the leaves, there are only six possible level-1 quarnets (see Fig. 2). Counting the number of different labelings then reveals that on any given set of four leaves there are 3 possible quartet trees, 18 single triangles, 18 double triangles, and 12 four-cycles. Thus, by an information-theoretic argument, any algorithm using only quarnets to reconstruct $\mathcal{N}$ needs to use $\log_{3+18+18+12}(2^{\Omega(n \log n)}) = \Omega(n \log n)$ quarnets in the worst case. □

6. Discussion

The main contributions of this paper are two-fold. First, we presented an $\mathcal{O}(n^3)$ time algorithm that reconstructs the tree-of-blobs of any binary n-leaf semi-directed network with unbounded level, using $\mathcal{O}(n^3)$ splits of its quarnets (assuming direct access to the splits of all quarnets). We have not shown that this is optimal but have shown that any such algorithm needs $\Omega(n^2)$ quarnet-splits in the worst case. Secondly, we created an algorithm that reconstructs binary n-leaf semi-directed level-1 networks in $\mathcal{O}(n^2)$ time, using an optimal number of $\mathcal{O}(n \log n)$ quarnets (assuming direct access to all quarnets). A variant of this algorithm can also reconstruct most of the semi-directed level-1 network using its displayed quartets instead.

Two obvious open questions that remain are whether the $\Theta(n)$ gap for the tree-of-blobs reconstruction algorithm can be bridged and whether the time complexity of the level-1 network reconstruction algorithm can be reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. For the latter improvement, one might draw some inspiration from [31], who present an optimal algorithm that reconstructs undirected phylogenetic trees of degree d from $\Theta(d \cdot n \cdot \log_d n)$ quartets and running in similar time.[1] The optimality of that algorithm also signifies an inherent difference between the seemingly similar tree-of-blobs reconstruction from quarnet-splits and the phylogenetic tree reconstruction from quartets. Whereas the number of quartets used in the algorithm by [31] reduces to $\Theta(n \log n)$ for binary trees, we showed that reconstructing the tree-of-blobs of any binary

[1] This result follows from the equivalence between undirected tree reconstruction from quartets and directed tree reconstruction from triplets [30].

semi-directed network requires $\Omega(n^2)$ quartet-splits. One might suspect this difference follows from the fact that the trees-of-blobs themselves could be of high degree. This is not true however, since we showed that trees-of-blobs of binary semi-directed level-1 networks can be reconstructed with $\mathcal{O}(n \log n)$ quartet-splits, even if they have high-degree blobs.

A further research direction related to this paper is the step from level-1 to level-2. In [41] this was achieved for the construction of rooted/directed level-2 phylogenetic networks from trinets (3-leaf subnetworks). Recent findings by Huber et al. [27] show that semi-directed level-2 networks can be distinguished by their quartets. This suggests the jump from level-1 to level-2 is possible for the semi-directed case. We already know how to reconstruct the tree-of-blobs of a level-2 network (although this could perhaps be sped up) which only leaves the separate blobs to be reconstructed. To this end, it might be worthwhile to generalize our canonical form to higher-level networks, thus formalizing the most refined level-2 network that can be reconstructed from quartet-splits. Note that [42] has already made some progress on identifiability results for the level-2 case under group-based models of evolution.

Lastly, from a practical point of view, a more robust algorithm that also works for imperfect data would be a logical next step. One objective for such a method might be to find a semi-directed level-1 network that induces as many given quartets as possible: the *maximum quartet compatibility* problem. However, this is NP-hard, which can easily be shown using the fact that it already is for trees and quartets [43]. Thus, there is a need for heuristics, ideally ones that are *consistent*: returning the true network whenever the data is perfect. The software tool Squirrel has taken a first step in this direction by allowing a *dense* set of quartets (i.e. exactly one quartet for each set of four leaves) that do not necessarily come from a single network [21]. In light of our work, a next step could be to allow a *non-dense* set of quartets (i.e. at most one quartet for each set of four leaves) as input, where the quartets are not necessarily all induced by the same network. Allowing fewer quartets in the input could lead to a significant speed-up. A sensible starting point is to investigate methods that construct rooted/directed level-1 phylogenetic networks from a non-dense set of subtrees on few leaves (e.g. Lev1athan [17]).

CRediT authorship contribution statement

Martin Frohn: Methodology, Conceptualization. **Niels Holtgrefe:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Conceptualization. **Leo van Iersel:** Writing – review & editing, Methodology, Funding acquisition, Conceptualization. **Mark Jones:** Methodology, Conceptualization. **Steven Kelk:** Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the reviewers for their helpful comments and suggestions to improve the paper.

Appendix A. Proof of Lemma 1

Lemma 1. *Given a semi-directed network $\mathcal{N}$ on $\mathcal{X}$, a non-trivial partition $A|B$ of $\mathcal{X}$ and any $a_1 \in A$, $b_1 \in B$, the following are equivalent:*

(i) $A|B$ is a split in $\mathcal{N}$;

(ii) $a_1a_2|b_1b_2$ is a quartet-split of $\mathcal{N}$ for all $a_2 \in A \setminus \{a_1\}$, $b_2 \in B \setminus \{b_1\}$.

Proof. As discussed in Section 3, the proof of this lemma is along the lines of the proof of a slightly weaker result by Huber et al. [27]. In particular, they show that $A|B$ is a split in $\mathcal{N}$ if and only if $a_1a_2|b_1b_2$ is a quartet-split of $\mathcal{N}$ for all $a_1, a_2 \in A$ and $b_1, b_2 \in B$. The first direction of our lemma follows directly from that result. As in [27], we prove the reverse direction by induction on the number of non-trivial splits in $\mathcal{N}$.

For the base case, assume that $\mathcal{N}$ is simple and let $A|B$ be any non-trivial partition of $\mathcal{X}$. The proof by Huber et al. [27] shows that in that case there exist $\tilde{a}_1 \in A$ and $\tilde{b}_1 \in B$ such that $\tilde{a}_1\tilde{a}_2|\tilde{b}_1\tilde{b}_2$ is not a quartet-split for any $\tilde{a}_2 \in A \setminus \{\tilde{a}_1\}$ and $\tilde{b}_2 \in B \setminus \{\tilde{b}_1\}$. Our base case follows directly from this statement.

It remains to do the induction step. First, let $A|B$ be any non-trivial partition of $\mathcal{X}$, fix some $a_1 \in A, b_1 \in B$ and assume that $a_1a_2|b_1b_2$ is a quartet-split for all $a_2 \in A \setminus \{a_1\}$ and $b_2 \in B \setminus \{b_2\}$ [Assumption 1]. Furthermore, suppose that $\mathcal{N}$ is not simple and that there exists a non-trivial split $C|D$. Towards a contradiction, assume that $A \not\subseteq C$, $A \not\subseteq D$, $B \not\subseteq C$, and $B \not\subseteq D$. Without loss of generality, suppose that $a_1 \in C$ and $b_1 \in D$. (The other cases are analogous.) Since $A \not\subseteq C$, there exists an $\tilde{a} \notin C$, so $\tilde{a} \in D$. Similarly, since $B \not\subseteq D$, there exists a $\tilde{b} \notin D$ such that $\tilde{b} \in C$. Note that we must have that $\tilde{a} \neq a_1$ and $\tilde{b} \neq b_1$. Thus, we have that $a_1, \tilde{b} \in C$ and $b_1, \tilde{a} \in D$. Since $C|D$ is a split in $\mathcal{N}$, the forward direction of the lemma says that then $a_1\tilde{b}|\tilde{a}b_1$ is a quartet-split of $\mathcal{N}$. This means that $a_1\tilde{a}|b_1\tilde{b}$ is not a quartet-split of $\mathcal{N}$: a contradiction with Assumption 1. All in all, we must have that $A \subseteq C$, $A \subseteq D$, $B \subseteq C$, or $B \subseteq D$.

Without loss of generality, assume that $B \subseteq D$, and consequently $C \subseteq A$. Let uv be the cut-edge in $\mathcal{N}$ inducing the split $C|D$. Then, its removal from $\mathcal{N}$ creates two connected components: $\mathcal{N}_C$ containing u and all leaves from C, and $\mathcal{N}_D$ containing v and all leaves from D. We now construct a new network $\mathcal{N}'$ from $\mathcal{N}$ by identifying all vertices of $\mathcal{N}$ that are in $\mathcal{N}_C$ as a single leaf c^*. Then, $\mathcal{N}'$ has one non-trivial split less than $\mathcal{N}$. Furthermore, if we let $A' = (A \setminus C) \cup \{c^*\}$ and $B' = B$, then $A'|B'$ is a partition of the leaves of $\mathcal{N}'$. Distinguishing between two cases, we now show that $A'|B'$ is a split in $\mathcal{N}'$. By the construction of $\mathcal{N}'$, this means that $A|B$ is a split in $\mathcal{N}$, proving the reverse direction.

Case 1: $a_1 \in C$. Let $\tilde{a}_2 \in A' \setminus \{c^*\}$ and $\tilde{b}_2 \in B' \setminus \{b_1\}$ be arbitrary. Then, $\tilde{a}_2 \in (A \setminus C) \setminus \{a_1\}$ and $\tilde{b}_2 \in B \setminus \{b_1\}$. Furthermore, $\tilde{a}_2 \neq a_1$ by definition of A'. Assumption 1 then ensures that $a_1\tilde{a}_2|b_1\tilde{b}_2$ is a quarnet-split of $\mathcal{N}$. Since $a_1 \in C$, we then have $c^*\tilde{a}_2|b_1\tilde{b}_2$ as a quarnet-split of $\mathcal{N}'$. Because $\tilde{a}_2 \in A' \setminus \{c^*\}$ and $\tilde{b}_2 \in B' \setminus \{b_1\}$ were arbitrary, the induction hypothesis (and the fact that $\mathcal{N}'$ has a non-trivial split less than $\mathcal{N}$) shows that $A'|B'$ is a split in $\mathcal{N}'$.

Case 2: $a_1 \notin C$. Let $\tilde{a}_2 \in A' \setminus \{a_1\}$ and $\tilde{b}_2 \in B' \setminus \{b_1\}$ be arbitrary. Clearly, $\tilde{b}_2 \in B \setminus \{b_1\}$. By Assumption 1, $a_1c|b_1\tilde{b}_2$ is a quarnet-split of $\mathcal{N}$ for all $c \in C \subseteq A \setminus \{a_1\}$. Consequently, if $\tilde{a}_2 = c^*$, then $a_1\tilde{a}_2|b_1\tilde{b}_2$ is a quarnet-split of $\mathcal{N}'$. Otherwise, if $\tilde{a}_2 \in (A \setminus C) \setminus \{a_1\}$, Assumption 1 implies that $a_1\tilde{a}_2|b_1\tilde{b}_2$ is a quarnet-split of $\mathcal{N}$ and thus of $\mathcal{N}'$. Because $\tilde{a}_2 \in A' \setminus \{a_1\}$ and $\tilde{b}_2 \in B' \setminus \{b_1\}$ were arbitrary, the induction hypothesis (and the fact that $\mathcal{N}'$ has a non-trivial split less than $\mathcal{N}$) shows that $A'|B'$ is a split in $\mathcal{N}'$. □

Data availability

No data was used for the research described in the article.

References

[1] E. Bapteste, L. van Iersel, A. Janke, S. Kelchner, S. Kelk, J.O. McInerney, D.A. Morrison, L. Nakhleh, M. Steel, L. Stougie, J. Whitfield, Networks: expanding evolutionary thinking, Trends Genet. 29 (8) (2013) 439–441, https://doi.org/10.1016/j.tig.2013.05.007.

[2] L. Wang, K. Zhang, L. Zhang, Perfect phylogenetic networks with recombination, in: Proceedings of the 2001 ACM Symposium on Applied Computing, 2001, pp. 46–50.

[3] D. Gusfield, S. Eddhu, C. Langley, Efficient reconstruction of phylogenetic networks with constrained recombination, in: Computational Systems Bioinformatics. CSB2003, in: Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003, IEEE, 2003, pp. 363–374.

[4] S. Kong, J.C. Pons, L. Kubatko, K. Wicke, Classes of explicit phylogenetic networks and their biological and mathematical significance, J. Math. Biol. 84 (6) (2022) 47, https://doi.org/10.1007/s00285-022-01746-y.

[5] C. Solís-Lemus, C. Ané, Inferring phylogenetic networks with maximum pseudolikelihood under incomplete lineage sorting, PLoS Genet. 12 (3) (2016) e1005896, https://doi.org/10.1371/journal.pgen.1005896.

[6] H. Baños, Identifying species network features from gene tree quartets under the coalescent model, Bull. Math. Biol. 81 (2019) 494–534, https://doi.org/10.1007/s11538-018-0485-4.

[7] C. Solís-Lemus, A. Coen, C. Ané, On the identifiability of phylogenetic networks under a pseudolikelihood model, arXiv:2010.01758, 2020.

[8] J. Xu, C. Ané, Identifiability of local and global features of phylogenetic networks from average distances, J. Math. Biol. 86 (1) (2023) 12, https://doi.org/10.1007/s00285-022-01847-8.

[9] E.S. Allman, H. Baños, M. Garrote-Lopez, J.A. Rhodes, Identifiability of level-1 species networks from gene tree quartets, Bull. Math. Biol. 81 (9) (2024), https://doi.org/10.1007/s11538-024-01339-4.

[10] E. Gross, C. Long, Distinguishing phylogenetic networks, SIAM J. Appl. Algebra Geom. 2 (1) (2018) 72–93, https://doi.org/10.1137/17M1134238.

[11] B. Hollering, S. Sullivant, Identifiability in phylogenetics using algebraic matroids, J. Symb. Comput. 104 (2021) 142–158, https://doi.org/10.1016/j.jsc.2020.04.012.

[12] E. Gross, L. van Iersel, R. Janssen, M. Jones, C. Long, Y. Murakami, Distinguishing level-1 phylogenetic networks on the basis of data generated by Markov processes, J. Math. Biol. 83 (2021) 1–24, https://doi.org/10.1007/s00285-021-01653-8.

[13] C. Than, D. Ruths, L. Nakhleh, Phylonet: a software package for analyzing and reconstructing reticulate evolutionary relationships, BMC Bioinform. 9 (2008) 1–16, https://doi.org/10.1186/1471-2105-9-322.

[14] Y. Yu, L. Nakhleh, A maximum pseudo-likelihood approach for phylogenetic networks, BMC Genomics 16 (2015) 1–10, https://doi.org/10.1186/1471-2164-16-s10-s10.

[15] C. Solís-Lemus, P. Bastide, C. Ané, Phylonetworks: a package for phylogenetic networks, Mol. Biol. Evol. 34 (12) (2017) 3292–3298, https://doi.org/10.1093/molbev/msx307.

[16] S. Kong, D.L. Swofford, L.S. Kubatko, Inference of phylogenetic networks from sequence data using composite likelihood, Syst. Biol. (2024) syae054, https://doi.org/10.1093/sysbio/syae054.

[17] K.T. Huber, L. van Iersel, S. Kelk, R. Suchecki, A practical algorithm for reconstructing level-1 phylogenetic networks, IEEE/ACM Trans. Comput. Biol. Bioinform. 8 (3) (2010) 635–649, https://doi.org/10.1109/TCBB.2010.17.

[18] J. Oldman, T. Wu, L. van Iersel, V. Moulton, TriLoNet: piecing together small networks to reconstruct reticulate evolutionary histories, Mol. Biol. Evol. 33 (8) (2016) 2151–2162, https://doi.org/10.1093/molbev/msw068.

[19] E.S. Allman, H. Baños, J.A. Rhodes, NANUQ: a method for inferring species networks from gene trees under the coalescent model, Algorithms Mol. Biol. 14 (2019) 1–25, https://doi.org/10.1186/s13015-019-0159-2.

[20] E.S. Allman, H. Baños, J.A. Rhodes, K. Wicke, NANUQ$^+$: a divide-and-conquer approach to network estimation, bioRxiv, https://doi.org/10.1101/2024.10.30.621146, 2024.

[21] N. Holtgrefe, K.T. Huber, L. van Iersel, M. Jones, S. Martin, V. Moulton, Squirrel: reconstructing semi-directed phylogenetic level-1 networks from four-leaved networks or sequence alignments, Mol. Biol. Evol. (2025), to appear.

[22] T. Warnow, Y. Tabatabaee, S.N. Evans, Advances in estimating level-1 phylogenetic networks from unrooted SNPs, J. Comput. Biol. 32 (1) (2024) 3–27, https://doi.org/10.1089/cmb.2024.0710.

[23] S. Huebler, R. Morris, J. Rusinko, Y. Tao, Constructing semi-directed level-1 phylogenetic networks from quarnets, arXiv:1910.00048, 2019.

[24] P. Gambette, V. Berry, C. Paul, Quartets and unrooted phylogenetic networks, J. Bioinform. Comput. Biol. 10 (04) (2012) 1250004, https://doi.org/10.1142/s0219720012500047.

[25] J. Keijsper, R. Pendavingh, Reconstructing a phylogenetic level-1 network from quartets, Bull. Math. Biol. 76 (2014) 2517–2541, https://doi.org/10.1007/s11538-014-0022-z.

[26] K.T. Huber, V. Moulton, C. Semple, T. Wu, Quarnet inference rules for level-1 networks, Bull. Math. Biol. 80 (2018) 2137–2153, https://doi.org/10.1007/s11538-018-0450-2.
[27] K.T. Huber, L. van Iersel, M. Jones, V. Moulton, L. Veenema-Nipius, When are quarnets sufficient to reconstruct semi-directed phylogenetic networks?, arXiv:2408.12997, 2024.
[28] T. Barton, E. Gross, C. Long, J. Rusinko, Statistical learning with phylogenetic network invariants, arXiv:2211.11919, 2022.
[29] S. Martin, V. Moulton, R.M. Leggett, Algebraic invariants for inferring 4-leaf semi-directed phylogenetic networks, bioRxiv, https://doi.org/10.1101/2023.09.11.557152, 2023.
[30] A. Lingas, H. Olsson, A. Östlin, Efficient merging, construction, and maintenance of evolutionary trees, in: Automata, Languages and Programming, Springer, Berlin, Heidelberg, 1999, pp. 544–553.
[31] G.S. Brodal, R. Fagerberg, C.N.S. Pedersen, A. Östlin, The complexity of constructing evolutionary trees using experiments, in: Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28, Springer, 2001, pp. 140–151.
[32] D. Gusfield, V. Bansal, V. Bafna, Y.S. Song, A decomposition theory for phylogenetic networks and incompatible characters, J. Comput. Biol. 14 (10) (2007) 1247–1272, https://doi.org/10.1089/cmb.2006.0137.
[33] E.S. Allman, H. Baños, J.D. Mitchell, J.A. Rhodes, The tree of blobs of a species network: identifiability under the coalescent, J. Math. Biol. 86 (1) (2023) 10, https://doi.org/10.1007/s00285-022-01838-9.
[34] J.A. Rhodes, H. Baños, J. Xu, C. Ané, Identifying circular orders for blobs in phylogenetic networks, Adv. Appl. Math. 163 (2025) 102804, https://doi.org/10.1016/j.aam.2024.102804.
[35] E.S. Allman, H. Baños, J.D. Mitchell, J.A. Rhodes, TINNiK: inference of the tree of blobs of a species network under the coalescent model, Algorithms Mol. Biol. 19 (1) (2024) 23, https://doi.org/10.1101/2024.04.20.590418.
[36] V. Berry, O. Gascuel, Inferring evolutionary trees with strong combinatorial evidence, Theor. Comput. Sci. 240 (2) (2000) 271–298, https://doi.org/10.1016/S0304-3975(99)00235-2.
[37] J. Hopcroft, R. Tarjan, Algorithm 447: efficient algorithms for graph manipulation, Commun. ACM 16 (6) (1973) 372–378, https://doi.org/10.1145/362248.362272.
[38] P. Buneman, The recovery of trees from measures of dissimilarity, in: Mathematics in the Archaeological and Historical Sciences, Edinburgh University Press, Edinburgh, 1971, pp. 387–395.
[39] S.K. Kannan, E.L. Lawler, T.J. Warnow, Determining the evolutionary tree using experiments, J. Algorithms 21 (1) (1996) 26–50, https://doi.org/10.1006/jagm.1996.0035.
[40] H. Bodlaender, J. Gilbert, H. Hafsteinsson, T. Kloks, Approximating treewidth, pathwidth, frontsize, and shortest elimination tree, J. Algorithms 18 (1995) 238–255, https://doi.org/10.1006/jagm.1995.1009.
[41] L. van Iersel, S. Kole, V. Moulton, L. Nipius, An algorithm for reconstructing level-2 phylogenetic networks from trinets, Inf. Process. Lett. 178 (2022) 106300, https://doi.org/10.1016/j.ipl.2022.106300.
[42] M. Ardiyansyah, Distinguishing level-2 phylogenetic networks using phylogenetic invariants, arXiv:2104.12479, 2021.
[43] D. Bryant, M. Steel, Constructing optimal trees from quartets, J. Algorithms 38 (1) (2001) 237–259, https://doi.org/10.1006/jagm.2000.1133.