Algorithmica (2014) 68:886–915
DOI 10.1007/s00453-012-9708-5

Constructing Minimal Phylogenetic Networks from Softwired Clusters is Fixed Parameter Tractable

Steven Kelk · Celine Scornavacca

Received: 18 August 2011 / Accepted: 24 October 2012 / Published online: 8 November 2012

Abstract Here we show that, given a set of clusters $\mathcal{C}$ on a set of taxa $\mathcal{X}$, where $|\mathcal{X}| = n$, it is possible to determine in time $f(k) \cdot poly(n)$ whether there exists a level-$\leq k$ network (i.e. a network where each biconnected component has reticulation number at most k) that represents all the clusters in $\mathcal{C}$ in the softwired sense, and if so to construct such a network. This extends a result from Kelk et al. (in IEEE/ACM Trans. Comput. Biol. Bioinform. 9:517–534, 2012) which showed that the problem is polynomial-time solvable for fixed k. By defining "k-reticulation generators" analogous to "level-k generators", we then extend this fixed parameter tractability result to the problem where k refers not to the level but to the reticulation number of the whole network.

Keywords Phylogenetics · Fixed parameter tractability · Directed acyclic graphs

1 Introduction

1.1 Phylogenetic Networks and Softwired Clusters

The traditional model for representing the evolution of a set of species $\mathcal{X}$ (or, more abstractly, a set of *taxa*) is the *rooted phylogenetic tree* [9, 10, 25]. Essentially, this is a singly-rooted tree where the leaves are bijectively labelled by $\mathcal{X}$ and the edges are

S. Kelk
Department of Knowledge Engineering (DKE), Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands
e-mail: steven.kelk@maastrichtuniversity.nl

C. Scornavacca (✉)
ISEM, UMR 5554, CNRS, Univ. Montpellier 2, Place Eugène Bataillon, Montpellier, France
e-mail: celine.scornavacca@univ-montp2.fr

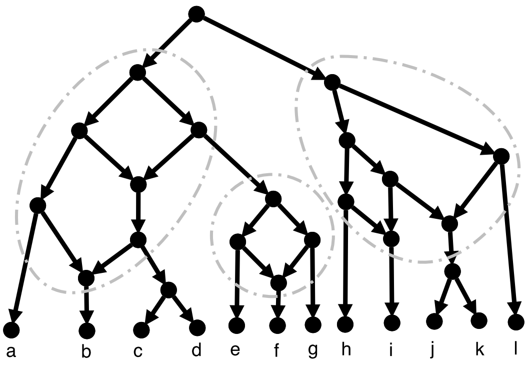

Fig. 1 Example of a phylogenetic network with five reticulations. The encircled subgraphs form its biconnected components, also known as its "tangles". This binary network has level equal to 2 since each biconnected component contains at most two reticulations

directed away from the root. In recent years there has been a growing interest in extending this model to also incorporate non-treelike evolutionary phenomena such as hybridizations, recombinations and horizontal gene transfers. This has subsequently stimulated research into *rooted phylogenetic networks* which generalize rooted phylogenetic trees by also permitting nodes with indegree two or higher, known as *reticulation* nodes, or simply *reticulations*. For detailed background information on phylogenetic networks we refer the reader to [14–16, 22, 24, 28]. Figure 1 shows an example of a rooted phylogenetic network.

We are interested in the following biologically-motivated optimization problem. We are given a set $\mathcal{C}$ of *clusters* on $\mathcal{X}$, where a cluster is simply a strict subset of $\mathcal{X}$. We wish to construct a phylogenetic network that "represents" all the clusters in $\mathcal{C}$ such that the amount of reticulation in the network is "minimized". There are several different definitions of "represents" and "minimized" present in the literature. In this article we will consider only the *softwired* definition of "represents" [14–16, 31]. Most of our formal definitions will be deferred to the preliminaries. Nevertheless, it is helpful to already formally state that a rooted phylogenetic tree T on $\mathcal{X}$ represents a cluster $C \subset \mathcal{X}$ if T contains an edge (u, v) such that C is exactly equal to the subset of $\mathcal{X}$ reachable from v by directed paths. A phylogenetic network N on $\mathcal{X}$, on the other hand, represents a cluster $C \subset \mathcal{X}$ in the softwired sense if there exists *some* rooted phylogenetic tree T on $\mathcal{X}$ such that T represents C and T is topologically embedded inside N. Regarding "minimized", we consider two closely related, but subtly different, variants of minimality. The first variant, *reticulation number minimization*, aims at minimizing the *total* number of reticulation nodes in the network.[1] The second, less well-known variant, *level minimization* [18, 19, 26, 29, 30], asks us to minimize the maximum number of reticulation nodes contained in any "tangled" region of the network, which correspond to the non-trivial biconnected components of the underlying undirected graph (see Fig. 1). The reticulation number is a global optimality criterion, while the level is a local optimality criterion. In general minimizing for one variant does not induce minimum solutions for the other variant (see e.g. [15, Fig. 3]), although the algorithmic techniques used to tackle these problems are often related

[1]This is the definition when all reticulation vertices have indegree-2, for more general networks reticulation number is defined slightly differently. See the Preliminaries for more information.

[20]. Both these problems are NP-hard and APX-hard [2, 28]. This raises the natural question: given a set of clusters $\mathcal{C}$ and a fixed integer r, is it NP-hard to determine whether or not there exists a network with reticulation number (respectively, level) equal to r representing $\mathcal{C}$?

Prior to this article there were only partial answers known to these questions. In [20] it was proven that the level-oriented question can be answered in polynomial time if the level is fixed. A striking aspect of this proof is that the running time of the algorithm is only polynomial time in a highly theoretical sense: it is too high to be of any practical interest. This exorbitant running time has two causes. Firstly, the exhaustive enumeration of all *level-k generators* [29], essentially the set of all possible underlying topologies of a network constituted of a biconnected component if the taxa are ignored. Secondly, after determining the correct generator, a second wave of exhaustive enumeration determines where a critical subset of $\mathcal{X}$ should be located within the network, after which all remaining elements of $\mathcal{X}$ can easily be added without much computational effort.

The question of whether a corresponding positive result would hold for reticulation number minimization was left open, although the emergence of several partial results and practically efficient algorithms [15, 20] suggested that this might well be the case. Furthermore, it was not obvious how the algorithm from [20] could be adapted to yield a fixed parameter tractable algorithm for level minimization—where the parameter is the level of the network k-since k appears as an exponent of $|\mathcal{X}|$ in the running time of the algorithm. (We refer to [6, 7, 11, 23] for an introduction to fixed parameter tractability). Curiously, the main problem is not the enumeration of the generators, because the number of generators is independent of $|\mathcal{X}|$ [8], but the allocation of the critical initial subset of taxa to their correct location in the network.

In this article we settle all these questions by proving for the first time that both level minimization and reticulation number minimization are fixed parameter tractable (where, in the case of reticulation number minimization, the parameter is the reticulation number of the whole network). We give one algorithm for level minimization and one algorithm for reticulation minimization, although the two algorithms have a large common core. The algorithms again rely heavily on generators, which we extend here to also be useful in the context of reticulation number minimization; generators had hitherto only appeared in the level minimization literature. In both algorithms the major non-triviality is showing how the network structure can still be adequately recovered if the parameter is no longer allowed to appear in the exponent of $|\mathcal{X}|$ as it was in [20].

1.2 Beyond Softwired Clusters: The Wider Context

We believe that this approach is significant beyond the softwired cluster literature. Other articles discuss the problem of constructing rooted phylogenetic networks not by combining clusters but by combining triplets [27, 30], characters [12, 13, 21, 35] or entire phylogenetic trees into a network. These models are in general mutually distinct although they do have a significant common overlap which reaches its peak in the case of data derived from *two* phylogenetic trees. To see this, note that if one takes the union of clusters represented by a set of two or more phylogenetic trees, then the reticulation number (or level) required to represent these clusters is in general less than or equal to the reticulation number (or level) required to topologically embed the trees themselves in the network, and this inequality is often strict. However, in the case of a set comprising *exactly* two trees the inequality becomes equality [28]. Hence for data obtained from two trees one could solve the reticulation number minimization and level minimization problems for clusters by using algorithms developed for the problem of topologically embedding the trees themselves into a network. These algorithms are highly efficient and fixed parameter tractable in a practical, as opposed to solely theoretical sense [1, 3, 5, 32]. However, these tree algorithms do not help us with more general cluster sets, because for more than two trees the optima of the cluster and tree models start to diverge. Indeed, the cluster model often saves reticulations with respect to the tree model by weakening the concept of "above" and "below" in the network, which is exactly why the input tree topologies do not generally survive if one atomizes them into their constituent clusters [28]. Moreover, the literature on embedding three or more trees into a network is not yet mature, with articles restricting themselves to preliminary explorations [4, 17, 34]. It therefore seems plausible that the generator approach might be adapted to the tree model (or the other constructive methods mentioned) to yield a unified technique for producing positive complexity results for reticulation number minimization and level minimization, even in the case of many input trees (or data obtained from many input trees).

2 Preliminaries

Consider a set of taxa $\mathcal{X}$, where $|\mathcal{X}| = n$. A *rooted phylogenetic network* (on $\mathcal{X}$), henceforth *network*, is a directed acyclic graph with a single node with indegree zero (the *root*), no nodes with both indegree and outdegree equal to 1, and nodes with outdegree zero (the *leaves*) bijectively labelled by $\mathcal{X}$. In this article we usually identify the leaves with $\mathcal{X}$. The indegree of a node v is denoted $\delta^-(v)$ and v is called a *reticulation* if $\delta^-(v) \geq 2$, otherwise v is a *tree* node. An edge (u, v) is called a *reticulation edge* if its target node v is a reticulation and is called a *tree edge* otherwise. When counting reticulations in a network, we count reticulations with more than two incoming edges more than once because, biologically, these reticulations represent several reticulate evolutionary events. Therefore, we formally define the *reticulation number* of a network $N = (V, E)$ as

$$r(N) = \sum_{v \in V : \delta^-(v) > 0} (\delta^-(v) - 1) = |E| - |V| + 1.$$

A *rooted phylogenetic tree* on $\mathcal{X}$, henceforth *tree*, is simply a network that has reticulation number zero. We say that a network N on $\mathcal{X}$ *displays* a tree T if T can be obtained from N by performing a series of node and edge deletions and eventually by suppressing nodes with both indegree and outdegree equal to 1, see Fig. 2 for an example. We assume without loss of generality that each reticulation has outdegree at least one. Consequently, each leaf has indegree one. We say that a network is *binary* if every reticulation node has indegree 2 and outdegree 1 and every tree node that is not a leaf has outdegree 2.

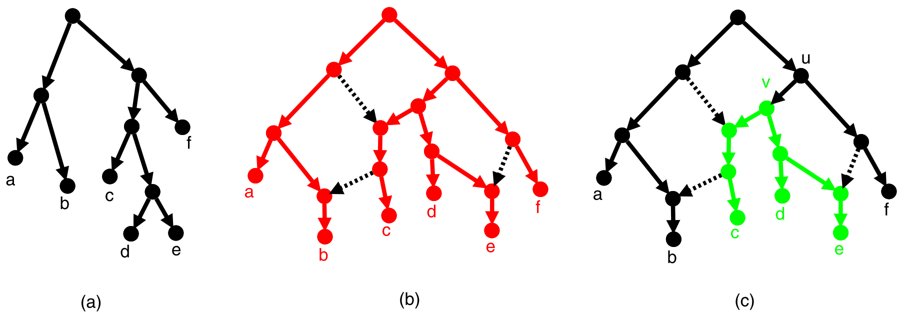

Fig. 2 A phylogenetic tree T (**a**) and a phylogenetic network N (**b**, **c**); (**b**) illustrates in *grey* that N displays T (deleted edges are *dashed*); (**c**) illustrates that N represents (amongst others) the cluster $\{c, d, e\}$ in the softwired sense (*dashed* reticulation edges are "switched off")

Proper subsets of $\mathcal{X}$ are called *clusters*, and a cluster C is a *singleton* if $|C| = 1$. We say that an edge (u, v) of a tree *represents* a cluster $C \subset \mathcal{X}$ if C is the set of leaf descendants of v. A tree T represents a cluster C if it contains an edge that represents C. It is well-known that the set of clusters represented by a tree is a laminar family, often called a *hierarchy* in the phylogenetics literature, and uniquely defines that tree. We say that a network N represents a cluster C "in the softwired sense" if N displays some tree T on $\mathcal{X}$ such that T represents C, see Figs. 2 and 3. In this article we only consider the softwired notion of cluster representation and henceforth assume this implicitly.[2] A network represents a set of clusters $\mathcal{C}$ if it represents every cluster in $\mathcal{C}$ (and possibly more). The set of *all* softwired clusters represented by a network can be obtained as follows. For a network N, we say that a *switching* of N is obtained by, for each reticulation node, deleting all but one of its incoming edges. Given a network N and a switching T_N of N, we say that an edge (u, v) of N represents a cluster C w.r.t. T_N if (u, v) is an edge of T_N and C is the set of labelled leaf descendants of v in T_N. The set of all softwired clusters represented by N, denoted $\mathcal{C}(N)$, is the set of clusters represented by all edges of N w.r.t. T_N, where T_N ranges over all possible switchings [16]. Note that every tree displayed by N corresponds to one or more switchings of N. It is also natural to define that an edge (u, v) of N represents a cluster C if there exists some switching T_N of N such that (u, v) represents C w.r.t. T_N. Note that, in general, an edge of N might represent multiple clusters, and a cluster might be represented by multiple edges of N.

Given a set of clusters $\mathcal{C}$ on $\mathcal{X}$, throughout the article we assume that, for any taxon $x \in \mathcal{X}$, $\mathcal{C}$ contains at least one cluster C containing x and that $\mathcal{C}$ contains at least one non singleton cluster.[3] For a set $\mathcal{C}$ of clusters on $\mathcal{X}$ we define $r(\mathcal{C})$ as $\min\{r(N)|N \text{ represents } \mathcal{C}\}$ and we refer to this as the *reticulation number* of $\mathcal{C}$. The related concept of *level* requires some more background. A directed acyclic graph is *connected* (also called "weakly connected") if there is an undirected path (ignoring edge orientations) between each pair of nodes. A node (edge) of a directed

[2] Alternatively, we say that a network N represents a cluster $C \subset \mathcal{X}$ "in the hardwired sense" if there exists a tree edge (u, v) of N such that C is the set of leaf descendants of v.

[3] Otherwise $\mathcal{C}$ can be trivially represented by the star tree on $\mathcal{X}$.

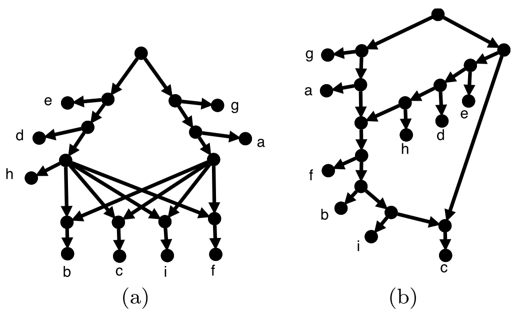

Fig. 3 Two examples of networks that represent, among others, the set of clusters $\mathcal{C} = \{\{a,b,f,g,i\}, \{a,b,c,f,g,i\}, \{a,b,f,i\}, \{b,c,f,i\}, \{c,d,e,h\}, \{d,e,h\}, \{b,c,f,h,i\}, \{b,c,d,f,h,i\}, \{b,c,i\}, \{a,g\}, \{b,i\}, \{c,i\}, \{d,h\}\}$. The network in (**a**) is a simple level-4 network, and the network in (**b**) is a binary simple level-2 network

graph is called a *cut-node* (*cut-edge*) if its removal disconnects the graph. A directed graph is *biconnected* if it contains no cut-nodes. A biconnected subgraph B of a directed graph G is said to be a *biconnected component* if there is no biconnected subgraph $B' \neq B$ of G that contains B. A phylogenetic network is said to be a *level-* $\leq k$ *network* if each biconnected component has reticulation number less than or equal to k.[4] A network is called *simple* if the removal of a cut-node or a cut-edge creates two or more connected components of which at most one is non-trivial (i.e. contains at least one edge). A (simple) level-$\leq k$ network N is called a (simple) *level-k network* if the maximum reticulation number among the biconnected components of N is *precisely* k. For example, the network in Fig. 1 is a level-2 network (which is not simple), the network in Fig. 3(a) is a simple level-4 network and the network in Fig. 3(b) is a simple level-2 network. Note that a tree is a level-0 network. For a set $\mathcal{C}$ of clusters on $\mathcal{X}$ we define the *level* of $\mathcal{C}$, denoted $l(\mathcal{C})$, as the smallest $k \geq 0$ such that there exists a level-k network that represents $\mathcal{C}$. It is immediate that for every cluster set $\mathcal{C}$ it holds that $r(\mathcal{C}) \geq l(\mathcal{C})$, because a level-$k$ network always contains at least one biconnected component containing k reticulations.

We say that two clusters $C_1, C_2 \subset \mathcal{X}$ are *compatible* if either $C_1 \cap C_2 = \emptyset$ or $C_1 \subseteq C_2$ or $C_2 \subseteq C_1$, and *incompatible* otherwise. Consider a set of clusters $\mathcal{C}$. The *incompatibility graph* $IG(\mathcal{C})$ of $\mathcal{C}$ is the undirected graph (V, E) that has node set $V = \mathcal{C}$ and edge set $E = \{\{C_1, C_2\} \mid C_1 \text{ and } C_2 \text{ are incompatible clusters in } \mathcal{C}\}$. We say that a set of taxa $\mathcal{X}' \subseteq \mathcal{X}$ is *compatible* with $\mathcal{C}$ if $\mathcal{X}' = \mathcal{X}$ or every cluster $C \in \mathcal{C}$ is compatible with $\mathcal{X}'$, and *incompatible* otherwise.

We say that a set of clusters $\mathcal{C}$ on $\mathcal{X}$ is *separating* if it is incompatible with all sets of taxa $\mathcal{X}'$ such that $\mathcal{X}' \subset \mathcal{X}$ and $|\mathcal{X}'| \geq 2$.

When we write $f(k)$ we mean "some function that only depends on k". For simplicity we overload $f(k)$ to refer to multiple different functions with this property. We write *poly*(n) to mean "some function $f(n)$ that is polynomial in n", where $|\mathcal{X}| = n$. As in the case of $f(k)$, we often overload this expression. Indeed, the goal of this article is not to derive exact expressions for the running time, but to show that it is bounded above by $f(k) \cdot poly(n)$. It should be noted that the $f(k)$ that we encounter in this article can be *extremely* exponential in k. Also, $|\mathcal{C}|$ can in general be exponentially large as a function of n, but (as we shall see in due course) we may assume

[4]Note that to determine the reticulation number of a biconnected component, the indegree of each node is computed using only edges belonging to this biconnected component.

that $|\mathcal{C}|$ is bounded above by $f(k) \cdot poly(n)$ when the parameter k (reticulation number or level) is fixed. This is indeed obvious for the reticulation number and will be shown to be also true for the level in Sect. 3.2. The next lemma ensures that, if our goal is to find a network representing a set of clusters and minimizing the level or the reticulation number, we can restrict our attention to binary networks:

Lemma 1 [20] *Let N be a network on $\mathcal{X}$. Then we can transform N into a binary network N' such that N' has the same reticulation number and level as N and all clusters represented by N are also represented by N'.*

It is well-known that for every set of taxa $\mathcal{X}$, it is possible to construct a binary network with at most $|\mathcal{X}| - 1$ reticulations which represents every possible cluster on $\mathcal{X}$ [28]. This implies that $r(\mathcal{C})$ and $l(\mathcal{C})$ are well defined for any set of clusters $\mathcal{C}$. Thanks to this observation and to Lemma 1, there exists a binary network N with reticulation number $r(\mathcal{C})$ (or with level $l(\mathcal{C})$ if we are interested in level minimization) that represents $\mathcal{C}$. We henceforth restrict our analysis to binary networks and, except in places where it might cause confusion to not be explicit, we will not emphasize again that we only deal with this kind of network.

3 Minimizing Level is Fixed Parameter Tractable

The aim of this section is to show that level-minimization is fixed parameter tractable. To compute $l(\mathcal{C})$, we will repeatedly query, *"Is $l(\mathcal{C}) = k$? If so, construct a network with level equal to k that represents $\mathcal{C}$"*, where k starts at 0 and is incremented by 1 until the query is answered positively. Assuming that the queries are correctly answered, this process will terminate after $l(\mathcal{C}) + 1$ iterations. Hence, to prove an overall running time of $f(l(\mathcal{C})) \cdot poly(n)$, it is sufficient to show that for each k we can correctly answer the query in time at most $f(k) \cdot poly(n)$. Note that $r(\mathcal{C}) = l(\mathcal{C}) = 0$ if and only if all the clusters in $\mathcal{C}$ are pairwise mutually compatible, which can be easily checked in time $poly(n)$, so we henceforth assume that $k \geq 1$.

The high-level idea is the following. In [16, 31] it is shown that level-k networks can be constructed using a divide and conquer strategy. Informally, the idea is to construct a level-$\leq k$ network for each connected component of the incompatibility graph $IG(\mathcal{C})$ and then to combine these into a single network. The clusters in each connected component first have to be processed, which creates (for each component) a separating set of clusters. From Lemma 1 of [20], we know that, if a level-k network representing a separating set of clusters $\mathcal{C}$ on $\mathcal{X}$ exists, a simple level-k network representing $\mathcal{C}$ has to exist. Moreover, the transformation underpinning Lemma 1 allows us to assume that this simple level-k network is binary. Hence, the divide and conquer strategy essentially reduces to constructing binary simple level-$\leq k$ networks for separating sets of clusters (and then combining them into a single network).

In Sect. 3.1 we show how to construct a simple level-k network in time $f(k) \cdot poly(n)$ from a separating set of clusters. Subsequently we show in Sect. 3.2 how to combine these networks in time $f(k) \cdot poly(n)$ into a single level-k network.

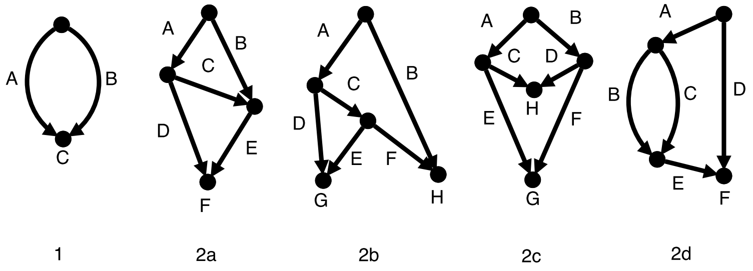

Fig. 4 The single level-1 generator and the four level-2 generators. Here the sides have been labelled with capital letters

3.1 Constructing Simple Networks from Separating Cluster Sets

Before proving the main result of this paper, we need to prove some preliminary results.

Proposition 1 *Given a simple level-k network N and a set of clusters $\mathcal{C}$ on $\mathcal{X}$, checking whether $\mathcal{C}$ is represented by N can be done in time $f(k) \cdot poly(n)$, where $n = |\mathcal{X}|$.*

Proof Note that a simple level-k network contains exactly k reticulations. Thus, there are at most 2^k trees displayed by N and each tree represents at most $2(n-1)$ clusters. This means that $|\mathcal{C}(N)|$ is at most $2^{k+1}(n-1)$. Since N cannot represent $\mathcal{C}$ if $|\mathcal{C}| > |\mathcal{C}(N)|$, checking whether $\mathcal{C} \subseteq \mathcal{C}(N)$ takes at most $f(k) \cdot poly(n)$ time. □

Thus, if $|\mathcal{C}| > 2^{k+1}(n-1)$, since $\mathcal{C}$ is assumed to be separating, it is not possible that $l(\mathcal{C}) = k$ and we can immediately answer "no" to the query.[5] We thus henceforth assume that $|\mathcal{C}| \leq 2^{k+1}(n-1)$ i.e. that $\mathcal{C}$ contains at most $f(k) \cdot poly(n)$ clusters.

If all the leaves of a binary simple level-k network N are removed and all nodes with both indegree and outdegree equal to 1 are suppressed, the resulting structure is called a *level-k generator* as defined in [29]. See Fig. 4 for the level-1 and level-2 generators. The number of level-k generators is bounded by $f(k)$ [8].[6]

The *sides* of a level-k generator are defined as the union of its edges (the *edge* sides) and its nodes of indegree-2 and outdegree-0 (the *node sides*). The number of sides in a generator is bounded by $f(k)$, because the sum of its vertices and edges is linear in k [30].

Definition 1 The set $\mathcal{N}^k$ (for $k \geq 1$) is defined as the set of all networks that can be constructed by choosing some level-k generator G and then applying the following *leaf hanging* transformation to G such that each taxon of $\mathcal{X}$ appears exactly once in the resulting network. (This is essentially identical to the definition given in [30], which is only a superficial refinement of the definition given in [29]).

[5]Recall that, by Lemma 1 of [20], the existence of a level-k network representing a separating set of clusters $\mathcal{C}$ on $\mathcal{X}$ implies that a simple level-k network representing $\mathcal{C}$ has to exist.

[6]Note that the number of level-k generators grows rapidly in k, lying between 2^{k-1} and $k!^2 50^k$ [8].

1. First, for each pair u, v of vertices in G connected by a single edge (u, v), replace (u, v) by a path with $l \geq 0$ internal vertices and, for each such internal vertex w, add a new leaf w', an edge (w, w'), and label w' with some taxon from $\mathcal{X}$. All the taxa added in this way are "on side s" where s is the side corresponding to the edge (u, v). (It is also permitted that the path has zero internal nodes i.e. that the side remains empty).
2. Second, for each pair u, v of vertices in G connected by two edges, treat the two edges as in step 1, but ensure that at least one of the two paths does not have zero internal nodes.
3. Third, for each vertex v of G with indegree 2 and outdegree 0 add a new leaf y, an edge (v, y) and label y with a taxon $x \in \mathcal{X}$; we say "taxon x is on side s" where s is the side corresponding to vertex v.

The main reason for step 2 in Definition 1 is to ensure that multi-edges in generators do not survive in the final network, since our definition of phylogenetic network does not allow multi-edges. The next lemma follows directly from the results in Sect. 3.1 of [29]:

Lemma 2 *The set $\mathcal{N}^k$ (for $k \geq 1$) is equal to the set of all binary simple level-k networks.*

For example, the simple network in Fig. 3(b) has been obtained from generator $2a$ (see Fig. 4) by putting 0 taxa on sides A and D, 1 taxon on side F, 2 taxa on side B and 3 taxa on sides C and E.

By Lemma 1 of [20] and Lemmas 1 and 2, we have the following:

Corollary 1 *Let $\mathcal{C}$ be a separating set of clusters on $\mathcal{X}$ such that $l(\mathcal{C}) \geq 1$. Then there exists a network N in $\mathcal{N}^{l(\mathcal{C})}$ such that N represents $\mathcal{C}$.*

Given a taxon set $\mathcal{X}$, we call any network resulting from adding all taxa in $\mathcal{X}$ to sides of a generator G (in the sense of Definition 1) a *completion* of G on $\mathcal{X}$. Here we call a side that receives ≥ 2 taxa a *long side*, a side that receives 1 taxon a *short side* and a side that receives 0 taxa an *empty side*. Figure 3(b) is thus a completion of generator $2a$, where sides A and D are empty, side F is short, and sides B, C, E are long. Note that in simple networks node sides (such as F in the example) are always short. Indeed, they cannot be empty—otherwise they will violate the definition of a network—and they cannot be long, since simple networks never have two or more taxa with the same parent [20].

Given a generator G, we call *a set of side guesses* for G, denoted by S_G, a set of guesses about the type of each side of G (i.e. whether it is empty, short, or long). A completion N of G on $\mathcal{X}$ *respects* S_G if all sides that are long in S_G receive at least 2 taxa in N, sides that are short in S_G one taxon and empty sides zero taxa. Then we have the following result:

Observation 1 *Searching in the space of all binary simple level-k networks on $\mathcal{X}$ is equivalent to searching in the space of all completions of a level-k generator G* *respecting a set of side guesses S_G, iterating overall all sets of side guesses for a generator and all level-k generators.*

Let G be a level-k generator and let S_G be a set of side guesses for G. We say that the pair (G, S_G) is *side-minimal* w.r.t. a separating cluster set $\mathcal{C}$ on $\mathcal{X}$ and an integer k, if there exists a completion N of G on $\mathcal{X}$ respecting S_G that is a level-k network representing $\mathcal{C}$ and, amongst all simple level-k networks that represent $\mathcal{C}$, N has a minimum number of long sides, and (to further break ties) amongst those networks it has a minimum number of short sides.

We define an *incomplete network* as a generator G, a set of side guesses S_G, a set of *finished sides* (i.e. those sides for which we have already decided that no more taxa will placed on them), a set of *future sides* (i.e. those short and long sides that have had no taxa allocated yet), at most one long side—the *active side*—on which at least one taxon has already been placed but where we might still want to add some more taxa, and information describing the position of already-allocated taxa on the finished and active sides. A *valid completion* of an incomplete network is an assignment of the unallocated taxa to the future sides and (possibly) *above* the taxa already placed on the active side, that respects S_G and such that the resulting network (which we call the *result* of the valid completion) represents $\mathcal{C}$. Informally, the result of a valid completion is any network on $\mathcal{X}$ respecting S_G and representing $\mathcal{C}$ that is obtained by respecting all placements of taxa made thus far.

For example, consider again the network in Fig. 3(b). Let N be the network in that figure and let N' be the network obtained from N by deleting taxa c, d, e and suppressing the resulting vertices with indegree and outdegree both equal to 1. Let G be generator 2a, and let S_G be the set of side-guesses where sides A and D are empty, side F is short, and sides B, C, E are long. Then N' is an incomplete network for (G, S_G) where sides A, B, D, E are finished, F is a future side and C is the active side. We can perform a valid completion of N' by putting taxa d and e above taxon h on side C and then putting c on side F. In this case, N is the result of the completion, although in general an incomplete network might have many valid completions, or none.

Given a cluster set $\mathcal{C}$, we write $x \rightarrow_{\mathcal{C}} y$ if and only if every non-singleton cluster in $\mathcal{C}$ containing x, also contains y. Then we have the following result.

Proposition 2 *Given a separating set of clusters $\mathcal{C}$ on $\mathcal{X}$ and an ordered set of distinct taxa of $\mathcal{X}$ $(x_1, \ldots, x_j)$ such that $j \geq 2$ and $x_i \rightarrow_{\mathcal{C}} x_{i+1}$ for $1 \leq i \leq (j-1)$. Then $x_j \not\rightarrow_{\mathcal{C}} x_1$.*

Proof If $x_i \rightarrow_{\mathcal{C}} x_{i+1}$ for $1 \leq i \leq (j-1)$ and $x_j \rightarrow_{\mathcal{C}} x_1$, this means that the set $\mathcal{X}' = \bigcup_{i=1}^{j} x_i$ is compatible with $\mathcal{C}$. Note that we can have that $\mathcal{X}' = \mathcal{X}$ only if all clusters in $\mathcal{C}$ are singleton clusters, but this is not possible (see footnote 3). So $\mathcal{X}' \subset \mathcal{X}$ and thus, since $|\mathcal{X}'| \geq 2$, we have a contradiction. □

The following observations will be useful to prove Lemma 3.

Observation 2 *Let N be a phylogenetic network on $\mathcal{X}$ representing a set of clusters $\mathcal{C}$ on $\mathcal{X}$ constructed by choosing some level-k generator G and then applying the* leaf hanging *transformation described in Definition 1 to G. If two taxa x and y in $\mathcal{X}$ are on the same side of the generator underlying N and the parent of x is a descendant of the parent of y, then $y \rightarrow_{\mathcal{C}} x$.*

Given a simple phylogenetic network N, we say that a side s' is *reachable* from a side s in N if there is a directed path in the generator underlying N from the head of side s to the tail of side s'.

Observation 3 *Let N be a phylogenetic network on $\mathcal{X}$ representing a set of clusters $\mathcal{C}$ on $\mathcal{X}$ constructed by choosing some level-k generator G and then applying the* leaf hanging *transformation described in Definition 1 to G. Moreover, let x and y be two taxa of $\mathcal{X}$ on the same side s of the generator underlying N such that $y \rightarrow_{\mathcal{C}} x$ and let z be a taxon on a side $s' \neq s$ such that s' is not reachable from s and $z \rightarrow_{\mathcal{C}} x$. Then we have that $z \rightarrow_{\mathcal{C}} y$.*

Proof Since $z \rightarrow_{\mathcal{C}} x$, we know that every non-singleton cluster that contains z also contains x. Now, let C such a cluster. C is represented by some tree T displayed by N, so some edge e in T is such that C is the set of all taxa reachable from directed paths from the head of e. Now, z and x are both in C, so there is a directed path from the head of e to z and a directed path from the head of e to x. Since s' is not reachable from s, the only way that such a directed path can reach x is via the parent of y, hence the fact that $z \rightarrow_{\mathcal{C}} y$. If no cluster C containing z and x exists, since $z \rightarrow_{\mathcal{C}} x$ we have that the only cluster containing z is the singleton cluster $\{z\}$. Then, obviously, $z \rightarrow_{\mathcal{C}} y$ too. □

Observation 4 *Let N be a phylogenetic network on $\mathcal{X}$ representing a set of clusters $\mathcal{C}$ on $\mathcal{X}$ constructed by choosing some level-k generator G and then applying the* leaf hanging *transformation described in Definition 1 to G. Let x and y be two taxa in $\mathcal{X}$ on the same side s of the generator underlying N such that there exists an edge e from the parent of y to the parent of x. Then e represents all clusters in $\mathcal{C}$ containing x but not y.*

Observation 5 *Let $\mathcal{C}$ be a separating cluster set on $\mathcal{X}$. Then every size-2 subset of $\mathcal{X}$ is incompatible with $\mathcal{C}$.*

Let N be a simple phylogenetic network, l a taxon and s a side of the generator G underlying N. We denote by $N(l, s)$ the following operations: If s is a short side, then $N(l, s)$ is simply the network obtained by putting l on side s (in the sense of Definition 1). Otherwise, s is a long side, and then $N(l, s)$ is the network obtained by placing l "just above" the highest taxon on side s. If there are not yet any taxa on side s then we simply let l be the first taxon on side s. (See Fig. 5 for clarification). Note that we exclude from consideration the case where s is a short side that already has a taxon on it.

The core of our proof of fixed-parameter tractability relies on Algorithm 1 (see pseudocode). Let us assume that we have an incomplete network N with an active side s (which is by definition long) such that *all* long sides $s' \neq s$ that are reachable

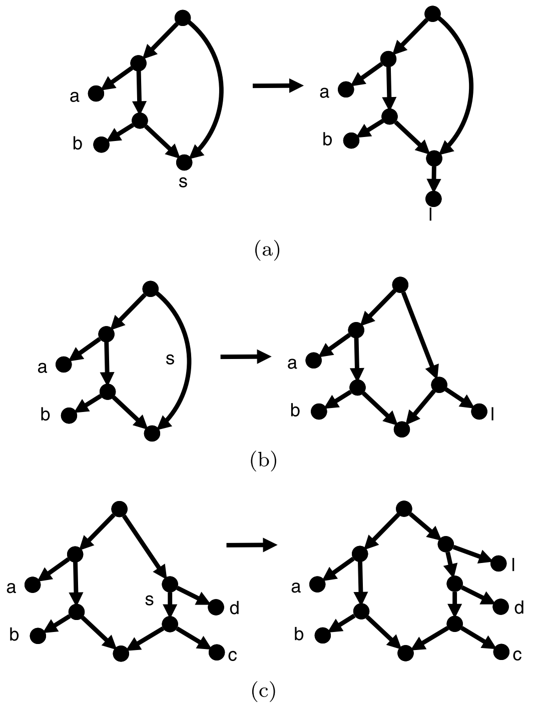

Fig. 5 Three examples of the $N(l,s)$ operation. (**a**) $N(l,s)$ when s is an unfinished short node side; (**b**) $N(l,s)$ when s is an unfinished short edge side (or a long side that does not yet have any taxa); (**c**) $N(l,s)$ when s is a long side that already has at least one taxon

from s, are *finished*. (These preconditions will be motivated in due course.) Informally, Algorithm 1 identifies and returns a set of possible next steps in constructing a valid solution. It decides whether to terminate the side s (and declare it finished), or to add a single new taxon to the top of it. In the case that it decides to add a taxon, it has to decide which specific taxon to add. Furthermore, when this taxon is added, it can force some other taxa to be allocated to unfinished short sides that are "beneath" s, and the algorithm also has to decide how to allocate these taxa. In this way, Algorithm 1 can also cause some short sides to be allocated a taxon, although the algorithm itself is only applied to long sides.[7]

Lemma 3 *Let $\mathcal{C}$ be a separating set of clusters on $\mathcal{X}$ and let k be the first integer for which a level-k network representing $\mathcal{C}$ exists. Let N be an incomplete network such that its underlying generator G and set of side guesses S_G are such that (G, S_G) is side-minimal w.r.t. $\mathcal{C}$ and k, and let s be the active side of N. Then, if a valid completion for N exists, Algorithm 1 computes a set of (incomplete) networks $\mathcal{N}$ such that this set contains at least one network for which a valid completion exists.*

Proof Recall that, from Corollary 1, we can restrict our search to networks in $\mathcal{N}^k$. We write $\mathcal{X}(N)$ to denote the set of taxa present in a (incomplete) network N. For

[7]Indeed, short sides can only be allocated taxa in two ways. Firstly, indirectly via Algorithm 1. Secondly, when there are no longer any unfinished long sides, at the very end of the entire procedure, in Algorithm 3.

Algorithm 1 addOnSide(N, s)

```
1:  $\mathcal{X}' \leftarrow \mathcal{X} \setminus \mathcal{X}(N)$;
2:  $x_i \leftarrow$ the most recent taxon inserted on side $s$;
3:  $L \leftarrow \{l \in \mathcal{X}' \mid l \rightarrow_{\mathcal{C}} x_i\}$;
4:  $L' \leftarrow \{l \in L |$ there does not exist $l' \in L$ such that $l' \neq l$ and $l \rightarrow_{\mathcal{C}} l'\}$;
5:  $U \leftarrow \{s' | s' \neq s$is a side of $N$ that is not yet finished and is reachable from $s\}$;
6:  **for** $l \in L'$ **do**
7:      $S(l) = \bigcup\{C \in \mathcal{C} \mid x_i \in C$ and $l \notin C\}$;
8:      $B(l) = \mathcal{X}' \cap S(l)$.
9:  **if** $U = \emptyset$ **then**
10:     **if** $|L'| \neq 1$ **then** declare $s$ as finished in $N$ and **return** $N$;
11:     $l \leftarrow$ removeFirst($L'$);
12:     **if** $B(l) \neq \emptyset$ **then** declare $s$ as finished in $N$ and **return** $N$;
13:     **if** $N(l, s)$ does not represent $\mathcal{C}$ restricted to $\mathcal{X}(N) \cup \{l\}$ **then**
14:         declare $s$ as finished in $N$ and **return** $N$;
15:     **else return** $N(l, s)$;
16: **else**
17:     **if** $L' = \emptyset$ **then**
18:         declare $s$ as finished in $N$ and **return** $N$;
19:     **if** $|L'| \geq 2$ **then**
20:         $\mathcal{N} \leftarrow N$, where $s$ is declared as finished;
21:         **if** $|L'| \leq |U|$ **then**
22:             $\mathcal{N}' \leftarrow$ the set of networks obtainable from $N$ by allocating all taxa
                in $L'$ to sides in $U$;
23:             $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'$;
24:         **if** $|L'| - 1 \leq |U|$ **then**
25:             **for** $l \in L'$ **do**
26:                 $\mathcal{N}' \leftarrow$ the set of networks obtainable from $N(l, s)$ by allocating
                    all taxa in $L' \setminus \{l\}$ to sides in $U$;
27:                 $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'$;
28:         **return** $\mathcal{N}$;
29:     **if** $|L'| = 1$ **then**
30:         $l \leftarrow$ removeFirst($L'$);
31:         **if** $B(l) \neq \emptyset$ **then**
32:             $\mathcal{N} \leftarrow N$, where $s$ is declared as finished;
33:             **for** each side $s' \in U$ **do**
34:                 $\mathcal{N} \leftarrow \mathcal{N} \cup \{N(l, s')\}$;
35:             **if** $|B(l)| \leq |U|$ **then**
36:                 $\mathcal{N}' \leftarrow$ the set of all networks obtainable from $N(l, s)$ by allocat-
                    ing all taxa in $B(l)$ to sides in $U$;
37:                 $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'$;
38:         **else**
39:             **if** $N(l, s)$ does *not* represent $\mathcal{C}$ restricted to $\mathcal{X}(N) \cup \{l\}$ **then**
40:                 $\mathcal{N} \leftarrow N$, where $s$ is declared $s$ as finished;
41:                 **for** each side $s' \in U$ **do**
```

```
42:                 $\mathcal{N} \leftarrow \mathcal{N} \cup \{N(l, s')\}$;
43:         else
44:             $D \leftarrow$ an arbitrary set of $|U|$ taxa such that $D \cap \mathcal{X} = \emptyset$;
45:             $N^*(l, s) \leftarrow$ a network obtained from $N(l, s)$ by arbitrarily and
                bijectively assigning each taxon in $D$ to a side in $U$;
46:             $\bar{\mathcal{C}} \leftarrow \{C \in \mathcal{C}$ such that $x_i \in C, l \notin C$, and $C \subseteq \mathcal{X}(N)\}$;
47:             if $N^*(l, s)$ does not represent $\bar{\mathcal{C}}$ then
48:                 $\mathcal{N} \leftarrow N$, where $s$ is declared $s$ as finished;
49:                 for each side $s' \in U$ do
50:                     $\mathcal{N} \leftarrow \mathcal{N} \cup \{N(l, s')\}$;
51:             else
52:                 $\mathcal{N} \leftarrow N(l, s)$;
53:     return $\mathcal{N}$;
```

a set of clusters $\mathcal{C}$ on $\mathcal{X}$ and a subset $\mathcal{X}' \subseteq \mathcal{X}$, we define the *restriction* of $\mathcal{C}$ to $\mathcal{X}'$ as $\{C \cap \mathcal{X}' | C \in \mathcal{C}\}$. We start the proof by analyzing the case when $U = \emptyset$ (see Algorithm 1 for the definition of $\mathcal{X}'$, U, $B(l)$, etc.). □

Case $U = \emptyset$ Suppose $|L'| \neq 1$. If $\boldsymbol{|L'| = 0}$ then there are two possibilities. If $L = \emptyset$ then clearly no taxon l can be placed directly above x_i on s, because that would mean $l \rightarrow_{\mathcal{C}} x_i$, and thus $l \in L$, contradiction. Hence the only correct move is to declare that the side s is finished and return N. If $L \neq \emptyset$ then, since $|L'| = 0$, we have that, for every $l \in L$ there exists some $l' \in L$ such that $l \neq l'$ and $l \rightarrow_{\mathcal{C}} l'$. Clearly the $\rightarrow_{\mathcal{C}}$ relation is not allowed to create cycles in L, because otherwise the set of taxa in the cycle would form a cluster compatible with $\mathcal{C}$ (see Proposition 2). Suppose we start at an arbitrary taxon in L and perform a non-repeating walk on the taxa of L by following the $\rightarrow_{\mathcal{C}}$ relation. Given that L is of finite size and this walk cannot visit a taxon of L that it has already visited earlier in the walk (thus creating a cycle), we will find a taxon $l \in L$ such that there is no $l' \in L$ such that $l \neq l'$ and $l \rightarrow_{\mathcal{C}} l'$, meaning that $l \in L'$, contradiction. So the case that $L \neq \emptyset$ but $L' = \emptyset$, cannot actually happen. Now, consider the case that $\boldsymbol{|L'| \geq 2}$. Algorithm 1 will always end the side s and return N in this case. Indeed, no valid completion of N can have some taxon p that has not yet been allocated above x_i on side s. Suppose this is not true. Clearly, from Observation 2, $p \rightarrow_{\mathcal{C}} x_i$, so $p \in L$. In this case, all taxa in L' are either equal to p, or underneath p and above x_i. Indeed, let $l \neq p$ be a taxon in L' and suppose, for the sake of contradiction, that l is above p on side s or on another side s'. If l is above p on side s, then from Observation 2 we have that $l \rightarrow_{\mathcal{C}} p$. If l is on another side s', the fact that $|U| = 0$ implies that there is no room under side s so, by Observation 3 we have that $l \rightarrow_{\mathcal{C}} p$. Thus, in both cases (i.e. if l is above p on side s or on a different side s') we have that $l \rightarrow_{\mathcal{C}} p$, meaning that $l \notin L'$, contradiction. We can hence conclude that each taxon in L' is either equal to p, or underneath p and above x_i in any completion of N where p is on s. But, however one arranges two or more taxa on one side, at least one taxon will imply another taxon in the sense of the $\rightarrow_{\mathcal{C}}$ relation. More formally, in any case there exist two taxa l and l' in L' such that $l \neq l'$

and $l \rightarrow_{\mathcal{C}} l'$. This implies that $l \notin L'$, contradiction. This concludes the correctness of the case $|L'| \neq 1$.

We now consider the case when $\boldsymbol{|L'| = 1}$. Let l be the only taxon in L'. In this case, Algorithm 1 will return N if $\boldsymbol{B(l) \neq \emptyset}$. Indeed, no valid completion of N exists where one or more taxa are placed above x_i on s. Suppose this is wrong. In that case, observe that in every valid completion l always has to be the taxon directly above x_i. Indeed, if there was some valid completion such that l is not directly above x_i, then there would exist some taxon $l' \neq l$ such that $l' \rightarrow_{\mathcal{C}} x_i$ (from Observation 2) and $l \rightarrow_{\mathcal{C}} l'$ (as before, this follows from the fact that $U = \emptyset$ and from Observations 2 and 3). This would mean that $l \notin L'$, contradiction. So we assume that l is directly above x_i. Now, since $B(l) \neq \emptyset$, there is some cluster in the input that contains x_i, does not contain l, and contains some not-yet allocated taxon distinct from l. From Observation 4, the only edge that can represent such a cluster is the edge e between the parents of x_i and l. But all the clusters represented by e consist only of already-allocated taxa, because $U = \emptyset$. This means that adding l on side s will only lead us to construct non-valid completions. Hence we conclude that, if $B(l) \neq \emptyset$, all valid completions of N do not contain any other taxon on s and ending the side s is the right choice.

Now consider the case $\boldsymbol{B(l) = \emptyset}$ and let $\mathcal{C}'$ be $\mathcal{C}$ restricted to $\mathcal{X}(N) \cup \{l\}$. If $\boldsymbol{N(l, s)}$ **does not represent** $\boldsymbol{\mathcal{C}'}$ we are definitely correct to declare the side s as finished and return N. Indeed, all valid completions of N do not contain any other taxon on s. Suppose it is not correct. Then there exists a valid completion of N where at least one taxon is above x_i on s. Again, for the same reasons as above we assume that l is always the taxon directly above x_i. Since $N(l, s)$ does not represent $\mathcal{C}'$, and this incompatibility cannot be eliminated by adding more taxa, we conclude that there are no valid completions of N with taxa above x_i on side s. Hence, ending the side s is the only correct option. Suppose now that $\boldsymbol{N(l, s)}$ **does represent** $\boldsymbol{\mathcal{C}'}$; Algorithm 1 adds l above x_i on side s, and does *not* declare s as finished. This conclusion can only be incorrect if *all* valid completions require that l is *not* directly above x_i. We observe that in any valid completion of N there can be no taxon $l' \neq l$ directly above x_i on s, because otherwise, as before, since $U = \emptyset$ we will have that $l \rightarrow_{\mathcal{C}} l' \rightarrow_{\mathcal{C}} x_i$ and hence $l \notin L'$, contradiction. So all valid completions terminate the side at x_i. Let N' be an arbitrary valid completion of N and denote by N'' the network obtained from N' by moving l, wherever it is, just above x_i. Firstly, we claim that N'' still represents $\mathcal{C}$. Recall that $l \rightarrow_{\mathcal{C}} x_i$, so the only potential problem is with clusters in $\mathcal{C}$ that contain x_i but do not contain l. Let C be such a cluster not represented by N''. Suppose $C \nsubseteq \mathcal{X}(N) \cup \{l\}$. But in this case we would have $B(l) \neq \emptyset$ in N, contradiction. So the only possibility is that $C \subseteq \mathcal{X}(N) \cup \{l\}$. Clearly C was in $\mathcal{C}'$ and was thus represented by $N(l, s)$. Moreover, from Observation 4, the edge that represents C in $N(l, s)$ is the edge between the parents of l and x_i. Given that $U = \emptyset$, no more taxa can be added "underneath" side s and this edge still represents C in N'' because N' is a valid completion of N. Hence moving l in the way *is* safe in terms of cluster representation.

Secondly, we claim that moving l in this way does not alter the side types i.e. the empty/short/long sides before moving l remain empty/short/long after moving l. To see this, note that moving l from its original location reduces the number of taxa by 1 on some side, and increases the number of taxa of s by 1. Side s is by assumption already long, so remains so. The side of N' containing l cannot change from being long to being short in N'', because this lowers the total number of long sides, and by assumption the pair (G, S_G) underlying N is side-minimal. Similarly it cannot change from being short to being empty, because this leaves the number of long sides the same but reduces the number of short sides, again contradicting the assumption that (G, S_G) is side-minimal. Combining these two claims—that moving l is safe for cluster representation and does not alter the side types nor the underlying generator—let us conclude that there *is* a valid completion for N in which l is placed directly above x_i. Hence it is correct to add l above x_i on side s, and *not* to declare s as finished.

Case $\boldsymbol{U \neq \emptyset}$ The case $\boldsymbol{|L'| = 0}$ is identical to the corresponding subcase when $U = \emptyset$. This means that in this case it is always correct to declare the side s as finished and return N.

Consider now the case $\boldsymbol{|L'| \geq 2}$. Observe firstly that, if some taxon $l \in L'$ is placed directly above x_i, then all remaining taxa in L' *must* be allocated to sides in U. To see why this is, note that for every $l' \in L'$ we have that $l' \to_{\mathcal{C}} x_i$. So, if $l' \neq l$ is placed above l on s or on a side not in U, then, from Observation 2 and 3 we would have that $l' \to_{\mathcal{C}} l \to_{\mathcal{C}} x_i$, contradicting the fact that l' is in L'. We only need to show that, if a valid completion for N exists, then the set $\mathcal{N}$ contains a network for which there exists a valid completion. Note that $\mathcal{N}$ contains (line 20) N, where s is declared as finished, (lines 21–23) all possible networks obtained from N by allocating all taxa in L' to sides in U and (lines 24–27) all possible networks obtained from $N(l, s)$ by allocating all taxa in $L' \setminus \{l\}$ to sides in U, iterating over all $l \in L'$. The only case that these three sets do not describe, is when every valid completion has a taxon $p \notin L'$ directly above x_i, but at least one taxon $l \in L'$ is not mapped to U. But this implies, similarly to the case $|U| = 0$, that $l \to_{\mathcal{C}} p \to_{\mathcal{C}} x_i$, so $l \notin L'$, contradiction. Hence this case cannot happen, and the three sets actually describe all possible outcomes in this situation. So at least one of them will contain a network with a valid completion in the case N does have a valid completion.

Consider now the case $\boldsymbol{|L'| = 1}$. We begin with the subcase $\boldsymbol{B(l) \neq \emptyset}$. Similar to previous arguments we know that, if we place l (the only element in L') directly above x_i, all taxa in $B(l)$ have to be allocated to U. This holds because, from Observation 4, any cluster that contains x_i but not l is represented by the edge between the parents of l and x_i. If $B(l) \neq \emptyset$, the set $\mathcal{N}$ is composed of (line 32) N, where s is declared as finished, (lines 33–34) all possible networks obtained from N by allocating l to a side in U and (lines 35–37) all possible networks obtained from $N(l, s)$ by allocating all taxa in $B(l)$ to sides in U. Observe that the only situation that these three guesses do not describe, is when some taxon $p \neq l$ is placed above x_i and l is not mapped to U. But in this case we would have that $l \to_{\mathcal{C}} p \to_{\mathcal{C}} x_i$, contradicting the fact that l is in L'. So $\mathcal{N}$ does again describe all possible outcomes.

This leaves us with the very last subcase, $|L'| = 1$ and $\boldsymbol{B(l) = \emptyset}$. The subcase when $\boldsymbol{N(l, s)}$ **does *not* represent** $\boldsymbol{\mathcal{C}}$ restricted to $\mathcal{X}(N) \cup \{l\}$ is actually fairly straightforward. It is clear that l cannot be placed in this position in a valid completion. Hence the only two situations that line 40 and lines 41–42 do not describe, is when some element $p \neq l$ is placed directly above x_i, and l is not mapped to U. But, as before, this implies that $l \rightarrow_{\mathcal{C}} p \rightarrow_{\mathcal{C}} x_i$, which as we have seen is not possible. So the only remaining subcase is when $|L'| = 1$, $B(l) = \emptyset$ and $\boldsymbol{N(l,s)}$ ***does*** **represent** $\boldsymbol{\mathcal{C}}$ restricted to $\mathcal{X}(N) \cup \{l\}$. Now, consider the network $N^*(l,s)$. Informally the dummy taxa in $N^*(l,s)$ act as "placeholders" for taxa that will only later in the algorithm be mapped to U. We do not know exactly what these taxa will be, but we know that they will definitely be there. Consider a cluster $C \in \bar{\mathcal{C}}$. If $N^*(l,s)$ does not represent C then this must be because of the dummy taxa, because we know that $N(l,s)$ did represent $\mathcal{C}$ restricted to $\mathcal{X}(N) \cup \{l\}$. Note that this holds irrespective of the true identity of the dummy taxa. Hence, C will never be represented by any completion of $N(l,s)$. For this reason we conclude that, if $N^*(l,s)$ does not represent $\bar{\mathcal{C}}$, it is definitely correct to declare the side s finished (line 48) or allocate l to a side in U (lines 49–50).

Finally, suppose $N^*(l,s)$ does represent $\bar{\mathcal{C}}$. This is the flip-side of the previous argument. Whatever the true identity of the dummy taxa, every valid completion of $N(l,s)$ will represent every cluster in $\bar{\mathcal{C}}$. Let N' be an arbitrary valid completion of N and denote by N'' the network obtained from N' by moving l, wherever it is, just above x_i. Now, as we did earlier we argue that in this case it is "safe" to put l directly above x_i. Indeed, because $B(l) = \emptyset$, the only clusters that might not be represented in N'' are clusters in $\bar{\mathcal{C}}$. But we have shown that when l is placed directly above x_i all the clusters in $\bar{\mathcal{C}}$ are represented regardless of how we complete the rest of the network. Secondly, we argue just as before that moving l in this way cannot alter the side types. So if we place l on side s directly above x_i there must still exist a valid completion. This concludes the proof of the lemma. □

Algorithm 2 will repeatedly call Algorithm 1 until it finally declares the active side finished. The algorithm works with sets of networks because each execution of Algorithm 1 potentially returns a set of (incomplete) networks, reflecting the various different decisions that Algorithm 1 can make. Note that Algorithm 2 is only executed for long sides. Also, we note that during the execution of Algorithm 2 a call to Algorithm 1 might return a network in which the side s is finished but has fewer than two taxa on it. Such an outcome violates the assumption that side s is long. In such a case we can easily detect this and cease exploring this particular branch of the search tree. (Throughout the algorithms in this paper there are actually many such situations in which we can easily detect that we are exploring a wrong search path, and subsequently prune the search tree. However, to keep the exposition clear we have not discussed these explicitly).

Algorithm 2 completeSide(N, s)

```
1: $\mathcal{N} \leftarrow N$;
2: while there exists $N \in \mathcal{N}$ such that $s$ is not finished in $N$ do
3:     $\mathcal{N} \leftarrow \mathcal{N} \setminus N$;
4:     $\mathcal{N} \leftarrow \mathcal{N} \cup$ addOnSide($N, s$);
```

Lemma 4 *Let $\mathcal{C}$ be a separating set of clusters on $\mathcal{X}$ and let k be the first integer for which a level-k network representing $\mathcal{C}$ exists. Let N be an incomplete network such* *that its underlying generator G and set of side guesses S_G are such that (G, S_G) is side-minimal w.r.t. $\mathcal{C}$ and k, and let s be the active side of N that contains only a single taxon. Algorithm 2 computes in $f(k) \cdot poly(n)$ time a set of (incomplete) networks $\mathcal{N}$ for which* ***s is a finished side****, such that $\mathcal{N}$ contains at least one network for which there exists a valid completion (if any exists).*

Proof The correctness follows from Lemma 3. We now prove the running time.

First, note that the size of the set $\mathcal{N}$ returned by Algorithm 1 is bounded by $f(|U|)$. This is evident for the sets $\mathcal{N}$ constructed on lines 33–34, 41–42 and 49–50 but it holds also for the sets $\mathcal{N}'$ constructed respectively on lines 21–23, 24–27 and 35–37, since these sets are constructed only if, respectively, $|L'| \leq |U|$, $|L'| - 1 \leq |U|$ or $|B(l)| \leq |U|$. Since in all other cases $|\mathcal{N}| = 1$, the size of the set $\mathcal{N}$ returned by Algorithm 1 is indeed bounded by $f(|U|)$. Since the number of sides in a generator is bounded by $f(k)$ and U is a subset of the short sides of the generator (which follows from the fact that all long sides reachable from s are assumed to be finished), we have that $|U|$ is bounded by $f(k)$. From that and from Proposition 1, it follows that the running time of Algorithm 1 is $f(k) \cdot poly(n)$.

Second, note that, each time that Algorithm 1 returns a set of networks $\mathcal{N}$ such that $|\mathcal{N}| > 1$, $|U|$ decreases or s is declared as finished. Additionally, when $U = \emptyset$, then Algorithm 1 returns only one network per call and we have at most $O(n)$ of these calls (because either s is declared finished or a new taxon is added to s). Since the search tree contains $f(k) \cdot poly(n)$ nodes and $f(k) \cdot poly(n)$ time is needed for each node of the search tree, the running time of Algorithm 2 is bounded by $f(k) \cdot poly(n)$. $\square$

We will subsequently use the term *lowest side* to denote an unfinished long side such that there is no other unfinished long side $s' \neq s$ that is reachable from s. The following lemma is basically the fixed parameter tractable version of Lemma 3 from [20]. It proves that Algorithm 3, which is the top-level part of the overall procedure, is FPT. Informally, the algorithm guesses the backbone topology of the network we are constructing (i.e. the level-k generator); guesses which sides should be long, short or empty; repeatedly picks a long side to allocate taxa to; guesses the first taxon on that side; completes the side using Algorithm 2; and collapses the taxa on that side into a single meta-taxon once it is finished. When all long sides are finished, Algorithm 3 guesses how to allocate taxa to any remaining unfinished short sides.

Lemma 5 *Let $\mathcal{C}$ be a separating set of clusters on $\mathcal{X}$. Then, for every fixed $k \geq 0$, Algorithm 3 determines whether a level-k network exists that represents $\mathcal{C}$, and if so, constructs such a network in time $f(k) \cdot poly(n)$.*

Proof Algorithm 3 starts by choosing a level-k generator G and a set of side guesses S_G ‘in increasing side order”, i.e. generators and sets of guesses are analyzed in such a way that generators with a smaller number of sides and sets of side guesses with a smaller number of long sides, and (to further break ties) short sides, are analyzed first. This implies that a side-minimal pair (G, S_G), if any exists, is analyzed before any other pair (G', S'_G) for which a valid completion exists. This is done to be able to apply Lemma 4.

Then (see lines 4–18), the algorithm constructs a set of *complete* networks, i.e. simple level-k networks where each short side has received a single taxon and each long side at least two, and returns the first of them that represents $\mathcal{C}$, if any exists.

Algorithm 3 ComputeLevel-$k(\mathcal{C})$

```
1: for each level-$k$ generator $G$ in increasing side order /* i.e. generators with a smaller number of sides are analyzed first */ do
2:     for each set of side guesses $S_G$ in increasing side order /* i.e. sets of side guesses with a smaller number of long sides, and (to further break ties) short sides, are analyzed first */ do
3:         $\mathcal{N} \leftarrow (G, S_G)$;
4:         while there exists $N \in \mathcal{N}$ such that $N$ contains a lowest side $s$ do
5:             $\mathcal{N} \leftarrow \mathcal{N} \setminus N$;
6:             for each $l^- \in \mathcal{X}(s)^-$ /* $\mathcal{X}(s)^-$ denotes the set of all taxa in $\mathcal{X}$ that are candidates to be the first taxon on side $s$ */ do
7:                 $\mathcal{N}' \leftarrow$ completeSide($N(l^-, s), s$);
8:                 $\mathcal{N}'' \leftarrow$ the networks in $\mathcal{N}'$ where $s$ contains more than one taxon;
9:                 for each $N \in \mathcal{N}''$ do
10:                    collapse all taxa on side $s$ into a single meta-taxon $S$ and
                       adjust the cluster set accordingly;
11:                $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}''$;
12:        if $|\mathcal{N}| > 0$ then
13:            while there exists $N \in \mathcal{N}$ do
14:                de-collapse the collapsed sides;
15:                $\mathcal{N}' \leftarrow$ the set of networks obtainable from $N$ by allocating the
                       taxa in $\mathcal{X} \setminus \mathcal{X}(N)$ to any short sides that have not yet been
                       allocated a taxon;
16:                if there is a network $N' \in \mathcal{N}'$ representing $\mathcal{C}$ then
17:                    return $N'$;
18:                $\mathcal{N} \leftarrow \mathcal{N} \setminus N$;
19: return $\emptyset$;
```

Note that lines 10 and 14 of the algorithm are only a technical step. Indeed, when we declare a side s as finished, we assume that we will never alter that side again. Hence it does not change the analysis if we collapse all the taxa on side s into a single meta-taxon. That is, if we have decided that the taxa on the side s are—from the bottom to the top—$l^-, x_1, \ldots, x_l$ we simply replace all these taxa by a single new taxon S and replace $l^-, x_1, \ldots, x_l$ by S in any clusters in $\mathcal{C}$ that they appear in (line 10). This collapsing step ensures that the set of sides reachable from the current lowest side are always empty or short sides. This will be helpful when proving the running time of Algorithm 3, see below. Note that $\mathcal{C}$ stays separating after the collapsing, since the taxa on s are such that $x_l \rightarrow_{\mathcal{C}} \cdots \rightarrow_{\mathcal{C}} x_1 \rightarrow_{\mathcal{C}} l^-$. When we are finished allocating all the taxa in $\mathcal{X}$ and are ready to check whether the resulting final network represents $\mathcal{C}$ we can simply de-collapse all the S i.e. "unfold" all the long sides that we have collapsed (line 14). This means that the correctness of Algorithm 3 follows by Observation 1 and Lemma 4.

We now need to prove the correctness of the running time. First, note that the number of pairs (G, S_G) to consider is bounded by $f(k)$ since both the number of generators and the number of sides per generator are bounded by $f(k)$.

We now need to prove that the size of $\mathcal{X}(s)^-$ is at most $f(k)$ for all sides s i.e. that the number of taxa that might be the first taxon l^- on side s, is not too big. So let l^- be any taxon which can fulfil this role, and let x be the taxon directly above l^- on side s. (The taxon x must exist because we assume that s is long). Clearly, $x \rightarrow_{\mathcal{C}} l^-$. By line 10, we have that the only sides reachable from side s are short and empty sides. Moreover, we know from Observation 5 that, because $\mathcal{C}$ is separating, there is some non-singleton cluster $C \in \mathcal{C}$ such that $l^- \in C$ but $x \notin C$. By Observation 4, such a cluster C has to be represented by the edge e between the parents of x and l^-. Now, any cluster represented by e can only contain taxa that are reachable from e by a directed path. The only sides that are reachable from side s are short and empty sides, so the cluster C can only contain at most $f(k)$ taxa (because there are at most $f(k)$ short sides). So we know that l^- is in some cluster C, and that C is "small" in the sense that its size is bounded above by $f(k)$. So if we take all "small" clusters, and let $\mathcal{X}(s)^-$ be their union, we know that we could simply try taking every element in $\mathcal{X}(s)^-$ and guessing that it is equal to l^-. To ensure that we do not use too many guesses, we have to show that $|\mathcal{X}(s)^-|$ is bounded by $f(k)$. To see that this holds, consider the question: how many taxa are only in clusters that contain at most c taxa? Observe that on every long side only the c taxa furthest away from the root are *potentially* in such clusters. Any taxon closer to the root on a long side cannot possibly be in a cluster of size at most c, because if it is in a cluster then so are at least c other taxa too. Hence there are at most $f(k)$ taxa that can be involved in "small" clusters: the taxa on the short sides and the taxa at the bottom of the long sides. So we have that $|\mathcal{X}(s)^-|$ is bounded by $f(k)$ and we can guess l^- with at most $f(k)$ guesses.

The collapsing and de-collapsing steps (lines 10 and 14) can be done in $f(k) \cdot poly(n)$ time, as well as completing each side s (line 7), by Lemma 4. Moreover, by Proposition 1, also checking whether N represents $\mathcal{C}$ takes $f(k) \cdot poly(n)$ time. Additionally, the allocation of remaining taxa to the unfinished short sides (line 15) takes a time bounded by $f(k)$. Indeed, if we have a number of not yet assigned taxa bigger than the number of unfinished short sides we can cease exploring this particular branch of the search tree. This means that, if the size of $\mathcal{N}$ is bounded by $f(k)$, then the entire algorithm can be executed in $f(k) \cdot poly(n)$ time. And this is indeed the case, since $|\mathcal{X}(s)^-|$ and (summing over all iterations) the total number of lowest sides are bounded by $f(k)$ and each time that a side is completed, the number of unfinished long sides decreases by 1. □

A Comment on the Running Time We have shown that the running time of Algorithm 3 is at most $f(k) \cdot poly(n)$. We wish to emphasize that, due to the wholly theoretical nature of the algorithm, the $f(k)$ term is astronomical. To give some indication of this, the number of level-k generators for $k = 1, 2, 3, 4, 5$ is 1,4,65,1993,91454 respectively [20], and guessing the correct level-k generator is only the first guess in an extensive guessing strategy (which also requires us to guess, amongst other things, whether a side is long, short or empty; the identity of the first taxon on a long side; and all possible ways of completing the currently active long side). The $poly(n)$ term is much more reasonable. To start with, after factoring out $f(k)$ terms $|\mathcal{C}|$ is only linearly large in $|\mathcal{X}| = n$ (see Proposition 1). Secondly, the only purely polynomial operations in the entire procedure are "housekeeping" tasks, such as determining whether a network represents a set of clusters, constructing the $\rightarrow_{\mathcal{C}}$ relation, identifying sets such as L and L' (in Algorithm 1), and collapsing sets of taxa. These are all low-degree polynomial operations and we estimate that the $poly(n)$ term is no larger than $O(n^5)$. To put this in perspective, the CASS algorithm (which also works with clusters, but which is not guaranteed to be optimal for $k > 2$ [20]) has a running time at least $O(n^{3k})$ [31]. Hence, already for $k = 2$ the $poly(n)$ term in our FPT algorithm is faster than the corresponding term in the running time of CASS.

3.2 From Simple Networks to General Networks

To prove the fixed parameter tractability of constructing general level-k networks, we need to introduce a few other concepts. The most important is the concept of a *decomposable network*.

Definition 2 Let $\mathcal{C}$ be a set of clusters on a taxon set $\mathcal{X}$ with incompatibility graph $IG(\mathcal{C})$ and let N be a phylogenetic network that represents $\mathcal{C}$. N is said to be *decomposable* w.r.t. $\mathcal{C}$ if and only if there exists a cluster-to-edge mapping $\alpha : \mathcal{C} \rightarrow E(N)$ such that, for any two clusters $C_1, C_2 \in \mathcal{C}$, C_1 and C_2 lie in the same connected component of $IG(\mathcal{C})$ if and only if the two tree edges $\alpha(C_1)$ and $\alpha(C_2)$ that represent C_1 and C_2 are contained in the same biconnected component of N.

Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$ with incompatibility graph $IG(\mathcal{C})$. The set of *backbone clusters* associated with $\mathcal{C}$ is defined as

$$B(\mathcal{C}) = \{\mathcal{X}(\mathcal{C}') \mid \mathcal{C}' \text{ is a connected component of } IG(\mathcal{C})\},$$

where $\mathcal{X}(\mathcal{C}') = \bigcup_{C \in \mathcal{C}'} C$ denotes the set of all taxa in $\mathcal{C}'$. Since the set $B(\mathcal{C})$ is compatible [16], we have the following result. (Note that the bound we give here is probably not tight; our goal here is simply to show that it is at most $f(k) \cdot poly(n)$.)

Proposition 3 *Given a decomposable level-k network N representing a set of clusters $\mathcal{C}$ on $\mathcal{X}$ such that the number of biconnected components of N is equal to the number of connected components of $IG(\mathcal{C})$. Then N can contain at most $2^{k+3} \cdot (n-1)^2$ clusters.*

Proof The fact that $B(\mathcal{C})$ is compatible ensures that the size of $B(\mathcal{C})$ is at most $2(n-1)$. In the following we will prove that the number of connected components of $IG(\mathcal{C})$ is at most $4(n-1)$. To prove this, we show that it is impossible to have two non-trivial connected components of $IG(\mathcal{C})$ (i.e. two connected components containing more than once cluster each), say $\bar{\mathcal{C}}$ and $\bar{\mathcal{C}}'$, such that $\mathcal{X}(\bar{\mathcal{C}}) = \mathcal{X}(\bar{\mathcal{C}}')$. For the sake of contradiction, let us suppose that two such components $\bar{\mathcal{C}}$ and $\bar{\mathcal{C}}'$ exist. Let $C_1 \in \bar{\mathcal{C}}$

and $C'_1 \in \bar{\mathcal{C}}'$ be two clusters such that $C_1 \cap C'_1 \neq \emptyset$. Since C_1 and C'_1 are compatible, we can suppose w.l.o.g. that $C'_1 \subset C_1$. Let C'_2 be another cluster of $\bar{\mathcal{C}}'$ incompatible with C'_1 (C'_2 exists because $\bar{\mathcal{C}}'$ is not trivial). Then, since $C'_1 \cap C'_2 \neq \emptyset$ we have that $C_1 \cap C'_2 \neq \emptyset$. But C'_2 cannot be a superset of C_1 so we have that $C'_2 \subset C_1$. Reiterating this reasoning we obtain that $\mathcal{X}(\bar{\mathcal{C}}') \subseteq C_1$. Since $\bar{\mathcal{C}}$ is not trivial, there exists another cluster C_2 in $\bar{\mathcal{C}}$ that is incompatible with C_1. So there exists at least one taxon in $\mathcal{X}(\bar{\mathcal{C}})$ that is not in C_1 and we cannot have that $\mathcal{X}(\bar{\mathcal{C}}) = \mathcal{X}(\bar{\mathcal{C}}')$, contradiction. This means that each non-singleton backbone cluster can correspond to two connected components, one trivial and one not. Then we have at most $4(n-1)$ connected components in $IG(\mathcal{C})$, and thus $4(n-1)$ biconnected components.

We now prove that each biconnected component B of N can represent at most $2^{k+1}(n-1)$ clusters. To see that, let us denote by V' the set of nodes of N that are not in B but whose parents are in B and, for each $v \in V'$, denote by $\mathcal{X}(v)$ the set of all leaves in N that are reachable by directed paths from v. It is easy to see that the set V' has a particularity: for each node $v \in V'$ we have that, no matter which switching T_N is chosen, there exists a path in the switching between v and each taxon $u \in \mathcal{X}(v)$. Indeed, if this was not true, we will have that v has to be in B, a contradiction. Then the network N can be modified in the following way: For each node $v \in V'$, label it with the set $\mathcal{X}(v)$ and delete all the outgoing edges of v. Let N' be the rooted phylogenetic network rooted at the root of the biconnected component B. Because of the peculiarity of the nodes in V', N' represents the same cluster set as B in N. With a line of reasoning similar to that used in Proposition 1, it is easy to see that B can represent at most $2^{k+1}(n-1)$ clusters in N', and thus also in N. This concludes the proof. □

The following theorem ensures that we can focus on decomposable level-k networks:

Theorem 1 [31] *Let $\mathcal{C}$ be a set of clusters. If there exists a level-k network representing $\mathcal{C}$, then there also exists such a network that is decomposable w.r.t. $\mathcal{C}$.*

We can now prove the main result of the section:

Theorem 2 *Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. Then, for every fixed $k \geq 0$, it is possible to determine in time $f(k) \cdot poly(n)$ whether a level-k network exists that represents $\mathcal{C}$, and if so to construct such a network.*

Proof By Theorem 1, we know that we can construct a decomposable level-k network using a divide-and-conquer strategy. A possible approach is described in Sect. 8.2 of [16]. This approach divides $\mathcal{C}$ in g subsets, where g is the number of connected components of the incompatibility graph. Then, each subset $\mathcal{C}_i$ is made separating w.r.t. $\mathcal{X}(\mathcal{C}_i)$ by merging every subset of $\mathcal{X}(\mathcal{C}_i)$ that is compatible with $\mathcal{C}_i$ (see [16] for more details). Then a *local* network is computed for each $\mathcal{C}_i$ and finally all the networks are merged together in a *global* level-k network representing $\mathcal{C}$. By construction [16], the number of biconnected components of the reconstructed network is equal to the number of connected components of $IG(\mathcal{C})$. Then, by Proposition 3 we can only have a $f(k) \cdot poly(n)$ number of clusters and a $poly(n)$ number of connected components in $IG(\mathcal{C})$, it is easy to see that the merging of the taxa in each $\mathcal{C}_i$ and the merging of all partial networks into the global one can be conducted in $f(k) \cdot poly(n)$ time. Moreover, since each subproblem $\mathcal{C}_i$ is separating, from Lemma 5 we have that constructing each local network takes $f(k) \cdot poly(n)$ time. This concludes the proof. □

4 Minimizing Reticulation Number is Fixed Parameter Tractable

The aim of this section is to show that reticulation number minimization is fixed parameter tractable. As pointed out for level minimization at the beginning of Sect. 3, it is sufficient to prove that, given a set of clusters $\mathcal{C}$ on taxon set $\mathcal{X}$, we can construct a phylogenetic network representing $\mathcal{C}$ with reticulation number r (if any exists) in time at most $f(r) \cdot poly(n)$.

To show the main result of this section we will introduce the concepts of *ST-collapsed cluster sets* and of *r-reticulation generators*. We will then prove that all the results and algorithms used in the previous section to prove that constructing *simple* level-k networks is fixed parameter tractable, hold not only for separating cluster sets and level-k generators but also for ST-collapsed cluster sets and r-reticulation generators. The main difficulty is to show that several key utility results still hold, since the other results do not exploit the biconnectedness of simple level-k generators. For ease of reading we will refer to the extended versions of these results using their original name followed by the term "(Extended)" (e.g. "Proposition 1" becomes "Proposition 1 (Extended)").

Observation 6 *Let N be a network on $\mathcal{X}$ with reticulation number r. Then N represents at most $2^{r+1}(n-1)$ clusters.*

Proof From Lemma 1 we may assume without loss of generality that N is binary. A binary network with reticulation number r contains exactly r reticulation nodes. Hence N displays at most 2^r trees, and each tree represents at most $2(n-1)$ clusters (because a rooted tree on n taxa contains at most $2(n-1)$ edges). □

We can thus henceforth assume that $|\mathcal{C}| \leq 2^{r+1}(n-1)$ i.e. that $\mathcal{C}$ contains at most $f(r) \cdot poly(n)$ clusters. Then we have that the following holds:

Proposition 1 (Extended) *Let N be a network on $\mathcal{X}$ with reticulation number at most r. Then, given a cluster set $\mathcal{C}$, we can check in time $f(r) \cdot poly(n)$ whether N represents $\mathcal{C}$.*

Given a set of taxa $S \subseteq \mathcal{X}$, we use $\mathcal{C} \setminus S$ to denote the result of removing all elements of S from each cluster in $\mathcal{C}$ and we use $\mathcal{C}|S$ to denote $\mathcal{C} \setminus (\mathcal{X} \setminus S)$ (i.e. the restriction of $\mathcal{C}$ to S). We say that a set $S \subseteq \mathcal{X}$ is an *ST-set* with respect to $\mathcal{C}$, if S is compatible with $\mathcal{C}$ and any two clusters $C_1, C_2 \in \mathcal{C}|S$ are compatible [20]. (We say that an ST-set S is *trivial* if $S = \emptyset$ or $S = \mathcal{X}$). An ST-set S is *maximal* if there is no ST-set S' with $S \subset S'$. The following results from [20] will be very useful:

Corollary 2 [20] *Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. Then there are at most n maximal ST-sets with respect to $\mathcal{C}$, they are uniquely defined and they partition $\mathcal{X}$.*

Lemma 6 [20] *The maximal ST-sets of a set of clusters $\mathcal{C}$ on $\mathcal{X}$ can be computed in polynomial time.*

Here "polynomial time" means $poly(n, |\mathcal{C}|)$, but given that the size of $\mathcal{C}$ is at most $f(r) \cdot poly(n)$ it follows that the maximal ST-sets of $\mathcal{C}$ can all be computed in time $f(r) \cdot poly(n)$.

The following corollary says, essentially, that if we want to construct networks with minimum reticulation number then it is safe to assume that each maximum ST-set corresponds to (the taxa in) a subtree that is attached to the main network via a cut-edge.

Corollary 3 [20] *Let N be a network that represents a set of clusters $\mathcal{C}$. There exists a network N' such that N' represents $\mathcal{C}$, $r(N') \leq r(N)$, $l(N') \leq l(N)$ and all maximal ST-sets (with respect to $\mathcal{C}$) are below cut-edges.*

Let $\mathcal{S} = \{S_1, \dots, S_m\}$ be the set of maximal ST-sets of $\mathcal{C}$. We construct a new cluster set $\mathcal{C}'$ from $\mathcal{C}$ as follows. For each $S_j \in \mathcal{S}$, and for each cluster C in $\mathcal{C}$ such that $S := S_j \cap C \neq \emptyset$, we replace the set S in C by the new taxon s_j. In other words we "collapse" all taxa in each maximal ST-set into a single new taxon that represents that ST-set. We say that $\mathcal{C}'$ is the *ST-collapsed* version of $\mathcal{C}$. We say that a cluster set is ST-collapsed if all its maximal ST-sets are singletons. Note that a separating cluster set $\mathcal{C}$ is necessarily ST-collapsed but the opposite implication does not hold. For example $\mathcal{C} = \{\{a, b\}, \{b, c\}, \{a, b, c, d\}, \{d, e\}\}$ on $\mathcal{X} = \{a, b, c, d, e\}$ is ST-collapsed but not separating because $\{a, b, c\}$ is compatible with $\mathcal{C}$.

Observation 5 (Extended) *Let $\mathcal{C}$ be a ST-collapsed cluster set on $\mathcal{X}$. Then every size-2 subset of $\mathcal{X}$ is incompatible with $\mathcal{C}$.*

Proof Suppose that this is not true and there exists a size-2 subset of $\mathcal{X}$, say A, that is compatible with $\mathcal{C}$. Since any two clusters $C_1, C_2 \in \mathcal{C}|A$ are necessarily compatible, A is a ST-set, contradicting the fact that $\mathcal{C}$ is ST-collapsed. □

Lemma 7 *Let $\mathcal{C}$ be a cluster set on $\mathcal{X}$, and let $\mathcal{C}'$ be the ST-collapsed version of $\mathcal{C}$. Then any network N' that represents $\mathcal{C}'$ can be transformed into a network N that represents $\mathcal{C}$ such that $r(N) = r(N')$ in $poly(|\mathcal{X}|)$ time.*

Proof Let $\mathcal{S} = \{S_1, \dots, S_m\}$ be the set of maximal ST-sets of $\mathcal{C}$. For each $S_j \in \mathcal{S}$ we replace the taxon s_j in N' with the tree on taxon set S_j that represents exactly the set of clusters $\mathcal{C}|S_j$. □

Corollary 4 *Let $\mathcal{C}$ be a cluster set on $\mathcal{X}$, and let $\mathcal{C}'$ be the ST-collapsed version of $\mathcal{C}$. Then $r(\mathcal{C}') = r(\mathcal{C})$.*

Proof Lemma 7 tells us that $r(\mathcal{C}) \leq r(\mathcal{C}')$. To see that $r(\mathcal{C}') \leq r(\mathcal{C})$, observe that Corollary 3 allows us to assume the existence of a network N with reticulation number $r(\mathcal{C})$ such that all the maximal ST-sets of $\mathcal{C}$ are below cut-edges in N. If, for each maximal ST-set S_j of $\mathcal{C}$, we replace the subtree corresponding to S_j with a single taxon s_j, we obtain a network with reticulation number at most $r(\mathcal{C})$ which represents $\mathcal{C}'$. □

Combining the fact that the transformation described in the proof of Lemma 7 can be executed in time $f(r) \cdot poly(n)$ with Lemma 7 and Corollary 4 we may thus henceforth restrict our attention to ST-collapsed cluster sets. Networks that represent ST-collapsed cluster sets have a rather restricted topology, as the following lemma shows.

Lemma 8 *Let $\mathcal{C}$ be an ST-collapsed cluster set on $\mathcal{X}$, and let N be a binary network that represents $\mathcal{C}$. Then it follows that, for each cut edge (u, v) of N, either v is a leaf labelled by a taxon from $\mathcal{X}$, or there is a directed path starting from v that can reach a reticulation node.*

Proof Let $\mathcal{X}(v) \subseteq \mathcal{X}$ be the set of taxa reachable from v by directed paths. If $|\mathcal{X}(v)| \geq 2$, but there are no reticulation nodes reachable by directed paths from v, then the subnetwork rooted at v is actually a tree with taxon set $\mathcal{X}(v)$, meaning that $\mathcal{X}(v)$ is an ST-set of cardinality 2 or higher. This violates the ST-collapsed assumption, giving a contradiction. If $|\mathcal{X}(v)| = 1$ then it follows that either v is a leaf labelled by a taxon, or (due to the fact that N is binary and contains no nodes with indegree and outdegree both equal to 1) at least one reticulation node is reachable from v by a directed path. □

We are now (finally) ready to define an r*-reticulation generator*. This is very closely related to the level-k generator discussed in Sect. 3. The only significant difference is that r-reticulation generators do not have to be biconnected, and (for technical reasons) the inclusion of a "fake root".

Definition 3 An r*-reticulation generator* is a directed acyclic multigraph, which has a single node of indegree 0, called the *fake root*, and this has outdegree 1; precisely r reticulation nodes (indegree 2 and outdegree at most 1), and apart from that only nodes of indegree 1 and outdegree 2.

Note that this definition implies that an r-reticulation generator cannot contain any leaf. As in the case of level-k generators, nodes with indegree 2 and outdegree 0 as well as all edges are called *sides*. Figure 6 shows the single 1-reticulation generator and the seven 2-reticulation generators.

Lemma 9 *There are at most $f(r)$ r-reticulation generators and each r-reticulation generator contains at most $f(r)$ sides.*

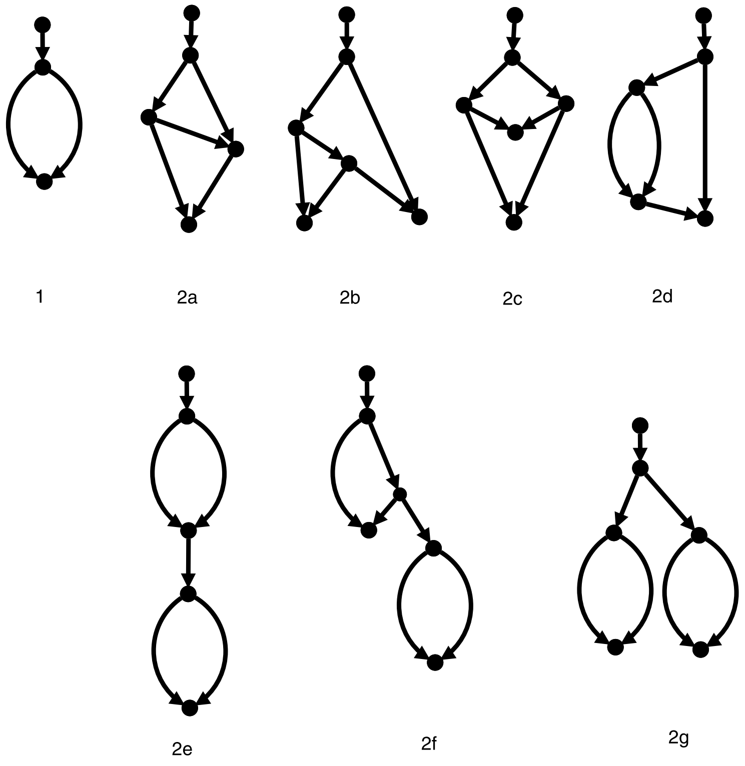

Fig. 6 The single 1-reticulation generator and the seven 2-reticulation generators

Proof In Lemma 1 of [30] it is proven that a level-k generator has at most $3k-1$ vertices and at most $4k-2$ edges. The proof there does not exploit the biconnectedness of level-k generators, so—with the exception of the fake root—also holds for r-reticulation generators. By adding 1 to both the vertex and edge upper bounds to account for the fake root we come to upper bounds of $3r$ and $4r-1$ respectively. Since an r-reticulation generator contains r reticulations, we have at most $4r-1$ edge sides and r node sides, so at most $5r-1$ sides. To see that there are at most $f(r)$ r-reticulation generators observe that between any pair of nodes u and v in the generator there is either no edge, an edge from u to v (or from v to u), or a multi-edge from u to v (or from v to u). Hence there are at most $5^{(3r)^2}$ r-reticulation generators. $\square$

Definition 4 The set $\hat{\mathcal{N}}^r$ (for $r \geq 1$) is defined as the set of all binary networks that can be constructed by choosing some r-reticulation generator G, then applying the leaf hanging transformation described in Definition 1 and finally deleting the fake root (i.e. the single vertex with indegree 0 and outdegree 1) and its incident edge.

Lemma 10 (Extending Corollary 1) *Let $\mathcal{C}$ be an ST-collapsed set of clusters on $\mathcal{X}$, such that $r(\mathcal{C}) \geq 1$. Then there exists a network N in $\hat{\mathcal{N}}^{r(\mathcal{C})}$ such that N represents $\mathcal{C}$.*

Proof Let N be any binary network with reticulation number $r(\mathcal{C})$ such that N represents $\mathcal{C}$. We show how applying the reverse of the transformation described in Definition 4 to N will give some $r(\mathcal{C})$-reticulation generator G. The lemma will then follow. We begin by adding a fake root to N i.e. a new vertex u' and an edge from u' to the root of N. (This is the inverse of deleting the fake root). We then delete all the leaves in N. Any nodes that are created with indegree 2 and outdegree 0 we leave as they are (this is the inverse of step 3 of Definition 1). Nodes with indegree 1 and outdegree 0 cannot be created, because this would require that there exists a node v in N which has indegree 1 and outdegree 2 such that both its children are leaves labelled by taxa. But this would mean that v is the head of a cut-edge e where e violates the condition described in Lemma 8. Now, consider the nodes that have been created with indegree and outdegree both equal to 1. Let u be any such node, and let $U = \{u\}$. Whenever U contains a node u whose unique parent $p(u)$ also has indegree and outdegree both equal to 1, add $p(u)$ to U. Whenever U contains a node u whose unique child $c(u)$ also has indegree and outdegree both equal to 1, add $c(u)$ to U. We continue expanding U this way until it cannot grow anymore. Clearly U stops growing at the point that U contains two nodes u_{top} and u_{bottom} (where possibly $u_{top} = u_{bottom}$) such that the parent of u_{top} (respectively, child of u_{bottom}) does not have indegree and outdegree both equal to 1. We suppress all the nodes in U, in the usual sense. Note, crucially, that this does not affect the indegree or outdegree of the parent of u_{top} or the child of u_{bottom}. While N still contains nodes of indegree 1 and outdegree 1 we repeat the above process, until none are left; this is the inverse of steps 1 and 2 of Definition 1. (Note that this process might create multi-edges, but because it leaves the indegree and outdegree of unsuppressed nodes intact, there will be at most two edges between any two nodes). Now, let G be the resulting structure. Observe that the reticulation number of G is the same as N, that every node in G with indegree 2 has outdegree 0 or 1, that G contains a single fake root, that all nodes in G with indegree 1 have outdegree 2, and that G contains no leaves. We conclude that G is an r-reticulation generator and that we could have constructed N by applying the transformation described in Definition 4 to G. □

Proposition 2 (Extended) *Given a ST-collapsed cluster set $\mathcal{C}$ on $\mathcal{X}$ and an ordered set of distinct taxa of $\mathcal{X}$ $(x_1, \dots, x_j)$ such that $j \geq 2$ and $x_i \rightarrow_{\mathcal{C}} x_{i+1}$ for $1 \leq i \leq (j-1)$. Then $x_j \not\rightarrow_{\mathcal{C}} x_1$.*

Proof If $x_i \rightarrow_{\mathcal{C}} x_{i+1}$ for $1 \leq i \leq (j-1)$ and $x_j \rightarrow_{\mathcal{C}} x_1$, this means that the set $\mathcal{X}' = \bigcup_{i=1}^{j} x_i$ is compatible with $\mathcal{C}$. Moreover, every non-singleton cluster that contains one element of $\mathcal{X}'$, contains them all. So every non-singleton cluster $C \in \mathcal{C}$ is either disjoint from $\mathcal{X}'$, or contains it, from which we conclude that any two clusters $C_1, C_2 \in \mathcal{C}|\mathcal{X}'$ are compatible. So $\mathcal{X}'$ is a ST-set and we have a contradiction. □

Since the number of r-reticulation generators and the number of sides in a generator is bounded by $f(r)$ from Lemma 9, and we have extended several critical utility results to also apply to ST-collapsed cluster sets and r-reticulation generators, we have the following:

Lemma 7 (Extended) *Let $\mathcal{C}$ be a ST-collapsed set of clusters on $\mathcal{X}$. Then, for every fixed $r \geq 0$, Algorithm 3 (Extended) determines whether a network that represents $\mathcal{C}$ with reticulation number equal to r exists, and if so, constructs such a network in time $f(r) \cdot poly(n)$.*

Here Algorithm 3 (Extended) coincides with Algorithm 3 but for the fact that in the former we loop through all r-reticulation generators instead of through all level-k generators.

From Lemma 7 and Corollary 4, we may finally conclude the following.

Theorem 3 *Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. Then, for every fixed $r \geq 0$, it is possible to determine in time $f(r) \cdot poly(n)$ whether a network that represents $\mathcal{C}$ with reticulation number at most r exists.*

5 Conclusions and Open Problems

In this article we have shown that, under the softwired cluster model of phylogenetic networks, constructing networks with minimum reticulation number (respectively, level) is fixed parameter tractable where the reticulation number (respectively, level) is the parameter. The obvious problem with the algorithms in this article is that the part of the running time that depends only on the parameter is massively exponential. This contrasts with fixed parameter tractable algorithms for combining *two binary trees* into a phylogenetic network. In that literature the dependence on the parameter is more modest, the state- of-the-art being $O(3.18^r \cdot n)$ [33] (where $n = |\mathcal{X}|$ and r is the hybridization number of the two trees).

However, the two binary tree case is rather special [20, 28] and does not obviously generalize to more complex inputs such as arbitrary sets of clusters or arbitrarily large sets of potentially nonbinary trees. Indeed, there is still no fixed parameter tractable algorithm for combining an arbitrary set of nonbinary trees into a phylogenetic network using a minimum number of reticulations. Could the ideas presented in this article—in particular, the use of generators—offer a theoretical route to this result?

References

1. Bordewich, M., Semple, C.: Computing the hybridization number of two phylogenetic trees is fixed-parameter tractable. IEEE/ACM Trans. Comput. Biol. Bioinf. **4**(3), 458–466 (2007)
2. Bordewich, M., Semple, C.: Computing the minimum number of hybridization events for a consistent evolutionary history. Discrete Appl. Math. **155**(8), 914–928 (2007)
3. Bordewich, M., Linz, S., John, K.St., Semple, C.: A reduction algorithm for computing the hybridization number of two trees. Evol. Bioinform. **3**, 86–98 (2007)
4. Chen, Z.-Z., Wang, L.: Algorithms for reticulate networks of multiple phylogenetic trees. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**, 372–384 (2012)
5. Collins, J., Linz, S., Semple, C.: Quantifying hybridization in realistic time. J. Comput. Biol. **18**(10), 1305–1318 (2011)

6. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (1999)
7. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
8. Gambette, P., Berry, V., Paul, C.: The structure of level-k phylogenetic networks. In: Proceedings of the 20th Annual Symposium on Combinatorial Pattern Matching, CPM '09, pp. 289–300. Springer, Berlin (2009)
9. Gascuel, O. (ed.): Mathematics of Evolution and Phylogeny. Oxford University Press, Oxford (2005)
10. Gascuel, O., Steel, M. (eds.): Reconstructing Evolution: New Mathematical and Computational Advances. Oxford University Press, Oxford (2007)
11. Gramm, J., Nickelsen, A., Tantau, T.: Fixed-parameter algorithms in phylogenetics. Comput. J. **51**(1), 79–101 (2008)
12. Gusfield, D., Bansal, V., Bafna, V., Song, Y.: A decomposition theory for phylogenetic networks and incompatible characters. J. Comput. Biol. **14**(10), 1247–1272 (2007)
13. Gusfield, D., Hickerson, D., Eddhu, S.: An efficiently computed lower bound on the number of recombinations in phylognetic networks: theory and empirical study. Discrete Appl. Math. **155**(6–7), 806–830 (2007)
14. Huson, D.H., Scornavacca, C.: A survey of combinatorial methods for phylogenetic networks. Genome Biol. Evol. **3**, 23–35 (2011)
15. Huson, D.H., Rupp, R., Berry, V., Gambette, P., Paul, C.: Computing galled networks from real data. Bioinformatics **25**(12), i85–i93 (2009)
16. Huson, D.H., Rupp, R., Scornavacca, C.: Phylogenetic Networks: Concepts, Algorithms and Applications. Cambridge University Press, Cambridge (2011)
17. Huynh, T.N.D., Jansson, J., Nguyen, N.B., Sung, W.-K.: Constructing a smallest refining galled phylogenetic network. In: Research in Computational Molecular Biology (RECOMB). Lecture Notes in Bioinformatics, vol. 3500, pp. 265–280 (2005)
18. Jansson, J., Sung, W.-K.: Inferring a level-1 phylogenetic network from a dense set of rooted triplets. Theor. Comput. Sci. **363**(1), 60–68 (2006)
19. Jansson, J., Nguyen, N.B., Sung, W.-K.: Algorithms for combining rooted triplets into a galled phylogenetic network. SIAM J. Comput. **35**(5), 1098–1121 (2006)
20. Kelk, S., Scornavacca, C., van Iersel, L.: On the elusiveness of clusters. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**, 517–534 (2012)
21. Myers, S.R., Griffiths, R.C.: Bounds on the minimum number of recombination events in a sample history. Genetics **163**, 375–394 (2003)
22. Nakhleh, L.: Evolutionary phylogenetic networks: models and issues. In: The Problem Solving Handbook for Computational Biology and Bioinformatics. Springer, Berlin (2009)
23. Niedermeier, R.: Invitation to Fixed Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford (2006)
24. Semple, C.: Hybridization networks. In: Reconstructing Evolution—New Mathematical and Computational Advances. Oxford University Press, Oxford (2007)
25. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
26. To, T.-H., Habib, M.: Level-*k* phylogenetic networks are constructable from a dense triplet set in polynomial time. In: CPM09. LNCS, vol. 5577, pp. 275–288 (2009)
27. van Iersel, L., Kelk, S.: Constructing the simplest possible phylogenetic network from triplets. Algorithmica **60**, 207–235 (2011)
28. van Iersel, L.J.J., Kelk, S.M.: When two trees go to war. J. Theor. Biol. **269**(1), 245–255 (2011)
29. van Iersel, L.J.J., Keijsper, J.C.M., Kelk, S.M., Stougie, L., Hagen, F., Boekhout, T.: Constructing level-2 phylogenetic networks from triplets. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(4), 667–681 (2009)
30. van Iersel, L.J.J., Kelk, S.M., Mnich, M.: Uniqueness, intractability and exact algorithms: reflections on level-*k* phylogenetic networks. J. Bioinform. Comput. Biol. **7**(2), 597–623 (2009)
31. van Iersel, L.J.J., Kelk, S.M., Rupp, R., Huson, D.H.: Phylogenetic networks do not need to be complex: Using fewer reticulations to represent conflicting clusters. Bioinformatics **26**, i124–i131 (2010). Special issue: Proceedings of Intelligent Systems for Molecular Biology 2010 (ISMB2010), 10th–13th September (2010)
32. Whidden, C., Zeh, N.: A unifying view on approximation and fpt of agreement forests. In: Salzberg, S., Warnow, T. (eds.) Algorithms in Bioinformatics. Lecture Notes in Computer Science, vol. 5724, pp. 390–402. Springer, Berlin (2009)

33. Whidden, C., Beiko, R.G., Zeh, N.: Fixed-parameter and approximation algorithms for maximum agreement forests. arXiv:1108.2664v1 [q-bio.PE]
34. Wu, Y.: Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees. Bioinformatics **26**, i140–i148 (2010). Special issue: Proceedings of Intelligent Systems for Molecular Biology 2010 (ISMB2010), 10th–13th September (2010)
35. Wu, Y., Gusfield, D.: A new recombination lower bound and the minimum perfect phylogenetic forest problem. J. Comb. Optim. **16**(3), 229–247 (2008)