

# A Row-wise Algorithm for Graph Realization

Rolf van der Hulst , Matthias Walter 

University of Twente, Enschede, The Netherlands  
 r.p.vanderhulst@utwente.nl, m.walter@utwente.nl

August 22, 2024

## Abstract

Given a  $\{0, 1\}$ -matrix  $M$ , the graph realization problem for  $M$  asks if there exists a spanning forest such that the columns of  $M$  are incidence vectors of paths in the forest. The problem is closely related to the recognition of network matrices, which are a large subclass of totally unimodular matrices and have many applications in mixed-integer programming. Previously, Bixby and Wagner have designed an efficient algorithm for graph realization that grows a submatrix in a column-wise fashion whilst maintaining a graphic realization. This paper complements their work by providing an algorithm that works in a row-wise fashion and uses similar data structures. The main challenge in designing efficient algorithms for the graph realization problem is ambiguity as there may exist many graphs realizing  $M$ . The key insight for designing an efficient row-wise algorithm is that a graphic matrix is uniquely represented by an SPQR tree, a graph decomposition that stores all graphs with the same set of cycles. The developed row-wise algorithm uses data structures that are compatible with the column-wise algorithm and can be combined with the latter to detect maximal graphic submatrices.

**Keywords:** Graph realization, graphic matrix, graphic matroid, network matrix, recognition algorithm, SPQR tree

**Acknowledgments:** The authors acknowledge funding support from the Dutch Research Council (NWO) on grant number OCENW.M20.151.

## 1 Introduction

Graphs are important objects in mathematics, and occur in many different fields and practical applications. Representing graphs using matrices, such as the adjacency matrix or the node-edge incidence matrix, can provide powerful insights into the structure and facilitate solving problems involving the graph. In this work, we consider the *representation matrix* of a graph, defined as follows.

Given a connected multigraph  $G$  with vertex set  $V(G)$ , edge set  $E(G)$  and a spanning tree  $T \subseteq E(G)$  of  $G$ , the *fundamental path*  $P_e(T) := P_{u,w}(T) \subseteq T$  of an edge  $e = \{u, w\} \in E \setminus T$  is defined as the unique path in  $T$  connecting the two end-vertices of  $e$ . For such a *graph-tree pair*  $(G, T)$ , let  $M(G, T)$  be a binary  $|T| \times |E \setminus T|$  matrix with rows indexed by edges in  $T$  and columns indexed by edges in  $E \setminus T$ . For a pair of edges  $(e, f) \in T \times (E \setminus T)$  we let  $M_{e,f} = 1$  if  $e \in P_f(T)$  and  $M_{e,f} = 0$  otherwise. The matrix  $M(G, T)$  constructed in this fashion is called the *representation matrix* of  $(G, T)$ . Figure 1 shows an example of a graph-tree pair and its representation matrix.

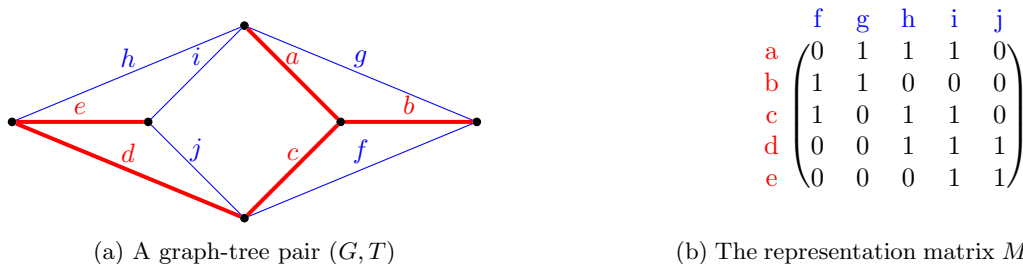


Figure 1: A graph-tree pair  $(G, T)$  and its representation matrix. Edges in  $T$  are marked bold and red, edges in  $E \setminus T$  are marked blue.

Our central problem, the *graph realization problem* asks for the reverse operation:

**Problem 1.** Given a binary matrix  $B$ , is there a multigraph  $G$  with spanning tree  $T \subseteq E(G)$  such that  $B = M(G, T)$  holds?

In the affirmative case,  $B$  is said to be *graphic*, and  $(G, T)$  is said to *realize*  $B$ .

**Related problems.** Our main motivation for investigating Problem 1 comes from *network matrices*, which are closely related to graphic matrices. Given a directed multigraph  $G = (V, A)$ , with some (not necessarily rooted) spanning forest  $T$ , the network matrix  $N(G, T)$  is a  $|T| \times |A \setminus T|$  matrix where for edge pair  $(e, f) \in T \times (A \setminus T)$  satisfies

$$N(G, T)_{e,f} = \begin{cases} +1 & \text{if } e \text{ occurs in } P_f(T) \text{ forwardly,} \\ -1 & \text{if } f \text{ occurs in } P_f(T) \text{ backwardly,} \\ 0 & \text{otherwise.} \end{cases}$$

The graph realization problem is closely connected to the problem of determining whether a given  $\{-1, 0, 1\}$ -matrix  $B$  is a network matrix. After solving the graph realization problem on the binary support of  $B$ , one only needs to determine the direction of each arc in order to determine whether the matrix is a network matrix. Finding the arc directions is relatively straightforward to do, as first outlined by Camion [7]. Bixby and Cunningham [5] go into more detail (see Algorithm 7 of [5]).

Network matrices are a large class of *totally unimodular* matrices [36]. This makes them of practical interest for (mixed)-integer linear optimization, because problems with a totally unimodular constraint matrix and integer right hand side can be solved in polynomial time using linear optimization [18] with, say, the Ellipsoid method [15, 23, 24] or interior-point methods [22]. Even if only part of the constraint matrix of an integer linear optimization program is a network matrix, a large network *submatrix* can be useful to reduce the solution time. For instance, branch-and-cut algorithms can use stronger cutting planes that exploit network-design substructures [1]. In recent work, Aprile et al. consider integer programs with  $\Delta$ -modular constraint matrices that contain a large transposed network matrix and only a constant number of rows that do not belong to the transposed network matrix, and show that these integer programs are solvable in time polynomial in  $\Delta$  and the size of the constraint matrix [2].

Additionally, the detection of total unimodularity requires the solution of graph realization problems [26, 32]. Although detecting whether a matrix is a network matrix can be done in polynomial time [34], both the problem of finding the largest graphic submatrix (using various definitions of ‘largest’) and the problem of finding the largest network submatrix are NP-hard [3].

The graph realization problem is closely connected to graphic matroids, and can be reformulated in terms of matroids. A binary matrix  $B$  defines a *linear matroid* that has the columns of  $[I \mid B]$  as a ground set, where subsets are independent if and only if the corresponding column vectors are linearly independent over  $\mathbb{F}_2$ . Every graph  $G = (V, E)$  also has an associated graphic matroid, which has the set  $E$  of edges as its ground set and the forests of  $G$  as its independent sets. The graph realization problem can then be reformulated as follows: given a binary matrix  $B$ , is the linear matroid given by  $B$  isomorphic to a graphic matroid? Although we will not use the matroid perspective throughout this work, it may be useful for readers that are familiar with matroid theory.

**Known methods.** Numerous methods have been proposed to solve the graph realization problem. Tutte gave a first polynomial-time algorithm [34, 35]. Many other polynomial-time algorithms were later developed by a large variety of authors [5, 6, 8, 13, 14, 20, 31], and the books [25, 33] explain some of these in further detail. The most impressive results for graph realization were obtained by Bixby and Wagner [6] and Fujishige [13]. Let  $k$  be the number of nonzeros of the input matrix  $B \in \{0, 1\}^{m \times n}$ . Both papers achieve an ‘almost linear’ running time of  $\mathcal{O}(k\alpha(k, m))$ , where  $\alpha$  denotes the extremely slowly growing *inverse Ackermann function* [30].

The algorithms described in both papers work in a similar column-wise fashion, namely by growing a graphic submatrix by one column in each step. First, they determine some initial graphic matrix  $M$  given by a subset of columns of  $B$ , typically by a single column. If  $M$  has a single column,  $M$  represents a graph that is a cycle. Then they augment  $M$  with a new column  $c$  and (efficiently) determine if the matrix  $[M \mid c]$  is graphic. In the affirmative case, they set  $M := [M \mid c]$  and repeat the augmentation, terminating only when  $M = B$  or when  $[M \mid c]$  is found to be non-graphic.

It is well known that there may be many graph-tree pairs  $(G, T)$  sharing the same representation matrix  $M(G, T)$  [39]. This ambiguity is one of the main challenges in designing algorithms for Problem 1. The algorithms due to Bixby and Wagner and due to Fujishige both maintain complicated data structures in order to represent the graphic matrix  $M$ , and efficiently check if the new column  $c$  can be augmented whilst preserving graphicness. Bixby and Wagner use a so-called *t-decomposition*, whereas Fujishige uses the more complicated *PQ-trees*. In this work, we will use *SPQR trees*, which are very similar to the *t-decomposition* used by Bixby and Wagner.

**Research gap.** As there is an interest in determining large graphic submatrices, we observe a gap in previous research. Although the existing algorithms are very efficient, there is no algorithm that grows a graphic (sub)matrix in a *row-wise* fashion. The lack of such an algorithm can be explained by its difficulty.

In the column augmentation algorithms the new column is added by reversing the deletion of an edge in a represented graph, which is equivalent to the addition of the column edge. Augmenting a row amounts to reversing the contraction of an edge in the graph, which is a more complicated operation than addition. However, a row-wise algorithm could be useful since for all fast algorithms known from the literature the found submatrix always contains all rows. This limits the set of graphic submatrices that can be efficiently found by existing algorithms. We would like to be able to determine graphicness of arbitrary submatrices, having the application of finding network submatrices in mixed-integer linear optimization in mind. Consequently, we are interested in solving the *graphic row augmentation problem*:

**Problem 2.** *Given a graphic matrix  $M$  and a binary vector  $b$ , determine graphicness of the matrix*

$$M' := \begin{bmatrix} M \\ b^\top \end{bmatrix}. \quad (1)$$

By repeatedly solving Problem 2, we can determine if a matrix is graphic. Note that similarly to the column case, any binary matrix with a single row is graphic. It is realized by a graph  $G$  that has two vertices that are connected by a set of parallel edges. Moreover, if the augmentation of a row  $b^\top$  does not preserve graphicness, one could also continue and ignore that row, which leads to an algorithm for greedily building an inclusion-wise maximal graphic submatrix of  $B$ .

Given an efficient algorithm to solve Problem 2, we could additionally combine row and column augmentation to efficiently determine graphicness of arbitrary graphic submatrices, and not just of all-row or all-column submatrices. In order to facilitate the practical implementation of such an algorithm, we use data structures that are highly similar to the ones used by Bixby and Wagner [6].

**Outline.** In Section 2 we show how to deal with a block structure of  $M$  and reduce the graphic row augmentation problem to the case in which  $M$  is connected. Section 3 characterizes the graphic row augmentation problem in terms of graphs. In Section 4 we introduce the SPQR tree data structure and show how they encode graphic matrices. Section 5 relates the previous two sections and shows how to update SPQR trees to reflect a graphic row augmentation. In Section 6 we present and discuss the complete algorithm and provide worst-case bounds for space and time complexity.

## 2 Separations and connectivity

It is well-known that there can be multiple graphs  $G$  and trees  $T$  with the same representation matrix  $M(G, T)$  [39]. In other words, the graph that is represented by a certain matrix is not unique. To formalize the corresponding ambiguity, let us define a  $k$ -separation as a partition of  $E(G)$  into  $E_1$  and  $E_2$  such that  $|E_1|, |E_2| \geq k$  holds and such that the *corresponding graphs*  $G_1$  and  $G_2$  have exactly  $k$  *separating vertices* in common, where  $G_i$  is the graph with edge set  $E_i$  and vertex set  $V_i := \bigcup_{e \in E_i} e$ . A graph is called  $k$ -connected if it is connected and it has no  $\ell$ -separation for all  $\ell \in \{1, 2, \dots, k-1\}$ . Our definition of  $k$ -connected is equivalent to the one used by Tutte [37]. Tutte  $k$ -connectivity implies the more commonly used  $k$ -vertex connectivity. A vertex that is the common to two parts of a 1-separation is called an *articulation vertex*.

Consider a multigraph  $G = (V, E)$ . For a vertex  $v \in V$ , we use  $\delta(v)$  to denote the edges incident to  $v$ . Occasionally, we will write  $\delta_G(v)$  to clarify in which graph we are considering the neighborhood of  $v$ . For a subset of edges  $F \subseteq E$ , we use the notation  $G \setminus F$  to denote the graph  $G = (V, E \setminus F)$  where the edges in  $F$  are removed. For a set of edges  $F \subseteq E$ , we use  $G/F$  to denote the graph obtained by contracting each edge  $e \in F$  into a single vertex. For a subset of vertices  $U \subseteq V$ , we use  $G - U$  to denote the induced subgraph  $G[V \setminus U]$ , where the vertices  $U$  and all incident edges have been removed. For a singular vertex  $v \in V$ , we occasionally abuse notation and write  $G - v$ , instead.

Although Problem 2 asks for the addition of a new row, the reverse operations of deleting a row (or a column) from a graphic matrix  $M = M(G, T)$  give us important intuition. In particular, if  $M'$  is the matrix obtained by deleting a column  $c$  from  $M$ , this corresponds to the deletion of the corresponding edge in  $E \setminus T$ , and for  $G' = (V, E \setminus \{c\})$  it holds that  $M' = M(G', T)$ . If  $M'$  is the matrix obtained by deleting a row  $r$  from  $M$ , then it can be obtained by contracting the corresponding tree edge, and we observe that  $M' = M(G/\{r\}, T/\{r\})$  holds. Since graphicness is maintained under the deletion of rows and columns, it follows that graphicness is maintained under taking submatrices. For more details, we refer to [33, Chapter 3].

We start with a reduction to the case in which we can assume that  $M$  in (1) is *connected*, meaning that the graph with the adjacency matrix  $\begin{bmatrix} 0 & M \\ M^\top & 0 \end{bmatrix}$  is connected. If our input matrix  $A$  is connected, one can, via a breadth-first search, reorder the rows in such a way that  $M$  remains connected in consecutive updates. However, this sequential connectivity property is lost if we may later skip rows whose addition does not preserve graphicness. Consequently, we show how to actually treat matrices  $M$  that are not connected.

To this end, we consider the case in which the matrix  $M$  consists of  $k$  block submatrices  $M_1, M_2, \dots, M_k$  each of which being connected, and that  $b^\top = (b_1^\top, b_2^\top, \dots, b_{k-1}^\top, b_k^\top)$  is the corresponding partition of the

new row  $b$ :

$$M' = \begin{bmatrix} M_1 & 0 & \dots & 0 & 0 \\ 0 & M_2 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & M_{k-1} & 0 \\ 0 & 0 & \dots & 0 & M_k \\ b_1^\top & b_2^\top & \dots & b_{k-1}^\top & b_k^\top \end{bmatrix}, \quad b_i \neq \mathbb{0} \quad (2a)$$

$$M'_i := \begin{bmatrix} M_i \\ b_i^\top \end{bmatrix} \quad (2b)$$

Note that we require each  $b_i$  containing at least one 1-entry. Moreover, we allow that any such submatrix  $M_i$  has no rows, in which case it must have a single column in order to be connected. The following theorem ensures that we can assume connectivity in subsequent sections.

**Theorem 3.** *Let  $M \in \{0, 1\}^{m \times n}$  be a graphic matrix and let  $b \in \{0, 1\}^n$  be such that the matrix  $M'$  is of the form (2a). Then  $M'$  is graphic if and only if all matrices  $M'_i$  as in (2b) are graphic for  $i = 1, 2, \dots, k$ . Moreover, every pair  $(G', T')$  with  $M' = M(G', T')$  can be obtained as follows. For  $i = 1, 2, \dots, k$ , let  $(G'_i, T'_i)$  be such that  $M'_i = M(G'_i, T'_i)$  holds and let  $e'_i \in T'_i$  be the edge corresponding to the last row of  $M'_i$ . Let  $G'_0$  be a graph with two vertices and exactly  $k + 1$  edges  $e_0, e_1, \dots, e_k$ . Obtain  $G'$  (resp.  $T'$ ) from  $G'_0, G'_1, \dots, G'_k$  (resp. from  $G'_0, T'_1, \dots, T'_k$ ) by identifying  $e_i$  with  $e'_i$  for every  $i = 1, 2, \dots, k$ , and then removing these  $2k$  edges. The edge  $e_0$  then corresponds to the last row of  $M'$ .*

*Proof.* Since graphicness is maintained under taking submatrices, graphicness of  $M'$  implies graphicness of  $M'_i$  for all  $i = 1, 2, \dots, k$ . In order to prove the reverse direction of the first statement, assume  $M'_i = M(G'_i, T'_i)$  for  $i = 1, 2, \dots, k$ . Let  $G'_0, G', T'$  as well as all  $e'_i$  and  $e_i$  be as in the theorem. By construction,  $T'$  consists of the union of the sets  $T'_i \setminus \{e'_i\}$ , augmented by  $e_0$ . We now show  $M(G', T') = M'$ . Consider any column of  $M'$ , which is also a column of  $M'_i$  for some  $i \in \{1, 2, \dots, k\}$  and let  $e \in E(G'_i) \setminus T'_i$  be the corresponding edge. By construction of  $G'$  and  $T'$ , the fundamental path  $P_e(T')$  contains the same edges as  $P_e(T'_i)$ , except that  $e_0$  in  $T'$  is exchanged for  $e'_i \in P_e(T')$ . Since the row vector of  $M'_i$  corresponding to  $e'_i$  and the row vector of  $M'$  corresponding to  $e_0$  are both  $b_i^\top$ , this shows that the column vectors of  $M'$  and of  $M(G', T')$  are identical. We conclude that  $M'$  is indeed graphic and that  $M' = M(G', T')$  holds.

It remains to show that all realizations of  $M'$  are of that type. To this end, consider any  $(G^*, T^*)$  with  $M' = M(G^*, T^*)$ . Let  $e^* \in T^*$  denote the edge corresponding to the last row of  $M'$ . Let, for every  $i \in \{1, 2, \dots, k\}$ ,  $G'_i$  and  $T'_i$  be obtained from  $G^*$  and  $T^*$ , respectively, by contracting all edges of  $T^*$  corresponding to rows that do not belong to  $M'_i$  and (for  $G'_i$ ) deleting all edges of  $E(G^*) \setminus T^*$  corresponding to columns that do not belong to  $M'_i$ . By construction we have  $M(G'_i, T'_i) = M'_i$ . The edge  $e^*$  remains present in each of the constructed graphs, and we denote this edge by  $e'_i$  in the corresponding graph and tree. Let  $G_i := G'_i / e'_i$  and  $T_i := T'_i / e'_i$ , respectively. Clearly, the edge sets of the  $G_i$  form a partition of  $E(G^*) \setminus \{e^*\}$ . Hence,  $G^*$  must be so that the contraction of  $e^*$  yields a graph that consists of the  $G_i$ , all having a single vertex  $v^* \in V(G_1) \cap V(G_2) \cap \dots \cap V(G_k)$  in common. Moreover, since each  $b_i$  contains at least one 1-entry, every  $G_i$  has some edge  $f$  such that  $e^* \in P_f(T^*)$  holds. Hence, the joint vertex  $v^*$  of the parts of  $G^* / e^*$  must be the one to which  $e^*$  was contracted. This shows that  $(G^*, T^*)$  is equal to the pair  $(G', T')$  (with respect to the  $(G'_i, T'_i)$  and  $e'_i$  constructed above) as stated in the theorem.  $\square$

In our introduction, we assumed that  $G$  was a connected multigraph. For any disconnected graph  $G$  and any spanning forest  $T$ ,  $M(G, T)$  has two or more disconnected blocks. As we can process each block individually, it is clear that the assumption that  $G$  is connected was not necessary and can be easily dealt with. Moreover, from now on we can assume that the current matrix  $M$  is connected. As a consequence, we can focus on 2-connected graphs due to the following result.

**Proposition 4** (Truemper [33, Prop. 3.2.31]). *Let  $G$  be a connected graph such that  $M = M(G, T)$  holds. Then  $G$  is 2-connected if and only if  $M$  is connected.*

### 3 Graphic row augmentation

In the previous section, we observed that the graphic row augmentation problem reduces to the case in which  $M' = \begin{bmatrix} M \\ b^\top \end{bmatrix}$  holds, where  $M$  is connected. In this section, we do not explicitly use the connectivity property of  $M$ , but derive more general results for when  $M$  is graphic. We denote by  $Y := \text{supp}(b)$  the subset of columns with a 1-entry in  $b^\top$ . To gain some intuition, let us assume that we are given some graph  $G$  with tree  $T$  such that  $M = M(G, T)$ . We then want to find operations on  $G$  that add a new edge  $r^*$  (indexing  $b^\top$ ) such that the paths  $P_y(T) \in Y$  are elongated with  $r^*$  and the paths  $P_c(T)$  are unaltered for all columns  $c$  of  $M$  where  $b_c = 0$ .

Suppose that  $M'$  is indeed graphic such that  $M' = M(G', T')$  for some graph  $G'$  with tree  $T'$ . If we were to remove the last row  $r^*$  of  $M'$ , we would obtain  $M$  back again. Now, we consider what happens to  $G'$  and  $T'$  in this case. The removal of the row  $r^*$  corresponds to a contraction of the edge corresponding to

$r^*$  in  $G'$ . Clearly, such a contraction shortens precisely those fundamental paths of the edges  $e \in E \setminus T$  for which  $r^* \in P_e(T)$  holds. Thus, after performing the contraction, we find a graph  $G$  with tree  $T = T' \setminus \{r^*\}$  such that  $M = M(G, T)$  holds.

An algorithm that solves Problem 2 would need to find the ‘reverse’ of such a contraction. Hence, we can intuitively determine that we need to take two steps. First, we identify a vertex  $v \in V(G)$  that is to be split into vertices  $v_1$  and  $v_2$ . Second, we check if there exists a bipartition of the edges in  $\delta(v)$ , so that we can reassign the edges in  $\delta(v)$  to either  $\delta(v_1)$  or  $\delta(v_2)$ . In particular, this bipartition should ensure that the fundamental paths of the columns  $Y$  are elongated and the fundamental paths of the columns  $C \setminus Y$  remain unchanged.

If there is such a bipartition, we can split  $v$  into two new vertices  $v_1$  and  $v_2$ , reassign the edges  $\delta(v)$  to  $\delta(v_1)$  and  $\delta(v_2)$ , and add the new edge  $r^*$  between  $v_1$  and  $v_2$ . This way, we create a new graph-tree pair  $(G', T')$  with  $T' = T \cup \{r^*\}$  such that  $M' = M(G', T')$ .

We will now formalize the above intuition. The following definitions were inspired by Truemper’s work on recognizing total unimodularity, which contains an algorithm for solving the problem in case the involved graphs are 3-connected (see case 2 of the TEST-C subroutine in [32]). However, Truemper’s algorithm lacks a detailed explanation and proofs. In fact, we identified a minor mistake and later provide a fix.

**Definition 5** (*Y-reduced graph, auxiliary graph, Y-splittable vertices*). *Let  $G$  be a multigraph and let  $Y \subseteq E(G)$  be a subset of its edges. We define the Y-reduced graph of  $G$  as the graph  $G_Y := (V(G), E(G) \setminus Y)$ . Moreover, by  $G_Y^v := G_Y[V \setminus \{v\}]$  we denote the graph obtained by removing edges  $Y$  and vertex  $v$  (along with its incident edges). The corresponding auxiliary graph  $H_Y^v := H_Y^v(G)$  is the graph having a vertex for each connected component of  $G_Y^v$  and with edges  $\{h_1, h_2\} \in E(H_Y^v)$  if and only if there is an edge  $\{v_1, v_2\} \in Y$  with  $v_1 \in V(h_1)$  and  $v_2 \in V(h_2)$ . Finally, we say that a vertex  $v \in G$  is Y-splittable with respect to  $G$  if  $H_Y^v$  is bipartite.*

Note that the auxiliary graph  $H_Y^v$  is not bipartite if it contains loops, i.e., if an edge from  $Y$  has both end-vertices in the same connected component of  $G_Y^v$ .

Figure 2 shows an example of a graph-tree pair with a Y-splittable vertex, and highlights how  $v$  and its neighbouring edges can be split so that we can obtain  $(G', T')$ .

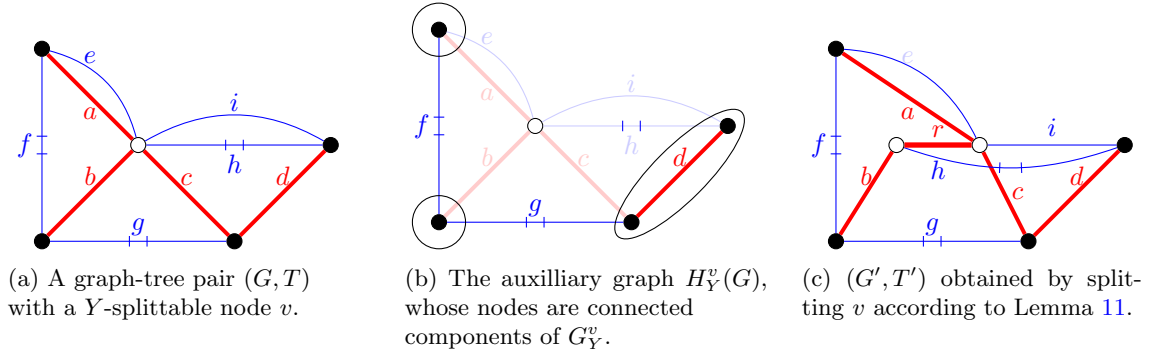


Figure 2: An example of a graph-tree pair  $(G, T)$  with a Y-splittable node  $v$ , its auxiliary graph  $H_Y^v(G)$  and the updated graph  $G'$  obtained after splitting  $v$  into two vertices, reassigning the neighbouring edges and adding the new row edge. Edges in  $T$  are marked red and bold, all other edges are marked in blue. The edges in  $Y = \{f, g, h\}$  are marked with two stripes. The Y-splittable node  $v$  is marked in white in a). In c), the two vertices that  $v$  is split into are marked in white, and the newly added row that is given by  $r$  connects them.

Based on Definition 5, a few basic but insightful results follow. Given a subset of edges  $Y \subseteq E$ , we say that  $Y$  is a *star centered at  $v$*  if  $Y \subseteq \delta(v)$ , such that all edges  $y \in Y$  have the vertex  $v$  in common.

**Proposition 6.** *Let  $G$  be a multigraph and let  $Y \subseteq E(G)$  be a subset of edges such that  $Y$  is a star centered at  $v$ . Then  $v$  is Y-splittable.*

*Proof.* From  $Y \subseteq \delta(v)$  it follows that the auxiliary graph  $H_Y^v$  has no edges, and thus  $H_Y^v$  is bipartite.  $\square$

**Corollary 7.** *Every vertex of a multigraph is  $\emptyset$ -splittable.*

*Proof.* Apply Proposition 6 to each vertex.  $\square$

We can also derive a few necessary conditions for a Y-splittable vertex  $v$  when  $Y$  is not a star.

**Lemma 8.** *Let  $G$  be a multigraph and let  $Y \subseteq E(G)$  be an edge subset. If  $v \in V(G)$  is Y-splittable and  $Y$  is not a star centered at  $v$ , then  $v$  is an articulation vertex of  $G_Y$ .*

*Proof.* As  $Y$  is not a star centered at  $v$ , there must exist an edge  $e \in Y$  that is not incident to  $v$ . Since  $e \in Y \setminus \delta(v)$ , it induces an edge  $\{h_1, h_2\}$  in the auxiliary graph  $H_Y^v$ . Since  $H_Y^v$  is bipartite,  $h_1 \neq h_2$  holds, which implies that  $G_Y^v$  has at least two connected components.  $\square$

**Lemma 9.** *Let  $G$  be a multigraph with spanning tree  $T \subseteq E(G)$  and a set of edges  $Y \subseteq E(G) \setminus T$ . If  $v$  is  $Y$ -splittable, then  $v$  must lie on the fundamental path  $P_y(T)$  for each  $y \in Y$ .*

*Proof.* Assume, for the sake of contradiction, that there exists an edge  $y \in Y$  such that  $v$  is not on  $P_y(T)$ . Since  $P_y(T) \subseteq T \setminus \delta(v)$  and  $T \cap Y = \emptyset$  hold,  $P_y(T)$  must lie in a single connected component of  $G_Y^v$ . This implies that  $H_Y^v$  has a loop for this component, and hence  $H_Y^v$  is not bipartite. This contradicts the assumption that  $v$  was  $Y$ -splittable, which concludes the proof.  $\square$

Proposition 6 and Lemmas 8 and 9 provide necessary conditions for a vertex  $v$  to be  $Y$ -splittable. Now, we consider the next step where we split  $v$  into two new vertices and reassign the edges incident to  $v$  to these vertices. This step is formalized in Definition 10 and Lemma 11, where Lemma 11 proves that for a  $Y$ -splittable node  $v$  a certain reassignment of incident edges elongates exactly the fundamental paths of edges  $Y$  by splitting  $v$  into two nodes  $v_1$  and  $v_2$  and subsequently setting the new tree edge  $r^*$  to connect  $v_1$  and  $v_2$ .

**Definition 10** (neighborhood split). *Let  $G$  be a multigraph with a set of edges  $Y \subseteq E(G)$ . Let  $v$  be a  $Y$ -splittable vertex and let  $I$  and  $J$  be the two sides of a corresponding bipartition of  $H_Y^v$ . Then the associated neighborhood split of  $v$  is the partition of  $\delta(v)$  into  $\delta^I(v)$  and  $\delta^J(v)$  defined via*

$$\delta^I(v) := \{\{u, v\} \in \delta(v) \mid \text{either } \{u, v\} \in Y \text{ or there exists } h \in I \text{ with } u \in h \text{ (but not both)}\}.$$

**Lemma 11.** *Let  $G$  be a multigraph with spanning tree  $T \subseteq E(G)$ , and let  $Y \subseteq E(G) \setminus T$  be a subset of the non-tree edges. Let  $v$  be a  $Y$ -splittable vertex and let  $I, J \subseteq V(H_Y^v)$  denote the bipartition of  $H_Y^v$ . Construct the multigraph  $G'$  from  $G$  by splitting  $v$  into vertices  $i$  and  $j$ , adding the new edge  $r^* = \{i, j\}$ , and replacing the edges  $\{u, v\} \in \delta(v)$  by  $\{u, i\}$  (resp. by  $\{u, j\}$ ) if  $u \in \delta^I(v)$  (resp.  $u \in \delta^J(v)$ ). Construct the spanning tree  $T' := T \cup \{r^*\}$  of  $G'$ . Then for each  $e \in E \setminus T$  we have*

$$P_e(T') = \begin{cases} P_e(T) \cup \{r^*\} & \text{if } e \in Y, \\ P_e(T) & \text{otherwise.} \end{cases}$$

*Proof.* First note that  $T'$  is indeed a spanning tree of  $G'$ .

Second, consider an edge  $e \in E \setminus (T \cup Y)$ . Denote by  $C := P_e(T) \cup \{e\}$  its fundamental cycle and observe that  $C \cap Y = \emptyset$  holds. If  $P_e(T) \cap \delta(v) = \emptyset$  then none of the edges in the path are changed by changing  $v$  and  $\delta(v)$  and thus  $P_e(T) = P_e(T')$  holds. Otherwise, let  $f, f' \in \delta(v) \cap C$  be the two cycle edges incident to  $v$ . From  $C \cap Y = \emptyset$  it follows that  $C \setminus \delta(v)$  is a path in the reduced graph  $G_Y^v$  and hence,  $f$  and  $f'$  connect to the same connected component of  $G_Y^v$ . Again from  $C \cap Y = \emptyset$  we have  $f, f' \notin Y$ , and thus  $f$  and  $f'$  are reassigned to the same vertex from  $\{i, j\}$  in  $G'$ . This implies that in  $G'$  the path  $P_e(T)$  remains unchanged, so we conclude that  $P_e(T) = P_e(T')$  holds.

Third, consider an edge  $y \in Y$ . Once again, denote by  $C := P_y(T) \cup \{y\}$  the fundamental cycle. By Lemma 9 we know that  $v$  must lie on  $P_y(T)$ , and we obtain  $C \cap \delta(v) = \{f, f'\}$  for suitable edges  $f, f'$ . We claim that these edges are reassigned to different end-vertices, and we distinguish two cases to prove it.

**Case 1:**  $y \in \{f, f'\}$ . We can assume  $y = f$  without loss of generality. Then  $f' \in T$  and  $P_y(T) \setminus \{f'\}$  is a path that lies in a single connected component of the reduced graph  $G_Y^v$ . Both  $y$  and  $f'$  must connect to this component. Since  $f \in Y$  and  $f' \notin Y$  hold, they are indeed reassigned to different end-vertices from  $\{i, j\}$  in  $G'$ .

**Case 2:**  $y \notin \{f, f'\}$ . In this case,  $P_y(T) \setminus \{f, f'\}$  is no longer a path and thus disconnected. Let  $P^1$  and  $P^2$  be the two (maximal) subpaths formed by  $P_y(T) \setminus \{f, f'\}$ , which may consist of just a single vertex. Each such path  $P^k$  ( $k = 1, 2$ ) lies in a single connected component  $h_k$  of  $G_Y^v$ . Since  $v$  is  $Y$ -splittable,  $H_Y^v$  is bipartite. Moreover, since  $y$  connects a vertex from  $P^1$  with one from  $P^2$ ,  $h_1$  and  $h_2$  must be on different sides of the bipartition. Due to  $f, f' \notin Y$ , they are indeed reassigned to different end-vertices from  $\{i, j\}$  in  $G'$ .

In both cases,  $f$  and  $f'$  were reassigned to different end-vertices from  $\{i, j\}$  in  $G'$ . Thus, the new edge  $r^* = \{i, j\}$  lies on the fundamental path of  $P_y(T')$ , which yields  $P_y(T') = P_y(T) \cup \{r^*\}$ . This concludes the proof.  $\square$

Now, we have the tools to show the main result of this section, which characterizes the graphic row augmentation problem for a matrix  $M$  in terms of  $Y$ -splittability of the graphs represented by  $M$ .

**Theorem 12.** *Let  $M$  and  $M'$  be binary matrices and  $b$  be a binary vector as in (1) and define  $Y := \text{supp}(b^\top)$ . Then  $M'$  is graphic if and only if there exists a graph  $G = (V, E)$  and a tree  $T \subseteq E$  with  $M = M(G, T)$  such that  $G$  has a  $Y$ -splittable vertex.*

*Proof.* To prove sufficiency, assume that  $M'$  is graphic, i.e., it has a realization  $M' = M(G', T')$ . Let  $r^*$  index the last row given by  $b^\top$ . Removal of row  $r^*$  from the matrix  $M'$  corresponds to the contraction of the edge  $r^* \in T'$  with end-vertices  $v_1, v_2$  into a new vertex  $v$ . Hence,  $M = M(G, T)$  holds, where  $T = T'/r^*$  and  $G = G'/r^*$ .

We claim that  $v$  is  $Y$ -splittable in  $G$ . If  $Y$  is a star centered at  $v$ , the statement follows from Proposition 6. Otherwise,  $Y$  is not a star centered at  $v$  so there must exist some edge  $y \in Y$  that is not incident to  $v$ . Deletion of  $r^*$  from the tree  $T'$  yields two trees  $T_1$  and  $T_2$  with vertex sets  $K_1 = V(T_1)$  and  $K_2 = V(T_2)$ , respectively, where  $v_1 \in K_1$  and  $v_2 \in K_2$ . Note that the edge  $y$  must connect  $K_1$  with  $K_2$ , since its fundamental path  $P_y(T')$  must contain  $r^*$  because  $M'$  is graphic. As  $y$  is not incident to  $v$  in  $G$ , there must exist vertices  $w_1 \in K_1$ ,  $w_1 \neq v_1$  and  $w_2 \in K_2$ ,  $w_2 \neq v_2$ . This shows that  $|K_i| \geq 2$  for  $i = 1, 2$ .

Consider any edge  $e \in E(G') \setminus T'$  that connects  $K_1$  with  $K_2$ . Since  $r^* \in P_e(T')$  holds, we must have  $e \in Y$ , and all edges in  $Y$  must connect  $K_1$  to  $K_2$ . Hence, no such edge belongs to  $G_Y$ . Then the vertex sets  $K_i \setminus \{v_i\}$  for  $i = 1, 2$  are non-empty (since  $|K_i| \geq 2$ ) and belong to  $G$ . Since  $K_1 \setminus \{v_1\}$  is not connected to  $K_2 \setminus \{v_2\}$  by any edge in  $G_Y$ , we conclude that  $v$  is an articulation vertex of  $G_Y$ .

Now, let us show that  $H_Y^v$  is bipartite. For  $i = 1, 2$ , we define  $H_i := \{h \in V(H_Y^v) : V(h) \subseteq K_i \setminus \{v_i\}\}$ . Clearly,  $H_1$  and  $H_2$  are disjoint since  $K_1$  and  $K_2$  are. The vertices of any connected component of  $G_Y^v$  are a subset of either  $K_1 \setminus \{v_1\}$  or  $K_2 \setminus \{v_2\}$  as these vertex sets are disconnected in  $G_Y^v$ . Then  $H_1 \cup H_2 = V(H_Y^v)$  holds since these vertex sets cover all vertices of  $G_Y^v$ , i.e.,  $(K_1 \setminus \{v_1\}) \cup (K_2 \setminus \{v_2\}) = V(G_Y^v)$ . Thus,  $H_1$  and  $H_2$  form a bipartition of the components  $V(H_Y^v)$ . As argued above, every edge  $y \in Y$  must connect  $K_1$  to  $K_2$  as  $M'$  is graphic. This implies that every edge  $(h_i, h_j) \in E(H_Y^v)$  has  $h_i \in H_1$  and  $h_j \in H_2$  (or vice versa). This shows that  $H_Y^v$  is bipartite with bipartition  $H_1, H_2$ . We conclude that  $v$  is  $Y$ -splittable.

To prove necessity, we consider a graph  $G$  with spanning tree  $T \subseteq E(G)$  with  $M = M(G, T)$  that has a  $Y$ -splittable vertex  $v$ . We obtain  $G'$  and  $T'$  according to Lemma 11. Then Lemma 11 shows that  $G'$  correctly elongates the fundamental paths of all edges in  $Y$  and does not modify those of other edges. This shows  $M' = M(G', T')$  and in particular that  $M'$  is graphic.  $\square$

Theorem 12 provides a good characterization of when the matrix  $M'$  is graphic. However, it is difficult to turn into a full algorithm. In particular, it does not tell us how to find  $G$  explicitly. This is problematic, as there may exist a large number of graphs that each realize  $M$ , of which only a few may have a  $Y$ -splittable vertex. Figure 3 shows an example with a series of graphic matrices with  $m$  rows and a given set  $Y$ , for which only a fraction of  $2/m$  of all represented graphs contains a  $Y$ -splittable vertex.

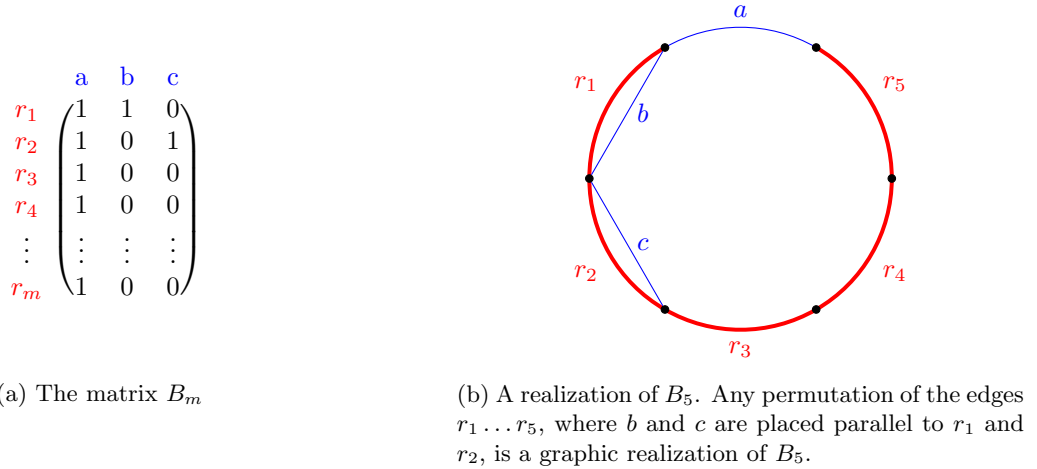


Figure 3: A series of matrices  $B_m$  with  $m \geq 2$  rows where each matrix  $B_m$  has  $\frac{m!}{2}$  realizations. Only  $(m-1)!$  realizations contain a  $\{b, c\}$ -splittable node (when  $b$  and  $c$  share an adjacent node). Thus, the fraction of graphs that realizes  $B_m$  and contains a  $\{b, c\}$ -splittable node is  $2/m$ .

In the next section, we will explain how one can use an *SPQR tree*, which is a graph decomposition that represents all graphs that realize  $M$ . Then our goal will become to update the SPQR tree of  $M$  to an SPQR tree for  $M'$ , by efficiently finding the  $Y$ -splittable vertices across all graphs realizing  $M$ .

## 4 Representing graphic matrices using SPQR trees

In the previous section we showed that the augmentation problem can be reduced to the search for a  $Y$ -splittable vertex in case the matrix  $M$  in (1) represents a unique graph. However, this does not need to be the case. Note that we can assume that  $M$  is connected by Theorem 3 and by Proposition 4 that every realization  $G$  (with spanning tree  $T$  such that  $M = M(G, T)$  holds) is 2-connected.

Suppose that  $G$  has a 2-separation  $(E_1, E_2)$  with separating vertices  $\{u, v\} = V(E_1) \cap V(E_2)$ . Let  $G'$  be a graph with the same edge labels but with the sets of incident edges of  $u$  and  $v$  in  $E_2$  swapped, i.e.,

setting  $\delta_{G'}(u) := (\delta_G(u) \cap E_1) \cup (\delta_G(v) \cap E_2)$  and  $\delta_{G'}(v) := (\delta_G(v) \cap E_1) \cup (\delta_G(u) \cap E_2)$ , and leaving all other edges unchanged. We say that  $G'$  is obtained from  $G$  by *reversing*  $E_2$ . Moreover, two graphs are called *2-isomorphic* if one can be obtained from the other by a sequence of reversals (for arbitrary 2-separations). An example of two 2-isomorphic graphs can be found in Fig. 4. Whitney [38, 39] showed that two graphs  $G$  and  $G'$  are 2-isomorphic if and only if they have the same cycles. In particular, 3-connected graphs are uniquely determined by their cycles.

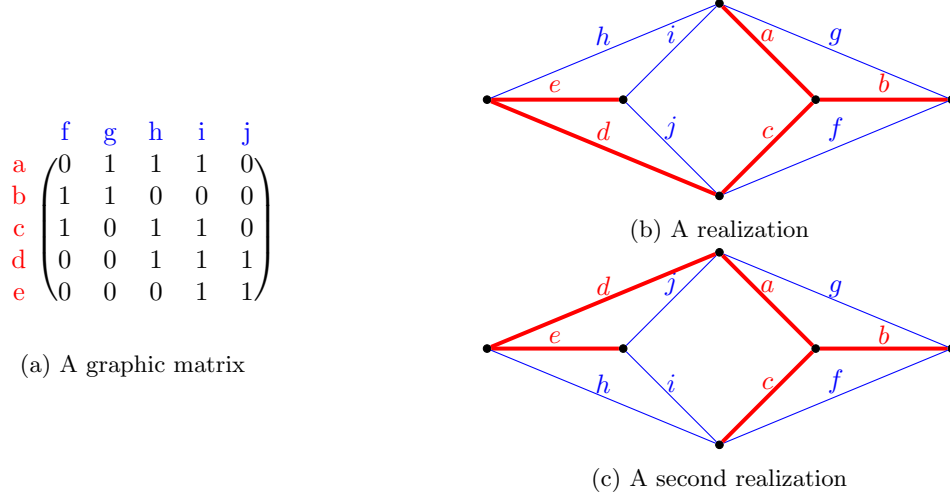


Figure 4: A graphic matrix with two 2-isomorphic realizations. The realization in c) can be obtained from b) using a reversal on the 2-separation given by  $E_1 = \{d, e, h, i, j\}$  and  $E_2 = E \setminus E_1$ .

This is highly relevant for the graph realization problem since the columns of a matrix  $M = M(G, T)$  encode the set of *fundamental cycles*  $P_e(T) \cup \{e\}$  for all  $e \in E(G) \setminus T$ . Consequently, 2-isomorphic graphs  $G$  and  $G'$  (with a spanning tree  $T$ ) satisfy  $M(G, T) = M(G', T)$ . Despite the fact that  $M(G, T)$  only encodes a (usually small) subset of  $G$ 's cycles, the reverse implication also holds. This follows from the fact that these fundamental cycles form a basis of the cycle vector space [12, Section 1.9].

**Proposition 13** (Consequence of Whitney [38, 39]). *Let  $G$  and  $G'$  be 2-connected graphs with identical edge labels  $E(G) = E(G')$  and let  $T$  be a spanning tree of  $G$ . Then  $M(G, T) = M(G', T)$  holds if and only if  $G$  and  $G'$  are 2-isomorphic. In particular, in the affirmative case  $T$  is also a spanning tree of  $G'$ .*

Thus, in order to represent the set of graphs represented by  $M$  we could, in principle, maintain one graph  $G$  that realizes  $M$  and maintain all its 2-separations, and thus its possible reversals. This is done by the fast column-wise algorithms for graph realization proposed by Bixby and Wagner [6] and Fujishige [13] by means of so-called *t-decompositions* and *PQ-graphs*, respectively. The former relies on fundamental work of Cunningham and Edmonds [9]. A variant of the t-decomposition appears in the more modern literature under the name *SPQR tree* or *SPQR decomposition* [10, 11, 16].

We will stick to that notion, but augment the represented 2-isomorphic graphs  $G$  with a spanning tree  $T \subseteq E(G)$ . Moreover, in order to keep the presentation simple, our notion of an SPQR tree is undirected.

**Definition 14** (SPQR tree, virtual edges, regular edges). *An SPQR tree is a tree  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$  that is supported by a map  $\pi$ . For every node  $\mu \in \mathcal{V}$  there is an associated connected multi-graph  $G_\mu$  without loops, called the skeleton, and a spanning tree  $T_\mu$  of  $G_\mu$ . We assume that the vertex sets of all skeletons are pairwise disjoint. We denote by  $E(\mathcal{T})$  the union of the edges  $E(G_\mu)$  over all  $\mu \in \mathcal{V}$ , and by  $V(\mathcal{T})$  the union of the vertices  $V(G_\mu)$  over all  $\mu \in \mathcal{V}$ .*

*A subset  $E^{\text{virt}}(\mathcal{T}) \subseteq E(\mathcal{T})$  of size  $|E^{\text{virt}}(\mathcal{T})| = 2|\mathcal{E}|$  of the edges is called virtual: for every edge  $\{\mu, \nu\} \in \mathcal{E}$  of the SPQR tree there is a pair of virtual edges  $e \in E(G_\mu)$  and  $f \in E(G_\nu)$ . This relationship is specified by means of the map  $\pi : E(\mathcal{T}) \rightarrow E(\mathcal{T}) \cup \{\emptyset\}$ , where  $\pi(e) = f$  and  $\pi(f) = e$  indicates that  $e \in E(G_\mu)$  and  $f \in E(G_\nu)$  are the two virtual edges corresponding to the edge  $\{\mu, \nu\} \in \mathcal{E}$ , and where  $\pi(e) = \emptyset$  holds whenever  $e$  is no virtual edge. Every virtual edge corresponds to exactly one edge in  $\mathcal{E}$ . The remaining edges  $E^{\text{reg}}(\mathcal{T}) := E(\mathcal{T}) \setminus E^{\text{virt}}(\mathcal{T})$  are called regular. By  $E_\mu^{\text{reg}}(\mathcal{T})$  we denote all regular edges  $E(G_\mu) \cap E^{\text{reg}}(\mathcal{T})$ . For each virtual edge pair  $e, f$ , either  $e \in T_\mu$  or  $f \in T_\nu$  holds, i.e., exactly one of them is part of the respective spanning trees. Moreover, each skeleton  $G_\mu$  is of exactly one of four types:*

- (S)  $G_\mu$  is a cycle of length at least 3;  $\mu$  is called series.
- (P)  $G_\mu$  has exactly two vertices and at least 3 edges;  $\mu$  is called parallel.
- (Q)  $G_\mu$  has at most 2 vertices and at most 2 edges, and  $\mu$  is the only skeleton, i.e.,  $\mathcal{V} = \{\mu\}$ .
- (R)  $G_\mu$  is simple, 3-connected and has at least 4 edges;  $\mu$  is called rigid.

*An SPQR tree is called minimal if its (S)-nodes are pairwise non-adjacent and its (P)-nodes are pairwise non-adjacent.*



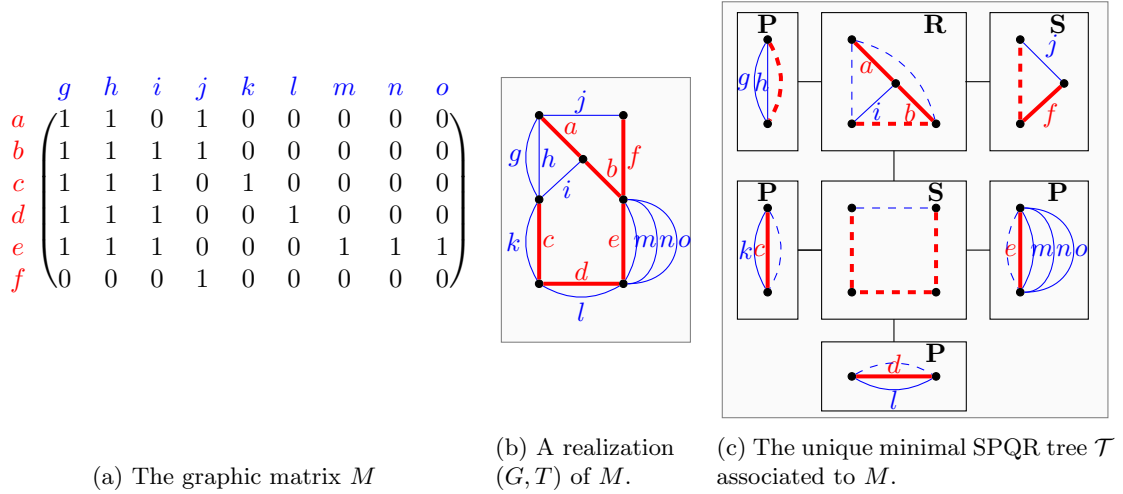


Figure 5: A graphic matrix with a realization and its unique minimal SPQR tree.

An example of a graphic matrix and its associated SPQR tree is depicted in Fig. 5.

To avoid confusion, we use *node* when referring to some  $\mu \in \mathcal{V}(\mathcal{T})$  for some SPQR tree  $\mathcal{T}$ , and we use *vertex* for the represented graphs. The above definition differs slightly from that of Gutwenger and Mutzel [16]. In particular, we do not consider each non-virtual edge to have its unique leaf node of type (Q), but rather only use (Q) when the graph has at most two edges, which is closer to the definition of  $t$ -decompositions as used by Bixby and Wagner [6].

To see how an SPQR tree  $\mathcal{T}$  encodes a family of 2-isomorphic graphs, consider one such graph  $G$ . First,  $E(G)$  consists of all the regular edges (of all skeletons) of  $\mathcal{T}$ . Second, an edge  $\{\mu, \nu\}$  with virtual edges  $e \in E(G_\mu)$  and  $f \in E(G_\nu)$  represents a 2-separation  $(E_\mu, E_\nu)$  of  $G$ , where  $E_\mu$  (resp.  $E_\nu$ ) consists of all regular edges of skeletons that are closer to  $\mu$  than to  $\nu$  (resp. closer to  $\nu$  than to  $\mu$ ). In Fig. 5, an example of such a 2-separation is the 2-separation defined by  $E_1 = \{a, b, f, g, h, i, j\}$  and  $E_2 = E \setminus E_1$ .

Although the edges of the SPQR tree encode many of the possible 2-separations of its represented graphs, there may still be additional 2-separations that exist within nodes of type (S) and (P). We will call a 2-separation  $(E_1, E_2)$  of any graph represented by  $\mathcal{T}$  *local* if  $E_i \subseteq E_\mu^{\text{reg}}(\mathcal{T})$  holds for some  $\mu \in \mathcal{V}(\mathcal{T})$  and some  $i \in \{1, 2\}$ . In other words, local 2-separations are those for which one side is contained entirely in a single skeleton of the SPQR tree. In Fig. 5, an example of such a 2-separation is the 2-separation defined by  $E_1 = \{n, o\}$  and  $E_2 = E \setminus E_1$ .

Note that every pair of non-adjacent vertices of a cycle are the separating vertices of a 2-separation. A corresponding reversal yields some reordering of the cycle edges. In fact, any permutation of a cycle's edges yields a 2-isomorphic cycle. These observations motivate the following definition.

**Definition 15** (Graphs and trees represented by an SPQR tree). *Let  $\mathcal{T}$  be an SPQR tree supported by the map  $\pi$ . A graph-tree pair  $(G, T)$  is represented by  $\mathcal{T}$  if it can be obtained by the following steps:*

1. For each  $\mu \in \mathcal{V}(\mathcal{T})$  of type (S), permute the edges of the cycle  $G_\mu$  arbitrarily.
2. Contract each edge  $\{\mu, \nu\} \in \mathcal{E}(\mathcal{T})$  (in any order) to a new node  $\xi$ . Let  $e \in E(G_\mu)$  and  $f \in E(G_\nu)$  be the two corresponding virtual edges. The skeleton  $G_\xi$  is the graph obtained from  $G_\mu$  and  $G_\nu$  by identifying the two end-vertices of  $e$  with the two end-vertices of  $f$  (via an arbitrary bijection) and then removing  $e$  and  $f$ . Define  $T_\xi$  to be  $(T_\mu \cup T_\nu) \setminus \{e, f\}$ . Define  $\Phi_G : \mathcal{V}(\mathcal{T}) \rightarrow \mathcal{V}(G)$  to be the mapping from the skeleton vertices to the resulting graph vertices, where initially  $\Phi_G(v) = v$  holds. For the end nodes  $w_e \in e$  and  $w_f \in f$  that are identified with one another into a new node  $v'$ , we update  $\Phi_G$  for all  $u$  where  $\Phi_G(u) = w_e$  or  $\Phi_G(u) = w_f$  holds to  $\Phi_G(u) = v'$ , instead.

Note that in Step 2 of the definition, since either  $e \in T_\mu$  or  $f \in T_\nu$  holds, exactly one of the edges  $e, f$  is removed from  $T_\mu \cup T_\nu$ . This (inductively) ensures that  $T_\xi$  is a spanning tree of  $G_\xi$ . By construction, any two graphs  $G, G'$  that are represented by  $\mathcal{T}$  are 2-isomorphic. Moreover, every 2-separation of a represented graph either corresponds to an edge of  $\mathcal{T}$  or to a 2-separation of an (S)- or a (P)-node. Cunningham and Edmonds have shown this more formally:

**Proposition 16** (Consequence of Cunningham and Edmonds [9], Theorem 3). *Let  $G$  be a graph represented by  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ , and let  $(E_1, E_2)$  be a 2-separation of  $G$ . Then,  $(E_1, E_2)$  is either a local 2-separation for some  $\mu \in \mathcal{V}(\mathcal{T})$ , or there exists an SPQR tree edge  $\epsilon = \{\mu, \nu\} \in \mathcal{E}$  such that the subtrees  $\mathcal{T}_\mu$  and  $\mathcal{T}_\nu$  formed by removing  $\epsilon$  from  $\mathcal{T}$  satisfy  $E_1 = E^{\text{reg}}(\mathcal{T}_\mu)$  and  $E_2 = E^{\text{reg}}(\mathcal{T}_\nu)$ .*

In the literature, SPQR trees are defined without the spanning tree  $T_\mu$  of each skeleton  $G_\mu$ . However, the last statement of Proposition 13 shows that the SPQR tree structure is independent of the spanning tree. By results from Hopcroft and Tarjan [19, Lemma 2] and Cunningham and Edmonds [9, Theorem 1],

minimal SPQR trees are unique. The discussion above and Proposition 13 together are summarized in the main result of this section.

**Theorem 17.** *Let  $A$  be a connected graphic matrix. Then there exists a unique minimal SPQR tree  $\mathcal{T}$  for which the set of represented graph-tree pairs  $(G, T)$  is equal to the set of graph-tree pairs  $(G, T)$  for which  $A = M(G, T)$  holds.*

The theorem in particular shows that the resulting SPQR tree  $\mathcal{T}$  is independent of the choice of  $(G, T)$  (among those represented by  $\mathcal{T}$ ). Consequently, if we construct a minimal SPQR tree  $\mathcal{T}$  from any graph-tree pair  $(G, T)$  for which  $A = M(G, T)$  holds, then we can (in principle) obtain all other graph-tree pairs  $(G', T')$  for which  $A = M(G', T')$  holds by enumerating all graph-tree pairs that are represented by  $\mathcal{T}$ .

In the next sections we will explain how to update an SPQR tree for a matrix  $M$  when augmenting it with a new row.

## 5 Graphic row augmentation using SPQR trees

In order to develop an algorithm that correctly determines graphicness of row augmentation, we need to combine the results of the previous sections. First, recall that in Section 3 we characterized graphic row augmentation by means of determining whether a given graph has a  $Y$ -splittable vertex. Second, in Section 4 we observed that the set of graphs represented by graphic matrices are exactly the set of graphs represented by a minimal SPQR tree. Combining these two ideas, the row augmentation problem then becomes to determine if the SPQR tree represents a graph that has a  $Y$ -splittable vertex, and if so, to find the unique minimal SPQR tree of the matrix after it has been updated by splitting a  $Y$ -splittable vertex in one of the represented graphs.

Note that by Theorem 17 it is sufficient to show that a single realization  $G$  of the original SPQR tree has a  $Y$ -splittable vertex: we could find all represented graphs of  $M'$  by computing the unique minimal SPQR tree of  $G'$ , where  $G'$  is obtained by splitting said  $Y$ -splittable vertex in  $G$ . In our algorithm we do not recompute the SPQR tree of  $G'$  from scratch, but instead construct it by modifying the SPQR tree of  $G$ . To establish the correctness of our algorithm, we additionally argue that the updated SPQR tree is minimal. In case of a non-graphic row augmentation we need to show that none of the graphs represented by the SPQR tree contains a  $Y$ -splittable vertex, which then by Theorem 17 shows that  $M'$  is not graphic.

Let us describe our proposed algorithm in general terms. First, it computes the set  $Y$  from the given row vector. Then the algorithm carries out certain reductions on the SPQR tree that preserve the existence of a  $Y$ -splittable vertex in any represented graph. These reductions work by successively removing suitable leaf nodes from the SPQR tree. Once no more reductions can be applied, the resulting *reduced* SPQR tree and its represented graphs have certain properties. In particular, if one of the represented graphs has a unique  $Y$ -splittable vertex and can not be further reduced, then splitting according to Lemma 11 will yield a 3-connected simple graph. From this it follows that the reduced SPQR tree is merged into a single node of type (R).

We start with a few structural results relating the concepts of Section 3 to 2-connected graphs. We first show that the bipartition of  $H_Y^v$  is unique for any  $Y$ -splittable vertex  $v$ , which shows that the resulting neighbourhood split is also unique.

**Lemma 18.** *Let  $G = (V, E)$  be a 2-connected multigraph with spanning tree  $T \subseteq E$ , let  $Y \subseteq E \setminus T$  and  $v \in V$ . Then  $H_Y^v$  is a connected graph.*

*Proof.* Let  $C$  be a connected component of  $H_Y^v$  and let  $U := \bigcup_{h \in V(C)} V(h)$  be the vertices of  $G_Y^v$  that belong to any vertex of  $H_Y^v$  belonging to  $C$ . Since  $G$  is 2-connected,  $G[V \setminus \{v\}]$  is connected. This implies that either  $W := V \setminus (U \cup \{v\})$  is empty or that, by connectivity,  $E$  contains an edge  $e = \{u, w\}$  with  $u \in U$  and  $w \in W$ . By construction, we would have  $e \in Y$ , which would contradict the assumption that  $C$  is a connected component of  $H_Y^v$ . We conclude that  $W = \emptyset$  holds, which proves that  $C$  is the only connected component of  $H_Y^v$ .  $\square$

**Corollary 19.** *Let  $G = (V, E)$  be a 2-connected multigraph with spanning tree  $T \subseteq E$  and let  $Y \subseteq E \setminus T$ . If  $H_Y^v$  is bipartite then it has a unique bipartition.*

*Proof.* By Lemma 18,  $H_Y^v$  must be connected. This implies that if  $H_Y^v$  is bipartite, that its bipartition is unique (up to switching sides).  $\square$

### 5.1 Reductions

In this section we present three reductions that preserve the existence of  $Y$ -splittable vertices and shrink the represented graph  $G$ . Since we are interested in reductions that work not just on  $G$  but also on its SPQR tree, it is natural to consider 2-separations of the represented 2-connected graphs. All three proposed reductions work in a similar fashion. For each reduction, we consider a 2-separation with a certain structure, and then replace one of its sides by a single edge, whilst preserving the splittability of any vertices in the remaining graph. Although we formulate the results in terms of general graphs, the

relevance of the following results on the SPQR tree are due to the fact that every 2-separation is encoded in the SPQR tree. Moreover, the proposed reductions do not depend on the orientation of the 2-separation; they work for both the original graph and its reversal, so that we can apply them directly to an SPQR tree  $\mathcal{T}$ , simultaneously performing the reduction on all represented graphs of  $\mathcal{T}$ .

First, let us introduce some notation to define the structure of a graph-tree pair  $(G, T)$  when considering a 2-separation of  $G$ .

**Definition 20** (tree partition induced by a 2-separation). *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  and the corresponding graphs  $G_1$  and  $G_2$  such that  $P_{u,w}(T) \subseteq E(G_1)$  holds. The tree partition of  $T$  induced by the 2-separation is the partition  $T = T_1 \cup T_u \cup T_w$  such that  $T_1 := T \cap E(G_1)$  and  $T_v := \{e \in T \cap E(G_2) \mid e \text{ and } v \text{ are in the same connected component of } T \setminus T_1\}$  for  $v \in \{u, w\}$ . The corresponding vertex partition is  $V(G_2) = V_u \cup V_w$  with  $V_v := V(T_v) \cup \{v\}$  for  $v \in \{u, w\}$ . The edges whose fundamental path contains  $P_{u,w}(T)$  are denoted by*

$$P_{u,w}^{-1}(G, T) := \{e \in E(G) \setminus T \mid P_{u,w}(T) \subseteq P_e(T)\}.$$

Note that the definition above ensures for  $v \in \{u, w\}$  that  $T_v = \emptyset$  implies  $V_v = \{v\}$ . Moreover, note that  $P_{u,w}^{-1}(G, T) \cap E_2$  consists of exactly those edges that connect  $V_u$  to  $V_w$  in  $E_2$ .

Lemma 21 provides a basic result which we will need several times. Thereafter, we show in Lemmas 22 and 23 that when  $E_i \cap Y = \emptyset$  holds,  $E_i$  can be replaced by a single edge whilst preserving splittability of the remaining graph. An overview of all three reductions can be found in Fig. 6.

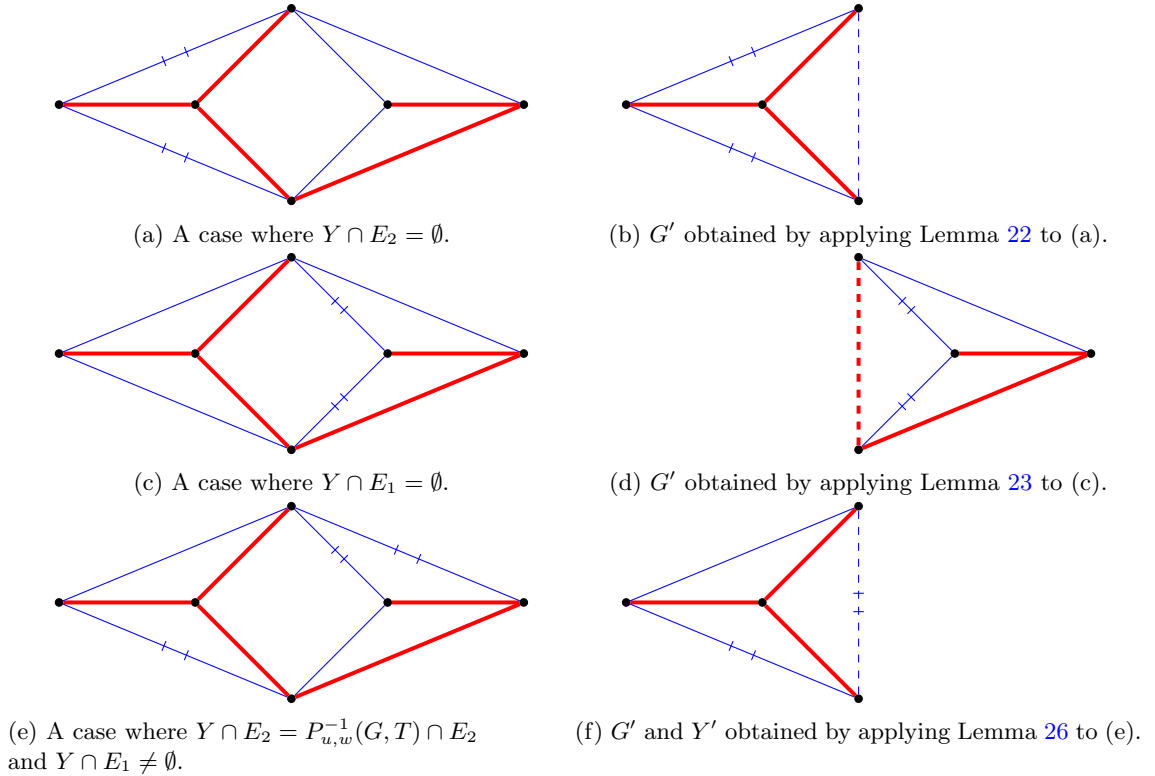


Figure 6: A graph-tree pair  $(G, T)$  with a 2-separation  $E_1, E_2$  with  $V(E_1) \cap V(E_2) = \{u, w\}$  such that  $P_{u,w}(T) \subseteq E_1$ . Subfigures (a),(c) and (e) present different sets  $Y$ . Subfigures (b),(d) and (f) present the reduced graph  $G'$  (and reduced set  $Y'$ ) obtained by applying a relevant reduction to (a),(c) and (e) respectively. The new edge introduced in  $G'$  is presented as a dashed edge. Edges in  $T$  or  $T'$  are marked bold and red, and edges in  $Y$  or  $Y'$  are marked with two stripes.

**Lemma 21.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  and the corresponding graphs  $G_1$  and  $G_2$  such that  $P_{u,w}(T) \subseteq E_1$  holds. Then any edge  $e \in E_1 \setminus T$  has  $P_e(T) \subseteq E_1$ , and any edge  $e \in E_2 \setminus T$  has  $P_e(T) \subseteq E_2 \cup P_{u,w}(T)$ .*

*Proof.* Consider an edge  $e \in E_1 \setminus T$ . As  $T_1$  is a spanning tree, there exists a path between every vertex pair  $u', w' \in V(G_1)$ , and  $P_e(T) \subseteq T_1$  must hold. By definition of  $T_1$  we have  $P_e(T) \subseteq T_1 \subseteq E_1$ .

Similarly, if both end-vertices of  $e \in E_2 \setminus T$  lie in  $T_u$  or  $T_w$  then  $P_e(T) \subseteq E_2$  holds as there exists a unique path between them in  $T_u$  or  $T_w$ . In the other case, the end-vertices of  $e$  lie on both  $T_u$  and  $T_w$ . Then  $P_e(T)$  must be a subset of  $T_u \cup T_w \cup P_{u,w}(T)$ , as  $u$  and  $w$  are connected by a unique path in  $T$ .  $\square$

**Lemma 22.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be non-empty. Let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  and the corresponding graphs  $G_1$  and  $G_2$  such that  $P_{u,w}(T) \subseteq E_1$  and  $Y \cap E_2 = \emptyset$  holds. Let  $G'$  be  $G_1$  augmented by a new edge  $e = \{u, w\}$ . Then the following hold:*

1. *A vertex  $v \in V(G)$  is  $Y$ -splittable with respect to  $G$  if and only if  $v \in V(G_1)$  and  $v$  is  $Y$ -splittable with respect to  $G'$ .*
2. *For any  $v \in V(G_1)$  that is  $Y$ -splittable in  $G$ , let  $\widehat{G}$  be the graph obtained by applying Lemma 11 to  $v$ . Similarly, let  $\widehat{G}'$  be the graph obtained from  $G'$  by applying Lemma 11 to  $v$ , and let  $\widehat{G}''$  be the graph obtained from  $\widehat{G}'$  by replacing  $e$  with  $G_2$ . Then  $\widehat{G}'' = \widehat{G}$ .*

*Proof.* Let us prove the first statement. First, consider a vertex  $v \in V(G_2) \setminus V(G_1)$ . Let  $y \in Y$ , which exists since  $Y \neq \emptyset$ , and note that  $y \in E_1$  since  $Y \cap E_2 = \emptyset$  holds. By Lemma 21,  $P_y(T) \subseteq E_1$ , and by Lemma 9, all  $Y$ -splittable vertices lie on  $P_y(T)$ . We conclude that  $v$  is not  $Y$ -splittable.

Second, consider a vertex  $v \in V(G_1)$ . Since  $G$  is 2-connected,  $v$  is not an articulation vertex of  $G$ , and thus all vertices of  $V(G_2) \setminus \{v\}$  are in the same connected component of  $G$  after removal of  $v$ . Since  $Y \cap E_2 = \emptyset$  holds, all these vertices are even in the same connected component  $h$  of  $G_Y^v$ . Similarly, by definition of  $G'$ , all vertices in  $\{u, w\} \setminus \{v\}$  are in the same connected component  $h'$  of  $G_Y'^v$ . For any vertex  $v' \in V(G_1) \setminus \{u, w\}$ , the neighborhood of  $v'$  is identical in  $G$  and  $G'$ . Hence, the connected components of  $G_Y^v$  and  $G_Y'^v$  are identical, except that in one component  $V(G_2) \setminus \{v\}$  is replaced by  $\{u, w\} \setminus \{v\}$ . This implies that the vertex sets of  $H_Y^v(G)$  and  $H_Y^v(G')$  are isomorphic by identifying  $h$  with  $h'$ . Since  $Y \subseteq E_1$  is the same set in  $G$  and  $G'$ , the graphs  $H_Y^v(G)$  and  $H_Y^v(G')$  are then isomorphic. This implies that  $v \in V(G_1)$  is  $Y$ -splittable with respect to  $G$  if and only if  $v$  is  $Y$ -splittable with respect to  $G'$ .

Now, let us prove the second statement. First, note that  $\widehat{G}''$  and  $\widehat{G}$  have the same edge set and the same number of vertices. To show their equality it thus suffices to show that the incident edges for each vertex are equal. Both  $\widehat{G}''$  and  $\widehat{G}$  were obtained by performing operations that only change the end-vertices of edges incident to  $v$  and re-link these edges either to  $v_1$  or to  $v_2$ . In particular, replacing  $E_2$  by  $e$  and replacing  $e$  by  $E_2$  again does not change the edges incident to vertices  $V(G_2) \setminus \{u, w\}$ . Thus, it suffices to show for the vertices  $v_1$  and  $v_2$  that were obtained from the neighborhood split of  $v$  in  $G$ , and  $v'_1$  and  $v'_2$  that were obtained from the neighborhood split of  $v$  in  $G'$ , that  $\delta_{\widehat{G}}(v_1) = \delta_{\widehat{G}''}(v'_1)$  holds.

Without loss of generality, we assume that  $v_1$  was obtained from the  $I$ -part of the neighborhood split, such that  $\delta_{\widehat{G}}(v_1) = \delta_G^I(v)$  holds. Since  $E_1$  and  $E_2$  partition the edges of the graph,  $\delta_G^I(v) \cap E_1$  and  $\delta_G^I(v) \cap E_2$  form a partition of  $\delta_G^I(v)$ .

Consider  $\delta_G^I(v) \cap E_1$ . We know that  $G$  is 2-connected, and it is easy to see that also  $G'$  is 2-connected. Then, because  $G$  and  $G'$  are 2-connected,  $H_Y^v(G)$  and  $H_Y^v(G')$  both have a unique bipartition by Corollary 19, which we denote by  $(I, J)$  and  $(I', J')$  respectively. In particular, it must be the case that  $I$  and  $I'$  both contain the same connected components (after identification) of  $G_Y^v$  and  $G_Y'^v$ . Without loss of generality we assume that  $h \in I$  and  $h' \in I'$  for the connected components  $h$  and  $h'$  that contain  $\{u, w\} \setminus \{v\}$ . Because all edges of  $E_1$  are also contained in  $G'$ , we then have that  $\delta_G^I(v) \cap E_1 = \delta_{G'}^{I'}(v) \cap E_1$ . Then, by Definition 10, we have that  $\delta_{G'}^{I'}(v) \cap E_1 = \delta_{\widehat{G}'}(v'_1) \cap E_1$ . Clearly,  $\delta_{\widehat{G}'}(v'_1) \cap E_1 = \delta_{\widehat{G}''}(v'_1) \cap E_1$  holds, since  $e \notin E_1$  and  $e$  is replaced only by edges in  $E_2$ . Thus, we conclude that  $\delta_G^I(v) \cap E_1 = \delta_{\widehat{G}''}(v'_1) \cap E_1$ .

Secondly, consider  $\delta_G^I(v) \cap E_2$ . In  $G'$ , we replace  $E_2$  by  $e$ . If  $v \notin \{u, w\}$ , then  $\delta_G(v) \cap E_2 = \emptyset$ , and replacing  $G_2$  by  $e$  and subsequently replacing  $e$  back by  $G_2$  does not modify the neighborhood of any vertex of the graph. In particular, we then have that  $\delta_{\widehat{G}''}(v_1) \cap E_2 = \emptyset$ . If  $v \in \{u, w\}$ , then replacing  $G_2$  by  $e$ , splitting the vertex, and then replacing  $e$  by  $G_2$ , means that all edges of  $\delta_{G'}(v) \cap E_2$  lie in the same partition of the neighborhood split of  $G'$ . However, we proved earlier that this must exactly be the case; since  $h$  and  $h'$  lie in  $I$  and  $I'$  respectively, we have  $E_2 \subseteq \delta_G^I(v)$  and  $e \in \delta_{G'}^{I'}(v)$ , which implies that  $e \in \delta_{\widehat{G}'}(v'_1)$ . Then, by replacing  $e$  with  $E_2$  in  $\widehat{G}''$ , we have that  $\delta_G^I(v) \cap E_2 = E_2 = \delta_{\widehat{G}''}(v'_1) \cap E_2$  holds. Thus,  $\delta_{\widehat{G}''}(v'_1) \cap E_2 = \delta_G^I(v) \cap E_2$ .

We conclude that  $\delta_{\widehat{G}}(v_1) = \delta_G^I(v) = (\delta_G^I(v) \cap E_1) \cup (\delta_G^I(v) \cap E_2) = (\delta_{\widehat{G}'}(v'_1) \cap E_1) \cup (\delta_{\widehat{G}''}(v'_1) \cap E_2) = \delta_{\widehat{G}''}(v'_1)$  holds, which establishes  $\widehat{G} = \widehat{G}''$ .  $\square$

**Lemma 23.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be non-empty. Let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  and the corresponding graphs  $G_1$  and  $G_2$  such that  $P_{u,w}(T) \subseteq E_1$  and  $Y \cap E_1 = \emptyset$  holds. Let  $G'$  be  $G_2$  augmented by a new edge  $e = \{u, w\}$  and define  $T' := (T \cap E_2) \cup \{e\}$ . Then the following hold:*

1. *A vertex  $v \in V(G_2)$  is  $Y$ -splittable with respect to  $G$  if and only if  $v$  is  $Y$ -splittable with respect to  $G'$ .*
2. *For any  $v \in V(G_2)$  that is  $Y$ -splittable, let  $\widehat{G}$  be the graph obtained by applying Lemma 11 to  $v$ . Let  $\widehat{G}'$  be the graph obtained by applying Lemma 11 to  $v$  in  $G'$ , and let  $\widehat{G}''$  be the graph obtained by replacing  $e$  with  $G_1$  in  $\widehat{G}'$ . Then  $\widehat{G}'' = \widehat{G}$ .*

*Proof.* Note that  $T'$  is indeed a spanning-tree, so that  $Y'$ -splittability of a node in  $G'$  is well-defined. First, let us prove the first point. Consider a vertex  $v \in V(G_2)$ . Since  $G$  is 2-connected,  $v$  is not an articulation vertex of  $G$ , and thus all vertices of  $V(G_1) \setminus \{v\}$  are in the same connected component of  $G$  after removal of  $v$ . Since  $Y \cap E_1 = \emptyset$  holds, all these vertices are even in the same connected component  $h$  of  $G_Y^v$ .

Similarly, by definition of  $G'$ , all vertices in  $\{u, w\} \setminus \{v\}$  are in the same connected component  $h'$  of  $G'^v$ . This implies that the vertex sets of  $H_Y^v(G)$  and  $H_Y^v(G')$  are isomorphic by identifying  $h$  with  $h'$ . Using  $Y \subseteq E_2$ , we observe that  $H_Y^v(G)$  and  $H_Y^v(G')$  are isomorphic. This implies that  $v \in V(G_2)$  is  $Y$ -splittable with respect to  $G$  if and only if  $v$  is  $Y$ -splittable with respect to  $G'$ .

The proof for the second statement is identical to the proof of the second statement of Lemma 22, except that  $E_1$  and  $E_2$  as well as  $G_1$  and  $G_2$  are swapped.  $\square$

Although Lemma 23 provides a strong argument for performing its proposed reduction, it is not sufficiently safe to do so. In particular, there may still be  $Y$ -splittable vertices in  $V(G_1) \setminus \{u, w\}$  that we miss by performing this reduction. However, Lemma 24 shows that in this case the two vertices  $u$  and  $w$  of the 2-separation must also be  $Y$ -splittable and a very specific set of conditions holds.

**Lemma 24.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be non-empty. Let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  and the corresponding graphs  $G_1$  and  $G_2$  such that  $P_{u,w}(T) \subseteq E_1$  and  $Y \cap E_1 = \emptyset$  holds. Then for each vertex  $v \in V(G_1) \setminus \{u, w\}$ ,  $G_Y^v$  consists of one or two connected components. Let  $h_u$  and  $h_w$  be the components containing  $u$  and  $w$ , respectively. Moreover,  $v$  is  $Y$ -splittable with respect to  $G$  if and only if the following statements hold:*

1.  $v$  is an internal vertex of  $P_{u,w}(T)$ ,
2.  $Y = P_{u,w}^{-1}(G, T)$ ,
3.  $u$  and  $w$  are  $Y$ -splittable, and
4.  $h_u \neq h_w$

*Proof.* We start by showing that  $G_Y^v$  has at most two connected components. To this end, we claim that each vertex  $s \in V(G) \setminus \{v\}$  is connected to  $u$  or to  $w$  within  $G_Y^v$ . Since  $(T_u \cup T_w) \cap E_2$  belongs to  $G_Y^v$ , the statement is obvious for  $s \in V(G_2)$ . Otherwise, in case  $s \in V(G_1) \setminus \{u, w\}$ , the 2-connectivity of  $G$  implies that there is a path  $P \subseteq E(G^v)$  from  $s$  to  $\{u, w\}$ . Let  $P$  have minimum length among such paths, which implies  $P \subseteq E_1$ . Since  $E_1 \cap Y = \emptyset$ , even  $P \subseteq E(G_Y^v)$  holds, and the claim follows. In particular, this implies that  $H_Y^v(G)$  can have at most two components  $h_u$  and  $h_w$ , which contain  $u$  and  $w$ , respectively.

To prove necessity, assume that  $v \in V(G_1) \setminus \{u, w\}$  is  $Y$ -splittable. From  $\delta(v) \cap Y \subseteq E_1 \cap Y = \emptyset$  and  $Y \neq \emptyset$  it follows that  $Y$  is not a star centered at  $v$ . By Lemma 8,  $v$  is an articulation vertex of  $G_Y$ . By the above argument, this implies that  $h_u \neq h_w$ , as these are the only two possible components of  $G_Y^v$ . First, let us show that  $v$  is an internal vertex of  $P_{u,w}(T)$ . Lemma 9 implies  $v \in P_y(T)$  for some  $y \in Y$  and thus  $y \in E_2$  holds. Since  $v \in V(G_1) \setminus \{u, w\}$ , we must have  $P_{u,w}(T) \subseteq P_y(T)$ . Then, since  $V(P_{u,w}(T))$  are the only vertices of  $P_y(T)$  that are in  $V(G_1)$  by Lemma 21, we have  $v \in V(P_{u,w}(T)) \setminus \{u, w\}$ . Since this argument applies to any edge  $y \in Y$ , we must have  $Y \subseteq P_{u,w}^{-1}(G, T)$ . Suppose, for the sake of contradiction, that an edge  $e \in P_{u,w}^{-1}(G, T) \setminus Y$  exists. This implies that a path that connects  $u$  and  $w$  in  $G_Y^v$  exists in  $T_u \cup T_w \cup \{e\}$ . However, this contradicts our observation that  $h_u \neq h_w$  must hold. Consequently,  $Y = P_{u,w}^{-1}(G, T)$  holds. In particular, this shows that each edge  $y$  connects  $V_u$  to  $V_w$  since we have  $E_1 \cap Y = \emptyset$ .

It remains to show that  $u$  and  $w$  are  $Y$ -splittable. We only show  $u$ ; the argument for  $w$  is symmetric, with  $u$  and  $w$  swapped. First, consider  $G_Y^u$  and let  $h'_w$  be the component of  $G_Y^u$  that contains  $w$ . Since  $T_w$  is a tree, it follows that  $V_w$  is also contained in  $h'_w$ . Additionally, note that  $E_1 \setminus \delta(u)$  is connected since  $G$  is 2-connected. Since  $E_1 \cap Y = \emptyset$  and  $w$  is incident to some edges from  $E_1$ , also  $E_1 \setminus \delta(u)$  is contained in  $h'_w$ .

Due to  $Y = P_{u,w}^{-1}(G, T)$ , the edges connecting  $V_u$  and  $V_w$  in  $E_2$  all belong to  $Y$ . Then, since we remove  $u$  in  $G_Y^u$  and  $\{u, w\}$  are separating vertices, there are no paths in  $G_Y^u$  connecting any vertex  $s \in V_u \setminus \{u\}$  to any vertex  $t \in V_w$ , which implies that each vertex from  $V_u \setminus \{u\}$  lies in some connected component different from  $h'_w$ . This, together with the observation that each  $Y$ -edge is incident to a vertex from  $h'_w$ , implies that  $H_Y^u$  is a star centered at  $h'_w$ , and thus bipartite.

To show sufficiency, assume that  $u$  and  $w$  are  $Y$ -splittable vertices such that  $v$  is an internal vertex of  $P_{u,w}(T)$ ,  $Y = P_{u,w}^{-1}(G, T)$  and  $h_u \neq h_w$  hold. Note that  $V_u$  is contained in  $h_u$  and  $V_w$  is contained in  $h_w$ . Due to  $Y = P_{u,w}^{-1}(G, T)$ , every edge in  $Y$  connects  $V_u$  to  $V_w$ , which implies that every edge  $y \in Y$  connects  $h_u$  to  $h_w$ . Since  $V(H_Y^v) = \{h_u, h_w\}$  holds,  $\{h_u\}$  and  $\{h_w\}$  are a bipartition of  $H_Y^v$ . Thus,  $v$  is  $Y$ -splittable.  $\square$

Note that since  $\{u, w\} \subseteq V(G_2)$ , the reduction proposed in Lemma 23 is also valid for  $u$  and  $w$  and thus  $u$  and  $w$  are  $Y$ -splittable in  $G$  if and only if  $u$  and  $w$  are  $Y'$ -splittable in  $G'$ , where  $G'$  is the graph formed by the reduction given in Lemma 23. As Lemma 24 shows that a vertex  $V(G_1) \setminus \{u, w\}$  is  $Y$ -splittable only if  $u$  and  $w$  are  $Y$ -splittable, this shows that the reduction proposed in Lemma 23 is indeed safe and valid, as we then always guarantee that  $G'$  has a  $Y$ -splittable vertex if  $G$  has a  $Y$ -splittable vertex.

An important addition to Lemmas 22 and 23 is that ‘smaller’ 2-separations can always be reduced as well. This later allows us to focus our algorithms on the leaves of the SPQR tree.

**Proposition 25.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$ . Let  $Y \subseteq E(G) \setminus T$  be non-empty. Let  $E_1, E_2$  be a 2-separation of  $G$  with  $E_2 \cap Y = \emptyset$ . Let  $E'_1, E'_2$  be a 2-separation of  $G$  with  $E'_2 \subset E_2$ . Then  $E'_2 \cap Y = \emptyset$  holds.*

*Proof.* Since  $E_2' \subset E_2$  and  $E_2 \cap Y = \emptyset$ ,  $E_2' \cap Y = \emptyset$  holds.  $\square$

Note in particular that Proposition 25 implies that one of Lemmas 22 and 23 can be applied to reduce the 2-separation  $E_1, E_2'$ .

Lemmas 22 and 23 show that if one side of a 2-separation contains no  $Y$ -edges, then we can replace it by a single edge that is not in  $Y$ . Lemma 26 does something similar in the setting where the  $Y$ -edges on one side constitute a particular cut of that side; in this case, one can replace this side of the 2-separation by a single edge that is then added to  $Y$ , which intuitively represents this cut.

**Lemma 26.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be non-empty. Let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  and the corresponding graphs  $G_1$  and  $G_2$  such that  $P_{u,w}(T) \subseteq E_1$ ,  $Y \cap E_1 \neq \emptyset$  and  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$  hold. Let  $G'$  be  $G_1$  augmented by a new edge  $e = \{u, w\}$  and define  $Y' := (Y \cap E_1) \cup \{e\}$ . Then, the following hold:*

1. *A vertex  $v \in V(G)$  is  $Y$ -splittable with respect to  $G$  if and only if  $v \in V(G_1)$  and  $v$  is  $Y'$ -splittable with respect to  $G'$ .*
2. *For any  $v \in V(G_1)$  that is  $Y$ -splittable, let  $\widehat{G}$  be the graph obtained by applying Lemma 11 to  $v$ . Let  $\widehat{G}'$  be the graph obtained by applying Lemma 11 to  $v$  in  $G'$ , and let  $\widehat{G}''$  be the graph obtained by replacing  $e$  with  $G_2$  in  $\widehat{G}'$ . Then,  $\widehat{G}'' = \widehat{G}$ .*

*Proof.* First, consider a vertex  $v \in V(G_2) \setminus V(G_1)$  and let  $y \in Y \cap E_1$ . By Lemma 21,  $P_y(T) \subseteq E_1$ , and by Lemma 9, all  $Y$ -splittable vertices lie on  $P_y(T)$ . We conclude that  $v$  is not  $Y$ -splittable.

Second, observe that  $P_{u,w}^{-1}(G, T) \cap E_2$  is the set of edges from  $G_2$  that connect  $V_u$  with  $V_w$ . Also observe that the set is non-empty since otherwise  $u$  or  $w$  would be an articulation vertex of  $G$ , contradicting 2-connectivity of  $G$ . Note that by the assumption from the lemma, this set is equal to  $Y \cap E_2$ .

Third, consider a vertex  $v \in V(G_1) \setminus V(G_2)$ . Since both trees  $T_u$  and  $T_w$  are part of  $G_Y^v$ , the connected components  $h_u$  and  $h_w$  that contain  $u$  and  $w$ , respectively, cover all vertices in  $V(G_2)$ . These components are connected by an edge in  $Y \cap E_2$ . Moreover, by the arguments above, all edges in  $Y \cap E_2$  connect  $h_u$  with  $h_w$ . Similarly, the connected components  $h'_u$  and  $h'_w$  of  $G_{Y'}^v$  that contain  $u$  and  $w$ , respectively, are connected by the new edge  $e \in Y'$ . All other connected components of  $G_Y^v$  and  $G_{Y'}^v$  are identical since they are disjoint from  $V(G_2)$ . This shows that  $H_Y^v(G)$  and  $H_{Y'}^v(G')$  are isomorphic by identifying  $h_u$  with  $h'_u$  and  $h_w$  with  $h'_w$ . We conclude that  $v$  is  $Y$ -splittable with respect to  $G$  if and only if  $v$  is  $Y'$ -splittable with respect to  $G'$ .

Fourth, consider the case  $v = u$ ; the only remaining case  $v = w$  is similar. Again, the tree  $T_w$  is part of  $G_Y^v$  and hence the connected component  $h_w$  that contains  $w$  covers  $V_w$ . Let  $H_u$  denote the set of all connected components of  $G_Y^v$  that contain vertices from  $V_u \setminus \{u\}$ . Each such component  $h_u \in H_u$  contains only vertices from  $V_u \setminus \{u\}$  and is not disconnected just by removal of  $v = u$  since otherwise the latter would be an articulation vertex. Hence, there exists an edge  $y' \in Y$  that connects  $h_u$  to some other component  $h$ . Since  $y'$  must lie in  $P_{u,w}^{-1}(G, T) \cap E_2$ , we must have  $h = h_w$ . Similarly, let  $h'_w$  denote the connected component of  $G_{Y'}^v$  that contains  $w$ . All other connected components of  $G_Y^v$  and  $G_{Y'}^v$  are identical since they are disjoint from  $V(G_2)$ . This shows that  $H_{Y'}^v(G')$  and the graph obtained from  $H_Y^v(G)$  by removing vertices  $H_u$  are isomorphic by identifying  $h_w$  with  $h'_w$ . Since all vertices  $h_u \in H_u$  of  $H_Y^v(G)$  have degree 1, this implies that  $H_Y^v(G)$  is bipartite if and only if  $H_{Y'}^v(G')$  is bipartite. We conclude that  $v$  is  $Y$ -splittable with respect to  $G$  if and only if  $v$  is  $Y'$ -splittable with respect to  $G'$ .

Let us now prove the second statement. First, note that  $\widehat{G}''$  and  $\widehat{G}$  have the same edge set and the same number of vertices. Thus, it suffices to show that the incident edges for each vertex are equal. For both  $\widehat{G}''$  and  $\widehat{G}$  we obtained the graph from  $G$  by performing operations that only change the end-vertices of edges incident to  $v$  and distribute these edges over  $v_1$  and  $v_2$ . In particular, replacing  $E_2$  by  $e$  and replacing  $e$  by  $E_2$  again does not change the edges incident to vertices  $V(G_2) \setminus \{u, w\}$ . Thus, it suffices to check for the vertices  $v_1$  and  $v_2$  obtained from the neighborhood split of  $v$  in  $G$ , and  $v'_1$  and  $v'_2$  obtained from the neighborhood split of  $v$  in  $G'$  that  $\delta_{\widehat{G}}(v_1) = \delta_{\widehat{G}''}(v'_1)$  holds.

Because  $G$  and  $G'$  are 2-connected,  $H_Y^v(G)$  and  $H_{Y'}^v(G')$  both have a unique bipartition by Corollary 19, which we denote by  $(I, J)$  and  $(I', J')$  respectively. In particular,  $I$  and  $I'$  both contain the same connected components (after identification) of  $G_Y^v$  and  $G_{Y'}^v$ . Moreover, we have shown already that  $I$  and  $I'$  contain exactly the same vertices of  $V(G_1)$ , such that  $\delta_{\widehat{G}}(v_1) = \delta_G^I(v)$  and  $\delta_{\widehat{G}'}(v'_1) = \delta_{G'}^{I'}(v)$  hold. Since  $E_1$  and  $E_2$  partition the edges of the graph,  $\delta_G^I(v) \cap E_1$  and  $\delta_G^I(v) \cap E_2$  form a partition of  $\delta_G^I(v)$ .

Consider  $\delta_G^I(v) \cap E_1$ . Then,  $\delta_G^I(v) \cap E_1 = \delta_{G'}^{I'}(v) \cap E_1$  holds, because  $I$  and  $I'$  contain the same vertices of  $V(G_1)$ . By applying Definition 10, we have that  $\delta_{G'}^{I'}(v) \cap E_1 = \delta_{\widehat{G}'}^{I'}(v'_1) \cap E_1$ , which is equal to  $\delta_{\widehat{G}''}^{I'}(v'_1) \cap E_1$  because replacing  $e$  by  $E_2$  does not affect the edges of  $E_1$ . Thus, we have  $\delta_G^I(v) \cap E_1 = \delta_{\widehat{G}''}^{I'}(v'_1) \cap E_1$ .

Secondly, we consider  $\delta_G^I(v) \cap E_2$ . First, note that if  $v \notin \{u, w\}$ , that  $\delta_G^I(v) \cap E_2 = \emptyset = \delta_{\widehat{G}''}^{I'}(v'_1)$  holds since then  $v$  is not incident to  $E_2$  or to  $e$  in any of the graphs. Next, consider the case  $v \in \{u, w\}$ , and let  $v' = \{u, w\} \setminus \{v\}$ . First, since  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$ , any edge connecting  $v$  to  $V_{v'}$  must be in  $Y$ . Then,  $V_{v'}$  is contained in one connected component  $h$ , since  $T_{v'}$  connects all these vertices. Because  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$ ,  $h$  must be connected in  $H_Y^v(G)$  to any component  $h_v \in H_v$  where  $h_v$  contains vertices in  $V_v$ .

Since  $H_Y^v$  is bipartite, this shows that these edges must be on opposite sides. By,  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$ , there can exist no edges in  $Y$  connecting  $v$  to any node in  $V_v$ . Thus, in  $E_2$ ,  $v$  can only connect to nodes

in  $V_v$  using edges that are not in  $Y$  and  $v$  can only connect to nodes in  $V_{v'}$  using edges in  $Y$ , and all vertices in  $V_v$  and  $V_{v'}$  are on opposite sides of the bipartition  $(I, J)$ . Thus, we either have  $\delta_G(v) \cap E_2 \subseteq \delta_G^I(v)$  or  $\delta_G(v) \cap E_2 \subseteq \delta_G^J(v)$ . From this, it follows that either  $\delta_G^I(v) \cap E_2 = \delta_G(v) \cap E_2$  or  $\delta_G^I(v) \cap E_2 = \emptyset$  holds.

Because we defined  $I$  and  $I'$  to have similar vertices, we have  $e \in \delta_{G'}^{I'}(v)$  if and only if  $\delta_G(v) \cap E_2 = \delta_G^I(v) \cap E_2$ . We observe that if  $e \in \delta_{G'}^{I'}(v)$  holds, then  $e \in \delta_{\widehat{G}''}(v_1')$  holds too. Then, in  $\widehat{G}''$ , all edges from  $\delta_G(v) \cap E_2$  are exactly placed incident to  $v_1'$ , i.e.,  $\delta_G(v) \cap E_2 = \delta_{\widehat{G}''}(v_1')$  holds. Then, by the above, it follows that  $\delta_G^I(v) \cap E_2 = \delta_{\widehat{G}''}(v_1')$ . Similarly, if  $e \notin \delta_{G'}^{I'}(v)$  then  $\delta_G^I(v) \cap E_2 = \emptyset = \delta_{\widehat{G}''}(v_1') \cap E_2$  holds because all edges are placed incident to  $v_2'$ , instead.

We obtain  $\delta_{\widehat{G}}(v_1) = \delta_G^I(v) = (\delta_G^I(v) \cap E_1) \cup (\delta_G^I(v) \cap E_2) = (\delta_{\widehat{G}''}(v_1') \cap E_1) \cup (\delta_{\widehat{G}''}(v_1') \cap E_2) = \delta_{\widehat{G}''}(v_1')$ , which shows that  $\widehat{G} = \widehat{G}''$  holds.  $\square$

Although we argued that all three reductions are valid on the graph, it is not immediately clear that they are also applicable to the SPQR tree. However, the important conditions for the reductions that either require  $E_1 \cap Y = \emptyset$  or  $E_2 \cap Y = \emptyset$  (Lemmas 22 and 23) or require that  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$  (Lemma 26), are invariant under performing reversals at the 2-separations given by  $u$  and  $w$ . Thus, we can also apply them to the SPQR tree.

It is worth to mention that it is in general nontrivial to check the condition  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$  of Lemma 26. However, in the case where  $G_2$  is a component of type (S), (P) or (Q) this is not a problem, as then it simply amounts to checking whether  $Y = E \setminus T$ . For nodes of type (R) we show that this can be done by testing for splittability.

**Lemma 27.** *Let  $G$  be a simple 3-connected graph with spanning tree  $T$ , let  $Y \subseteq E(G) \setminus T$  be non-empty, and let  $\{u, w\} \in T$  be a tree edge. Then  $Y = P_{u,w}^{-1}(G, T)$  holds if and only if  $u$  and  $w$  are both  $Y$ -splittable.*

*Proof.* We first show necessity. Let  $T_u$  and  $T_w$  be the subtrees of  $T \setminus \{\{u, w\}\}$  containing  $u$  and  $w$ , respectively, and let  $V_u$  and  $V_w$  be their respective sets of vertices. Due to  $Y \neq \emptyset$  there is an edge  $y \in Y$ . Since  $G$  is simple,  $y \neq \{u, w\}$  holds, so there exists a vertex  $v \in V(G) \setminus \{u, w\}$  with  $y \in \delta(v)$ . Our assumption  $Y = P_{u,w}^{-1}(G, T)$  implies that the only edge from  $E \setminus Y$  that connects  $V_u$  with  $V_w$  is  $\{u, w\}$ . Let  $h_w$  denote the connected component of  $G_Y^u$  consisting of the vertices  $V_w$  and let  $H$  denote the set of all other components, whose vertices form a partition of  $V_u \setminus \{u\}$ . Now, consider any edge  $y' \in Y \setminus \delta(u)$ . Since  $\{u, w\} \in P_{y'}(T)$  holds, the two end-vertices of  $y'$  belong to both  $V_u$  and  $V_w$ , which means that they belong to  $h_w$  and to one of the components in  $H$ . This proves that  $H$  and  $\{h_w\}$  form a bipartition of  $H_Y^u$ , so we conclude that  $u$  is  $Y$ -splittable. The proof that  $w$  is  $Y$ -splittable is similar.

To show sufficiency, let  $u$  and  $w$  be  $Y$ -splittable for  $(G, T)$ . By Lemma 9, both  $u$  and  $w$  must lie on the intersection of the paths  $P_y$  for all  $y \in Y$  (which is again a path due to  $Y \neq \emptyset$ ). This implies  $Y \subseteq P_{u,w}^{-1}(G, T)$ . Assume that there exists an edge  $e \in P_{u,w}^{-1}(G, T) \setminus Y$ , i.e.,  $\{u, w\} \in P_e(T)$  and  $e \notin Y$  hold. Since  $G$  is simple,  $e \neq \{u, w\}$  holds, which implies that at least one of  $G_Y^u$  or  $G_Y^w$  must contain  $e$ . Without loss of generality we assume that this is the case for  $G_Y^u$  since the argument for  $G_Y^w$  is similar.

Now consider the connected components of  $G_Y^u$  and let  $h_e$  be the connected component containing edge  $e$ . Note that since  $h_e$  is connected to  $T_w$ ,  $T_w$  is also contained in  $h_e$ . Since  $T$  is connected and disjoint from  $Y$ , every connected component of  $G_Y^u$  must be connected to  $u$  by some tree edge in  $T$ . Denote the set of connected components other than  $h_e$  by  $H := V(H_Y^u) \setminus \{h_e\}$ . In particular, this implies that for each  $h \in H$ ,  $V(h) \subseteq V_u$ . Since  $e$  is not incident to  $u$ , there must exist some  $u' \in V_u \setminus \{u\}$  to which  $e$  is incident. Let  $G_Y^{u,w}$  be the graph  $G$  with vertex set  $V(G) \setminus \{u, w\}$  and edge set  $E(G) \setminus (\delta(u) \cup \delta(w) \cup Y)$ , and denote by  $V_{u'}$  and  $E_{u'}$  the vertices and edges of its connected component containing  $u'$ .

Then  $|\delta(u) \cap \delta(V_{u'})| \geq 1$  holds since the set must contain a tree edge connecting to  $V_{u'}$  by the above argumentation. Similarly,  $|\delta(w) \cap \delta(V_{u'})| \geq 1$  holds: if there is some vertex  $w' \in V_w \setminus \{w\}$  contained in  $V_{u'}$ , then there must exist a tree edge by the above reasoning. Otherwise,  $e = \{u', w\}$  must hold since  $w$  is the only vertex in  $T_w$  within  $E_{u'}$ , which shows that  $e \in \delta(w) \cap \delta(V_{u'})$ . In the following, we will argue that  $E_1 := E_{u'} \cup (\delta(u) \cap \delta(V_{u'})) \cup (\delta(w) \cap \delta(V_{u'}))$  and  $E_2 := E \setminus E_1$  form a 2-separation of  $G$  with separating vertices  $u$  and  $w$ , which contradicts 3-connectivity of  $G$ . Note that  $|\delta(u) \cap \delta(V_{u'})| \geq 1$  and  $|\delta(w) \cap \delta(V_{u'})| \geq 1$  together imply  $|E_1| \geq 2$ .

Next, observe that  $E_{u'} \subseteq E[h_e]$  holds since  $G_Y^{u,w}$  is a subgraph of  $G_Y^u$ . Since  $u$  is  $Y$ -splittable,  $h_e$  has no self-loops. In particular, this implies there is no edge in  $Y$  that connects  $V_{u'}$  with  $T_w$  (and thus with  $V_w$ ). Since  $w$  is also  $Y$ -splittable, the component containing  $T_u$  in  $H_Y^w$  has no self-loops and hence there are also no edges in  $Y$  that connect  $V_{u'}$  to  $V_u$  (since  $V_{u'} \subseteq V_u$ ). Note that  $\delta(u) \cap \delta(V_{u'})$  and  $\delta(w) \cap \delta(V_{u'})$  connect  $E_{u'}$  to  $u$  and  $w$ , and that by the above argumentation there are no edges in  $Y$  incident to any vertex from  $V_{u'}$  since  $V_u \cup V_w = V$  spans all possible vertices. Additionally, since we defined  $E_{u'}$  as a connected component of  $G_Y^{u,w}$ , there can be no other edges connecting  $V_{u'}$  to vertices in  $V \setminus (V_{u'} \cup \{u, w\})$ . Thus, we can then conclude that  $E_1$  and  $E_2$  are only adjacent in  $u$  and  $w$ .

In the case where  $E_{u'}$  is the only connected component of  $G_Y^{u,w}$ ,  $h_e$  is the only connected component of  $G_Y^u$ . Similarly,  $G_Y^w$  also consists of a single connected component. Since  $u$  and  $w$  are both  $Y$ -splittable, these components cannot have self-loops. This implies that  $Y \setminus \delta(u) = \emptyset$  and  $Y \setminus \delta(w) = \emptyset$ . As  $G$  is simple, there is no edge in  $y \in Y$ , with  $y = \{u, w\}$ , which contradicts the assumption  $Y \neq \emptyset$ .

In the alternative case,  $G_Y^{u,w}$  has more than one connected component, which implies that there exists some vertex  $q \in V \setminus (V_{u'} \cup \{u, w\})$ . Now 3-connectivity of  $G$  implies  $|\delta(q)| \geq 2$ , and  $\delta(q) \subseteq E_2$  implies  $|E_2| \geq 2$ , which in turn implies that  $E_1$  and  $E_2$  form a 2-separation of  $G$ . This contradicts 3-connectivity

of  $G$ . We conclude that there exists no edge  $e \in P_{u,w}^{-1}(G, T) \setminus Y$ , and consequently that  $Y = P_{u,w}^{-1}(G, T)$  holds. This completes the proof.  $\square$

The application of Lemma 27 is primarily in that we can use it to efficiently check the condition of Lemma 26 for SPQR tree nodes of type (R) (where  $G = G_2$  as stated in Lemma 26). Lemma 27 hints at an important fact; rather than figuring out the intersection of all paths over  $G$ , it is sufficient to check splittability of the vertices of the smaller graph  $G_2$  augmented by a single edge. In fact, we will later see that this intuition holds more generally, and that we only need to test for splittability of vertices in the skeletons of the SPQR tree, rather than on one of the represented graphs.

Although we have argued that the reduction is valid, it is still difficult to apply to 2-separations that are not leaves of the SPQR tree. The following lemma shows that the reduction from Lemma 26 is applicable only if every other 2-separation contained in it is also reducible. This shows that it is sufficient to only consider leaf nodes of the reduced SPQR tree to find all reductions.

**Lemma 28.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be non-empty. Let  $E_1, E_2$  be a 2-separation of  $G$  with  $V(E_1) \cap V(E_2) = \{u, w\}$  and  $P_{u,w}^{-1}(T) \subseteq E_1$ . Let  $E'_1, E'_2$  be another 2-separation of  $G$  with  $V(E'_1) \cap V(E'_2) = \{u', w'\}$  and  $E'_2 \subset E_2$ . Then  $Y \cap E_2 = P_{u,w}^{-1}(G, T) \cap E_2$  implies that  $Y \cap E'_2 = P_{u',w'}^{-1}(G, T) \cap E'_2$  or  $Y \cap E'_2 = \emptyset$  holds.*

*Proof.* We distinguish two cases, depending on whether  $P_{u',w'}(T)$  uses edges from  $E_1$  or from  $E_2$ .

First, suppose  $P_{u,w}(T) \subseteq P_{u',w'}(T)$  holds. Consider any edge  $e \in E'_2$ . Note that  $P_{u',w'}(T) \subseteq P_e$  holds if and only if  $P_{u,w}(T) \subseteq P_e$  holds. Hence,  $e \in P_{u',w'}^{-1}(G, T)$  holds if and only if  $e \in P_{u,w}^{-1}(G, T)$  holds. From  $E'_2 \subseteq E_2$  it now follows that  $Y \cap E'_2 = P_{u,w}^{-1}(G, T) \cap E'_2 = P_{u',w'}^{-1}(G, T) \cap E'_2$  holds.

Otherwise,  $P_{u,w}(T) \not\subseteq P_{u',w'}(T)$  implies  $P_{u',w'}(T) \subseteq E_2$ . Lemma 21 implies that for each edge  $e \in E_2 \setminus T \subseteq E_2 \setminus T$  we have  $P_e(T) \subseteq E_2$  and thus  $P_{u,w}(T) \not\subseteq P_e(T)$ . This implies  $\emptyset = P_{u,w}^{-1}(G, T) \cap E'_2 = Y \cap E'_2$ .  $\square$

In some sense, SPQR trees already naturally encode some reductions using the virtual edge pairs. Consider a node  $\mu$  of an SPQR tree  $\mathcal{T}$  and let  $e$  be a virtual edge of  $\mu$  that connects  $\mu$  to some subtree  $\mathcal{T}_S$  of  $\mathcal{T}$ . In the reductions we replace one side of the 2-separation with a single edge; in the SPQR tree this corresponds to replacing any realization of  $\mathcal{T}_S$  by  $e$ , which is equivalent to setting  $\mathcal{T}_R = \mathcal{T} \setminus \mathcal{T}_S$  and making  $e$  non-virtual. Additionally, in case the reduction from Lemma 26 is used, we add  $e$  to  $Y$  and remove any edges in  $\mathcal{T}_S$  from  $Y$ . Then  $e$  is virtual in  $\mathcal{T}$ , but has become a regular edge in  $\mathcal{T}_R$ . In our algorithm, we iteratively consider the leaves of  $\mathcal{T}_R$ , as Lemma 28 and Proposition 25 show that larger subtrees can only be reduced if all 2-separations in the subtree can be reduced as well. Thus, if a leaf node cannot be reduced, we can not reduce a larger subtree containing it, and thus it suffices to focus on the leaf nodes.

Because local 2-separations do not have an associated virtual edge pair, we instead replace 2 or more regular edges that define a reducible local 2-separation by a single edge. For a local 2-separation of a node  $\mu \in \mathcal{V}(\mathcal{T})$ , we can also view this as splitting  $\mu$  into two members (of the same type as  $\mu$ ), dividing the edges based on the 2-separation, adding a virtual edge pair, and then reducing the 2-separation using one of the appropriate results.

First, let us define a  $Y$ -reduced SPQR tree, which is simply an SPQR tree that has been fully reduced. We show a few properties that must hold, regardless of the algorithm used to derive the reduced SPQR tree.

**Definition 29** ( $Y$ -reduced SPQR tree). *Let  $\mathcal{T}$  be an SPQR tree. We say that  $\mathcal{T}$  is  $Y$ -reduced if  $\emptyset \neq Y \subseteq E^{\text{reg}}(\mathcal{T})$  holds and if either  $\mathcal{T}$  consists of a single cycle skeleton or Lemmas 22, 23 and 26 are not applicable to any 2-separation of any graph represented by  $\mathcal{T}$ .*

Note that Definition 29 has one notable exception, which is that if  $\mathcal{T}$  consists of a single cycle skeleton, that not necessarily all 2-separations are irreducible. We now present elementary properties of  $Y$ -reduced SPQR trees.

**Proposition 30.** *Every  $Y$ -reduced SPQR tree  $\mathcal{T}$  satisfies these properties:*

1. Each leaf  $\mu$  of  $\mathcal{T}$  satisfies  $E_\mu \cap Y \neq \emptyset$ .
2. If  $|\mathcal{V}(\mathcal{T})| \geq 2$ , no leaf  $\mu$  of  $\mathcal{T}$  is of type (S).
3. If  $|\mathcal{V}(\mathcal{T})| \geq 2$ , then each  $\mu \in \mathcal{V}(\mathcal{T})$  of type (S) satisfies  $|E_\mu^{\text{reg}}(\mathcal{T})| \leq 1$ .

*Proof.* The first statement follows from  $Y \neq \emptyset$  in case  $|\mathcal{V}(\mathcal{T})| = 1$  holds. Otherwise, a leaf  $\mu \in \mathcal{V}(\mathcal{T})$  with  $E_\mu \cap Y = \emptyset$  cannot exist since then one of Lemmas 22 to 24 would be applicable, in contradiction to Definition 29.

For the second statement, consider a leaf  $\mu$  of type (S) and assume  $|\mathcal{V}(\mathcal{T})| \geq 2$ . By the first statement, we have  $E_\mu \cap Y \neq \emptyset$ . Since  $|T_\mu| = |E_\mu| - 1$  holds for nodes of type (S), the single edge  $e \in E_\mu \setminus T_\mu$  lies in  $Y$ , which implies that Lemma 26 is applicable. Again this contradicts Definition 29.

For the third statement, assume for the sake of contradiction, that there exist at least two regular edges  $e_1, e_2 \in E_\mu^{\text{reg}}(\mathcal{T})$ . Clearly,  $\{e_1, e_2\}$  and  $E' := E_\mu^{\text{reg}}(\mathcal{T}) \setminus \{e_1, e_2\}$  form a local 2-separation (of  $\mu$ ), where  $|E'| \geq |E_\mu^{\text{virt}}(\mathcal{T})| \geq 2$  holds because  $\mu$  is, by the second statement, not a leaf. We now distinguish two cases. If  $e_1, e_2 \in T_\mu$  holds, then  $e_1, e_2 \notin Y$  implies that Lemmas 23 and 24 are applicable. Otherwise,



we can assume without loss of generality that  $e_1 \in T_\mu$  and  $e_2 \notin T_\mu$  hold. Then Lemma 22 or Lemma 26 is applicable, depending on whether  $e_2 \notin Y$  or  $e_2 \in Y$  holds. In all cases, applicability of the reductions contradicts Definition 29.  $\square$

Let us now present the complete reduction algorithm **REDUCETREE**, which performs all possible reductions on  $\mathcal{T}$ . It first tries to apply the reductions from Lemmas 22, 23 and 26 to leaf skeletons of the SPQR tree. Then it performs the reductions given by **REDUCESERIES** and **REDUCEPARALLEL**, which remove the local 2-separations contained in (S) and (P) nodes. Fig. 7 shows an example of a run of **REDUCETREE**.

The procedures for removing local 2-separations are given in **REDUCESERIES** and **REDUCEPARALLEL**. In particular, if  $|Y_\mu| \geq 2$  for an SPQR tree node  $\mu$  of type (S) or (P), then  $Y_\mu$  can be replaced by a single edge using Lemma 26. Similar reductions can be made using Lemmas 22 and 23 if  $|E_\mu \setminus (E^{\text{reg}}(\mathcal{T}_R) \cup Y)| \geq 2$  holds.

---

**Algorithm 1:** Removing local 2-separations within (P)-nodes

---

**Algorithm:** **REDUCEPARALLEL**( $\mu, \mathcal{T}_R, Y_R$ )

**Input:** SPQR tree node  $\mu$  of type (P), partially reduced SPQR tree  $\mathcal{T}_R$ , marked edges  $Y_R$

**Output:** further reduced SPQR tree  $\mathcal{T}_R$ , marked edges  $Y_R$

- 1 Let  $Y_\mu := Y_R \cap E_\mu$ .
  - 2 **if**  $|Y_\mu| > 1$  **then** replace  $Y_\mu$  by a new edge  $e$  in  $\mathcal{T}_R$  and update  $Y_R := (Y_R \cup \{e\}) \setminus Y_\mu$ .
  - 3 Let  $Z := E_\mu \setminus (E^{\text{virt}}(\mathcal{T}_R) \cup Y_R)$ .
  - 4 **if**  $|Z| > 1$  **then**
  - 5 replace  $Z$  by  $f$  in  $\mathcal{T}_R$ .
  - 6 **if**  $T_\mu \cap Z \neq \emptyset$  **then** Update  $T_\mu := (T_\mu \cup \{f\}) \setminus Z$ .
  - 7 **end**
  - 8 **if**  $E^{\text{virt}}(\mathcal{T}_R) = \emptyset$  **then** change  $\mu$  to type (Q).
  - 9 **return** ( $\mathcal{T}_R, Y_R$ )
- 

---

**Algorithm 2:** Removing local 2-separations within (S)-nodes

---

**Algorithm:** **REDUCESERIES**( $\mu, \mathcal{T}_R, Y_R$ )

**Input:** SPQR tree node  $\mu$  of type (S), partially reduced SPQR tree  $\mathcal{T}_R$ , marked edges  $Y_R$

**Output:** further reduced SPQR tree  $\mathcal{T}_R$ , marked edges  $Y_R$

- 1 Let  $Z := E_\mu \setminus E^{\text{virt}}(\mathcal{T}_R)$ .
  - 2 **if**  $|Z| \neq |E_\mu|$  and  $|Z| > 1$  **then**
  - 3 Replace  $Z$  by a new edge  $e$  in  $\mathcal{T}_R$ .
  - 4 Let  $Y_\mu := Z \cap Y_R$ .
  - 5 **if**  $Y_\mu \neq \emptyset$  **then**
  - 6  $Y_R := Y_R \cup \{e\} \setminus Y_\mu$
  - 7 **end**
  - 8 **if**  $Z \subseteq T_\mu$  **then** Update  $T_\mu := (T_\mu \cup \{e\}) \setminus Z$ .
  - 9 **else** Update  $T_\mu := T_\mu \setminus Z$ .
  - 10 **end**
  - 11 **return** ( $\mathcal{T}_R, Y_R$ )
-

---

**Algorithm 3:** Compute the reduced minimal SPQR tree  $\mathcal{T}_R$  of a minimal SPQR tree  $\mathcal{T}$ .

---

**Algorithm:** REDUCETREE( $\mathcal{T}, Y$ )

**Input:** Minimal SPQR tree  $\mathcal{T}$ , edges  $Y$

**Output:** Reduced SPQR tree  $\mathcal{T}_R$ , edges  $Y_R$

```

1 Let  $\mathcal{T}_R := \mathcal{T}$  and  $Y_R := Y$ .
2 Let  $\mathcal{L} \subseteq \mathcal{V}$  be a list of all leaves of  $\mathcal{T}_R$ .
3 Let  $\mathcal{L}_R := \emptyset$ .
4 while  $\mathcal{L} \neq \emptyset$  and  $|\mathcal{V}(\mathcal{T}_R)| \geq 2$  do                                     // Reductions for Lemmas 22 and 23
5   | Let  $\mu \in \mathcal{L}$  be an arbitrary leaf of  $\mathcal{T}_R$ .
6   | Let  $\nu$  be the unique neighbor of  $\mu$  in  $\mathcal{T}_R$ .
7   | Update  $\mathcal{L} := \mathcal{L} \setminus \{\mu\}$ .
8   | Let  $e \in E(G_\mu)$  and  $f \in E(G_\nu)$  be the virtual edge pair connecting  $\mu$  and  $\nu$ .
9   | if  $E_\mu \cap Y_R = \emptyset$  then
10  |   | Remove  $\mu$  from  $\mathcal{T}_R$ .
11  |   | Mark  $f$  as a non-virtual edge.
12  |   | if  $\nu$  is a leaf of  $\mathcal{T}_R$  then Update  $\mathcal{L} := \mathcal{L} \cup \{\nu\}$ .
13  | else
14  |   | Update  $\mathcal{L}_R := \mathcal{L}_R \cup \{\mu\}$ .
15  | end
16 end
17 while  $\mathcal{L}_R \neq \emptyset$  and  $|\mathcal{V}(\mathcal{T}_R)| \geq 2$  do                                     // Reductions for Lemma 26
18  | Let  $\mu \in \mathcal{L}_R$  be an arbitrary leaf of  $\mathcal{T}_R$ .
19  | Let  $\nu$  be the unique neighbor of  $\mu$  in  $\mathcal{T}_R$ .
20  | Update  $\mathcal{L}_R := \mathcal{L}_R \setminus \{\mu\}$ 
21  | Let  $e \in E(G_\mu)$  and  $f \in E(G_\nu)$  be the virtual edge pair connecting  $\mu$  and  $\nu$ .
22  | if  $e \in T_\mu$  and  $E_\mu \cap Y_R = P_e^{-1}(G_\mu, T_\mu)$  then
23  |   | Remove  $\mu$  from  $\mathcal{T}_R$ .
24  |   | Mark  $f$  as a non-virtual edge.
25  |   | if  $\nu$  is a leaf of  $\mathcal{T}_R$  then Update  $\mathcal{L}_R := \mathcal{L}_R \cup \{\nu\}$ .
26  |   | Update  $Y_R := (Y_R \cup \{f\}) \setminus E_\mu$ .
27  | end
28 end
29 for  $\mu \in \mathcal{V}(\mathcal{T}_R)$  do                                                         // Reductions for Lemmas 22, 23 and 26 within skeletons
30  | if  $\mu$  is of type (S) then
31  |   | Update  $(\mathcal{T}_R, Y_R) := \text{REDUCESERIES}(\mu, \mathcal{T}_R, Y_R)$ .
32  | else if  $\mu$  is of type (P) then
33  |   | Update  $(\mathcal{T}_R, Y_R) := \text{REDUCEPARALLEL}(\mu, \mathcal{T}_R, Y_R)$ .
34  | end
35 end
36 return  $(\mathcal{T}_R, Y_R)$ 

```

---

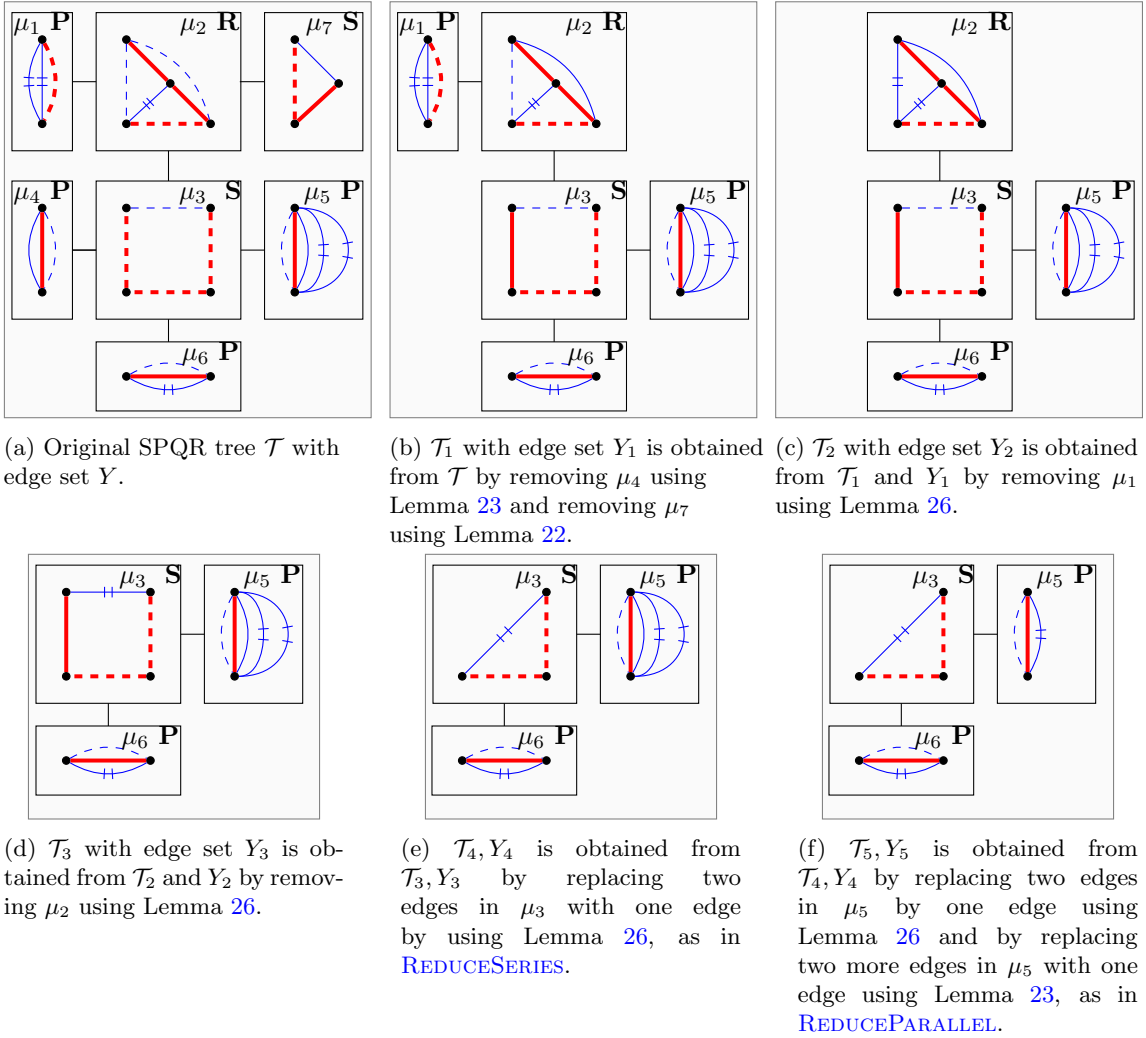


Figure 7: A sample run of REDUCETREE, applying the reductions to the SPQR tree  $\mathcal{T}$  from Fig. 5 for a particular set  $Y$ . SPQR trees  $\mathcal{T}_1, \dots, \mathcal{T}_5$ , along with updated edge sets  $Y_1, \dots, Y_5$  are derived from  $\mathcal{T}$  by repeatedly applying reductions to  $\mathcal{T}$ . Virtual edges in the SPQR tree are given by dashed edges, tree edges are marked in bold and red and all other edges are marked in blue. Edges in  $Y, Y_1, \dots, Y_5$  are indicated by two stripes.

In order to show that REDUCETREE outputs an SPQR tree, it is crucial that we show that the node labels of  $\mathcal{T}_R$  are correct. Thus, we need to consider REDUCESERIES and REDUCEPARALLEL, which can modify the skeletons and node labels.

**Lemma 31.** *Let  $\mathcal{T}$  be an SPQR tree, let  $Y \subseteq E^{\text{reg}}(\mathcal{T})$  be non-empty and let  $(\mathcal{T}_R, Y_R)$  be obtained from REDUCETREE( $\mathcal{T}, Y$ ). Then each of the following holds:*

1. Each  $\mu \in \mathcal{V}(\mathcal{T}_R)$  of type (S) has  $|E_\mu| \geq 3$ .
2. Each  $\mu \in \mathcal{V}(\mathcal{T}_R)$  of type (P) has  $|E_\mu| \geq 3$ .
3. If  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$ , then  $\mu$  is not of type (P).
4. If  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$  of type (Q), then  $|E_\mu| = 2$ .

*Proof.* The only part of REDUCETREE where skeleton graphs are modified is the loop 29–35. Let  $\mathcal{T}'$  and  $Y'$  be the tree and the reduced edges before this loop. Note that  $\mathcal{T}'$  is a minimal SPQR tree since it is a subtree of  $\mathcal{T}$ . We denote by  $E_\mu^{\mathcal{T}'}$  the set of edges of the skeleton  $G_\mu$  before execution of the loop, and by  $E_\mu^{\mathcal{T}_R}$  the set of edges after execution of the loop. Since  $\mathcal{T}$  is a valid SPQR tree, we initially have  $|E_\mu^{\mathcal{T}'}| \geq 3$  for each  $\mu \in \mathcal{V}(\mathcal{T}')$  of type (S) or (P).

First, consider the case in which  $\mathcal{V}(\mathcal{T}') = \{\mu\}$  holds. If  $\mu$  is of type (S), then REDUCESERIES does not modify  $E_\mu^{\mathcal{T}'}$ , which implies  $|E_\mu^{\mathcal{T}_R}| = |E_\mu^{\mathcal{T}'}| \geq 3$ . In case  $\mu$  is of type (P), then REDUCEPARALLEL changes it to a node of type (Q). Because this is the only place in REDUCETREE where a node's type is changed, it thus cannot occur that  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$  where  $\mu$  is of type (P), which proves the third statement.

To show the fourth statement, first note that if  $\mathcal{T}$  is a single node of type (Q) that then REDUCETREE does not change it. In the other case where  $\mathcal{T}_R$  is a single node that is changed to type (Q) by REDUCEPARALLEL,

note that we must have  $E_\mu^{\text{virt}}(\mathcal{T}_R) = \emptyset$ , which implies that  $Y' \cap E_\mu^{\mathcal{T}'}$  and  $E_\mu^{\mathcal{T}'} \setminus Y'$  partition the edges of  $E_\mu$ . Since  $T_\mu \neq \emptyset$ , we have  $E_\mu^{\mathcal{T}'} \setminus Y' \neq \emptyset$  is non-empty, and since the reductions preserve existence of edges in  $Y$ , it follows that  $Y' \cap E_\mu^{\mathcal{T}'} \neq \emptyset$ . Since both of these two sets are replaced by exactly one edge, we have  $|E_\mu^{\mathcal{T}_R}| = 2$ , showing the fourth statement.

Second, consider the case in which  $|\mathcal{V}(\mathcal{T}')| \geq 2$  holds and  $\mu$  is of type (S). Because Lemma 26 is applicable if  $\mu$  is a leaf of type (S),  $\mu$  must have degree at least 2, since otherwise  $\mu$  would have been removed before reaching line 29. In particular, this implies  $|E^{\text{virt}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}| \geq 2$ . Since  $\mathcal{T}'$  is a valid SPQR tree,  $|E_\mu^{\mathcal{T}'}| \geq 3$  holds. Moreover, REDUCESERIES does not remove edges in  $E^{\text{virt}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}$  and retains at least one edge  $E^{\text{reg}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}$  (if present), which shows  $|E_\mu^{\mathcal{T}_R}| \geq 3$ .

Third, consider the case in which  $|\mathcal{V}(\mathcal{T}')| \geq 2$  holds, and where  $\mu$  is a leaf of type (P), which implies  $|E_\mu^{\text{virt}}(\mathcal{T}')| = 1$ . Since  $\mu$  is a leaf of  $\mathcal{T}'$ , the set  $Y'_\mu = Y' \cap E_\mu$  must be non-empty. Additionally, we have that  $Z = E_\mu^{\mathcal{T}'} \setminus (E^{\text{virt}}(\mathcal{T}') \cup Y')$ , as defined in line 3 of REDUCEPARALLEL, is non-empty, as otherwise Lemma 26 could be applied to  $\mu$  since  $E_\mu$  contains exactly one virtual tree edge with all other edges in  $Y$ . Note that each reduction in REDUCEPARALLEL preserves exactly one edge from  $Y_\mu$  and from  $Z$ . Then,  $E_\mu^{\mathcal{T}_R}$  contains exactly one edge from each set and the single edge in  $E^{\text{virt}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}$ , which shows  $|E_\mu^{\mathcal{T}_R}| = 3$ .

Finally, consider the case where  $|\mathcal{V}(\mathcal{T}')| \geq 2$  holds, and where  $\mu$  is of type (P), but not a leaf, which implies  $|E^{\text{virt}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}| \geq 2$ . Since  $E_\mu^{\mathcal{T}'}$  has at least 3 edges, there must exist a third edge  $e'$  which lies in either  $Y'_\mu$ ,  $Z$  (as defined in line 3 of REDUCEPARALLEL) or  $E^{\text{virt}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}$ . Since REDUCEPARALLEL preserves at least one edge from  $Y'_\mu$  or  $Z$  and keeps the edges from  $E^{\text{virt}}(\mathcal{T}') \cap E_\mu^{\mathcal{T}'}$ , this shows that  $|E_\mu^{\mathcal{T}_R}| \geq 3$  holds.  $\square$

**Theorem 32.** *Let  $\mathcal{T}$  be a minimal SPQR tree,  $Y \subseteq E^{\text{reg}}(\mathcal{T})$  be non-empty and let  $(\mathcal{T}_R, Y_R)$  be obtained from REDUCETREE( $\mathcal{T}, Y$ ). Then,  $\mathcal{T}_R$  is a  $Y_R$ -reduced and minimal SPQR tree. Moreover, there exists a graph that is represented by  $\mathcal{T}$  that has a  $Y$ -splittable vertex if and only if there exists a graph that is represented by  $\mathcal{T}_R$  that has a  $Y_R$ -splittable vertex.*

*Proof.* First, observe that the loop in lines 4–16 of REDUCETREE performs the reductions described in Lemmas 22 and 23, where for the latter Lemma 24 shows that the possible existence of a  $Y_R$ -splittable vertex in  $\mathcal{T}_R$  is preserved. Additionally, after line 16 we have that every leaf  $\mu \in \mathcal{V}(\mathcal{T}_R)$  has  $E_\mu \cap Y_R \neq \emptyset$ . In particular, this also shows that after the loop exits, no 2-separations that are given by the edges of the SPQR tree can correspond to a reduction as in Lemma 22 or Lemma 23, as then both subtrees of the tree formed by removing the edge must contain an edge in  $Y_R$ , as argued by Proposition 25.

Lines 17–28 perform the reduction described in Lemma 26 on the remaining tree  $\mathcal{T}_R$ . In particular, this implies that after exiting the loop, each leaf  $\mu$  of  $\mathcal{T}_R$  cannot be reduced using Lemma 26. Note that Lemma 26 preserves the existence of a  $Y_R$ -edge in every leaf, which implies that we do not need to check Lemmas 22 and 23 again. Lemma 28 shows that it is sufficient to only consider the reduction in Lemma 26 to the leaf nodes of the SPQR tree. In particular, it shows that if a 2-separation given by an edge  $\{\mu, \nu\} \in \mathcal{T}$  is reducible using Lemma 26 then all leaves on one side of the subtree formed by removing  $\{\mu, \nu\}$  from  $\mathcal{T}$  must be reducible. Thus, once the loop terminates, no edges of the SPQR tree can be further reduced using Lemma 26.

Finally, in lines 29–35 we ensure that none of the skeletons  $G_\mu$  has local 2-separations that can be reduced by applying Lemmas 22, 23 and 26 to  $G_\mu$  itself. Note that we do not check nodes of type (R) as these are 3-connected, and nodes of type (Q) can occur only when the SPQR tree consists of a single skeleton  $G_\mu$  with  $|E_\mu| \leq 2$ , and thus has no 2-separations at all.

There is one case, where REDUCEPARALLEL can perform a reduction that we have not explicitly shown to be valid. In particular, if  $\mu$  is a node of type (P) and  $E_\mu^{\text{virt}}(\mathcal{T}) = \emptyset$ , i.e.  $\mu$  is the only node in  $\mathcal{T}$ , and  $|E_\mu| = 3$  holds, then we do still apply the reduction from Lemma 23 and Lemma 26 to it, even though we do not have a 2-separation. However, it can easily be seen that both nodes in  $V_\mu$  are star-nodes, and that their  $Y$ -splittability is preserved by replacing two edges by a single edge as done in REDUCEPARALLEL.

For technical reasons, there is one notable exception, which is that REDUCESERIES does not perform local reductions on nodes  $\mu$  of type (S) when  $E_\mu \cap E^{\text{virt}}(\mathcal{T}_R) = \emptyset$ . In this case,  $\mathcal{T}_R$  must consist of a single cycle skeleton  $\mu$ , which implies that  $\mathcal{T}_R$  is still  $Y_R$ -reduced.

To show that  $\mathcal{T}_R$  is an SPQR tree, we show that each node  $\mu \in \mathcal{V}(\mathcal{T}_R)$  is correctly labeled. Since  $\mathcal{T}$  is an SPQR tree, initially all nodes  $\mu \in \mathcal{V}(\mathcal{T})$  are correctly labelled. Hence, we only need to check those nodes for which the skeleton  $G_\mu$  or the label changes. Then, Lemma 31 proves that the node labels of  $\mathcal{T}_R$  are still correct.

To observe minimality of  $\mathcal{T}_R$ , note that the node labels of  $\mathcal{T}$  are not modified, unless we have  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$  where  $\mu$  is of type (P) in REDUCEPARALLEL. In this case  $\mathcal{T}_R$  consists of a single (Q)-node by Lemma 31, which is clearly minimal. Thus, since  $\mathcal{T}$  is minimal,  $\mathcal{T}_R$  must be minimal, too.

By Proposition 16, all 2-separations of any graph  $G$  represented by  $\mathcal{T}$  are either local 2-separations or given by an SPQR tree edge. Consequently, after line 35 the algorithm has applied the reductions Lemmas 22, 23 and 26 to all 2-separations of  $\mathcal{T}$  where these could be applicable. Since each of the reductions preserves the existence of a  $Y_R$ -splittable vertex in any represented graph  $G$ , it follows that  $\mathcal{T}_R$  is  $Y_R$ -reduced.  $\square$

**REDUCETREE** does not present the most efficient way to perform the reductions. In particular, they can be performed more efficiently by starting with the smallest SPQR subtree that contains the edges  $Y$ . Doing so, one implicitly performs the reductions where  $E_i \cap Y = \emptyset$  for  $i = 1, 2$ . Thereafter, such reductions cannot happen because each leaf will have an edge from  $Y$ , and this property is preserved when applying Lemma 26.

Note that we have not yet discussed how to test in line 22 of **REDUCETREE** whether Lemma 26 is applicable. Lemma 27 shows that it suffices to determine all  $Y$ -splittable vertices of the 3-connected skeleton. In the following sections, we will discuss how to (efficiently) find the  $Y$ -splittable vertices in further detail. More precisely, we will characterize when the reduced SPQR tree  $\mathcal{T}_R$  represents a graph that contains a  $Y_R$ -splittable vertex, and describe the structure of such a graph.

First, we will treat the case where  $\mathcal{T}_R$  consists of a single node. Afterwards, we consider the case where  $\mathcal{T}_R$  consists of multiple nodes. Throughout the following sections, we consider the reduced tree  $\mathcal{T}_R$  and its marked edges  $Y_R$ . For ease of notation, we use  $\mathcal{T}$  and  $Y$  rather than  $\mathcal{T}_R$  and  $Y_R$  to indicate the reduced tree and its marked edges.

## 5.2 Updating a single skeleton

First, we consider the case where  $\mathcal{V}(\mathcal{T}) = \{\mu\}$ . The algorithms that we present in this section do not immediately add a new edge, but rather modify  $G_\mu$ , and return a set of two vertices between which the new tree edge can be added. We will say that an SPQR tree that represents such a modified  $G_\mu$  is an  $Y$ -processed SPQR tree. In Definition 33, we formally define  $Y$ -processed SPQR trees.

**Definition 33** ( $Y$ -processed tree). *Let  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$  be an SPQR tree and let  $Y \subseteq E^{\text{reg}}(\mathcal{T})$ . We call another tree  $\mathcal{T}' = (\mathcal{V}', \mathcal{E}')$  with vertices  $v_1, v_2 \in V_\mu$  of a skeleton  $\mu \in \mathcal{V}'$  the  $Y$ -processed tree of  $\mathcal{T}$  if*

- (a) *the tree  $\overline{\mathcal{T}'}$  that arises from  $\mathcal{T}'$  by adding an edge  $e'$  between  $v_1$  and  $v_2$  (in skeleton  $\mu$ ) is an SPQR tree,*
- (b)  *$\overline{\mathcal{T}'}$  represents the set of graph-tree pairs  $(G', T')$  for which there exists a graph-tree pair  $(G, T)$  represented by  $\mathcal{T}$  such that applying the construction of Lemma 11 to any  $Y$ -splittable vertex yields  $G'$  and  $T'$ , and*
- (c)  *$\overline{\mathcal{T}'}$  is a minimal SPQR tree.*

The following lemma highlights that we need to show less than property (b).

**Lemma 34.** *In order to establish Definition 33 (b) it suffices to consider only one graph-tree pair  $(G, T)$  that is represented by  $\mathcal{T}$  and only one  $Y$ -splittable vertex  $v$  of  $G$  such that the resulting graph-tree pair  $(G', T')$  is represented by  $\overline{\mathcal{T}'}$ .*

*Proof.* By Theorem 17, the graphs  $G$  with spanning trees  $T$  represented by  $\mathcal{T}$  are those with the same matrix  $M := M(G, T)$ . By Theorem 12, applying the construction of Lemma 11 to each such  $G$  yields a graph  $G'$  with spanning tree  $T'$ , which corresponds to the augmentation of  $M$  with the binary vector  $b^T$  with  $\text{supp}(b) = Y$  as in (1). By construction, the unique resulting matrix  $M'$  represents each of the graph-tree pairs  $(G', T')$ . Again by Theorem 17, these are represented by the unique minimal SPQR tree  $\overline{\mathcal{T}'}$ .  $\square$

In a few algorithms, the virtual edges  $E_\mu^{\text{virt}}(\mathcal{T})$  connecting to skeletons of the reduced tree, are mentioned. While in our case  $E_\mu^{\text{virt}}(\mathcal{T}) = \emptyset$  holds, this will turn out to be useful later because we will re-use the algorithms in the setting where the reduced tree has more than one node.

First, let us treat the case where  $\mu$  is of type (Q) or (S) after applying **REDUCETREE**. It is not difficult to see that in this case, we can extend  $\mu$  by elongating the existing cycle with the new edge to create a longer cycle.

**Proposition 35.** *Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with  $\mathcal{V}(\mathcal{T}) = \{\mu\}$  and let  $\mu$  be of type (Q) or (S). Let  $G'_{\mu,v}$  be the graph obtained by applying the construction of Lemma 11 to  $G_\mu, T_\mu$  at vertex  $v \in V(G)$ . For all  $v \in V(G)$ ,  $v$  is  $Y$ -splittable and  $G'_{\mu,v}$  is a cycle of length  $|E(G_\mu)| + 1$ .*

*Proof.* Since  $Y$  is nonempty and  $\mu$  has a unique edge  $\{e\} = E_\mu \setminus T_\mu$ , we must have  $Y = \{e\}$ . Because  $G_\mu$  is of type (Q) or (S),  $T_\mu$  forms a path. Applying the construction of Lemma 11 to any internal vertex of the path  $T_\mu$ , which has degree 2 and has two edges from  $T_\mu$  incident to it, we observe that the resulting graph always places exactly one edge in  $\delta^I(v)$  and one edge in  $\delta^J(v)$ . This implies that  $G'_{\mu,v}$  is indeed a cycle of length  $|V(G'_\mu)| + 1$ . The reasoning for the end-vertices of  $T_\mu$  is similar, where a single tree edge and  $e$  must always be placed in different neighborhoods of  $\delta^I(v)$  and  $\delta^J(v)$ .  $\square$

Note that Proposition 35 shows that **EXTENDSERIES** is simply a specialized version of the construction of Lemma 11 for cycles. In particular, we could also simply use **BIPARTITESPLIT**. **SPLITSKELETON** contains the complete algorithm for splitting a single skeleton.

---

**Algorithm 4:** Extend a series skeleton with a new edge

---

**Algorithm:** EXTENDSERIES( $\mathcal{T}, \mu$ )**Input:** SPQR tree  $\mathcal{T}$  with skeleton  $\mu$  of type (Q) or type (S)**Output:**  $Y$ -processed tree  $\mathcal{T}'$  with vertices  $v_1, v_2$ 

- 1 Obtain  $\mathcal{T}'$  from  $\mathcal{T}$  by splitting an arbitrary vertex  $v \in V(G_\mu)$  into two vertices  $v_1$  and  $v_2$  such that both have degree 1 in  $G_\mu$ .
  - 2 **return**  $(\mathcal{T}', \{v_1, v_2\})$
- 

---

**Algorithm 5:** Find all  $Y$ -splittable vertices in a skeleton

---

**Algorithm:** FINDSPLITTABLEVERTICES( $\mathcal{T}, \mu, Y$ )**Input:** SPQR tree  $\mathcal{T}$ , a skeleton  $\mu$ , edges  $Y \subseteq E_\mu^{\text{reg}}(\mathcal{T}) \setminus T_\mu$ **Output:** Set  $X \subseteq V(G_\mu)$  of  $Y$ -splittable vertices of  $\mu$ 

- 1 **if**  $Y = \emptyset$  or  $\mu$  is of type (Q), (S) or (P) **then return**  $V(G_\mu)$
  - 2 Let  $X \subseteq V(G_\mu)$  be the set of vertices  $v$  to which all  $y \in Y$  are incident.
  - 3 **if**  $|X| = 2$  **then return**  $X$
  - 4 Let  $Q := V(G_\mu) \cap \bigcap_{y \in Y} V(P_y(T_\mu))$  be the intersection of the vertices of  $P_y(T_\mu)$  for all  $y \in Y$ .
  - 5 Let  $A \subseteq V(G_\mu)$  be the articulation vertices  $a$  of  $E(G_\mu) \setminus Y$  that also satisfy  $a \in Q$ .
  - 6 **for**  $a \in A$  **do**
  - 7     Construct  $H_Y^a$ .
  - 8     **if**  $H_Y^a$  is bipartite **then**  $X := X \cup \{a\}$
  - 9 **end**
  - 10 **return**  $X$
- 

---

**Algorithm 6:** Find all  $Y$ -splittable vertices in a skeleton that are incident to all virtual edges

---

**Algorithm:** FINDTREESPLITTABLEVERTICES( $\mathcal{T}, \mu, Y$ )**Input:** SPQR tree  $\mathcal{T}$ , a skeleton  $\mu$ , edges  $Y \subseteq E_\mu^{\text{reg}}(\mathcal{T})$ **Output:** Set  $X \subseteq V(G_\mu)$  of  $Y$ -splittable vertices of  $\mu$  that are incident to all virtual edges

- 1  $X := \text{FINDSPLITTABLEVERTICES}(\mathcal{T}, \mu, Y)$
  - 2 **for**  $e \in E_\mu^{\text{virt}}(\mathcal{T})$  **do**  $X := X \cap e$
  - 3 **return**  $X$
- 

---

**Algorithm 7:** Split a vertex as in Lemma 11

---

**Algorithm:** BIPARTITESPLIT( $\mathcal{T}, \mu, Y, v$ )**Input:** SPQR tree  $\mathcal{T}$ , a skeleton  $\mu$ , edges  $Y \subseteq E_\mu^{\text{reg}}(\mathcal{T})$  and a vertex  $v \in V_\mu$ **Output:**  $Y$ -processed tree  $\mathcal{T}'$  with vertices  $v_1, v_2$ 

- 1 Construct the bipartite graph  $H_{Y_\mu}^v$  with bipartition  $I, J \subseteq V(H_Y^v)$ .
  - 2 Obtain  $\mathcal{T}'$  from  $\mathcal{T}$  by the following modifications.
  - 3 Add the new vertices  $v_1$  and  $v_2$  to  $G_\mu$ .
  - 4 **for**  $e = \{u, v\} \in \delta(v)$  **do**
  - 5     **if either**  $e \in Y_\mu$  **or** there is a component  $h \in I$  such that  $u \in h$  **then**
  - 6         replace  $e$ 's end-vertex  $v$  by  $v_1$ .
  - 7     **else**
  - 8         replace  $e$ 's end-vertex  $v$  by  $v_2$ .
  - 9     **end**
  - 10 **end**
  - 11 Remove  $v$  from  $G_\mu$ .
  - 12 **return**  $(\mathcal{T}', \{v_1, v_2\})$
- 

**Lemma 36.** *Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with  $\mathcal{V}(\mathcal{T}) = \{\mu\}$  and let  $\mu$  be of type (Q) or (S). Then the tree resulting from  $\text{SPLITSKELTON}(\mathcal{T}, \mu, Y)$  is  $Y$ -processed with respect to the vertices returned by the algorithm.*

*Proof.* Proposition 35 shows that  $\text{EXTENDSERIES}$  indeed correctly updates  $\mathcal{T}$ . Note that if  $\mu$  is of type (Q) and has  $|E_\mu| = 2$  adding an edge to  $E_\mu$  creates a cycle of length 3, which shows that  $\mu$  must indeed become a (S) node. Clearly, the returned SPQR tree is minimal since it consists of only  $\mu$ .  $\square$

The remaining case where  $\mu$  is of type (R) is also the most complex one. First, we investigate the structure of  $Y$ -splittable vertices in 3-connected graphs. More precisely, we show that there are only a few vertices in a 3-connected graph that may be  $Y$ -splittable.

---

**Algorithm 8:** Construct  $Y$ -processed tree using a  $Y$ -splittable vertex that belongs to a given skeleton and is incident to all virtual edges

---

**Algorithm:** SPLIT\_SKELETON( $\mathcal{T}, \mu, Y$ )

**Input:** SPQR tree  $\mathcal{T}$ , a skeleton  $\mu$ , edges  $Y \subseteq E_{\mu}^{\text{reg}}(\mathcal{T}) \setminus T_{\mu}$

**Output:**  $Y$ -processed tree  $\mathcal{T}'$  with vertices  $v_1, v_2$  or  $(\mathcal{T}, \emptyset)$  if splitting is impossible

- 1 Let  $A := \text{FIND\_TREE\_SPLITTABLE\_VERTICES}(\mathcal{T}, \mu, Y)$ .
- 2 **if**  $\mu$  is of type (Q) **then**
- 3     **if**  $|E_{\mu}| = 2$  **then** Change  $\mu$  to type (S).
- 4     **return** EXTEND\_SERIES( $\mu$ )
- 5 **else if**  $\mu$  is of type (S) **then**
- 6     **if**  $A = V(G_{\mu})$  **then** **return** EXTEND\_SERIES( $\mu$ )
- 7     **else if**  $|A| = 1$  **then**
- 8         Let  $a \in A$  be the unique  $Y$ -splittable vertex.
- 9         **return** BIPARTITE\_SPLIT( $\mathcal{T}, \mu, Y, a$ )
- 10    **else** **return**  $(\mathcal{T}, \emptyset)$
- 11 **else if**  $\mu$  is of type (P) **then**
- 12     Pick an arbitrary  $a \in A$ .
- 13     **return** BIPARTITE\_SPLIT( $\mathcal{T}, \mu, Y, a$ )
- 14 **else if**  $\mu$  is of type (R) **then**
- 15     **if**  $|A| = 0$  **then** **return**  $(\mathcal{T}, \emptyset)$
- 16     **else if**  $|A| = 1$  **then**
- 17         Let  $a \in V(G_{\mu})$  be such that  $A = \{a\}$ .
- 18         **return** BIPARTITE\_SPLIT( $\mathcal{T}, \mu, Y \cap E_{\mu}, a$ )
- 19     **else if**  $|A| = 2$  **then**
- 20         Let  $a_1, a_2 \in V(G_{\mu})$  be such that  $A = \{a_1, a_2\}$ .
- 21         Let  $e = \{a_1, a_2\}$  be the edge connecting  $a_1$  and  $a_2$ .
- 22         Create a new series node  $\omega$  and move  $e$  from  $G_{\mu}$  to  $G_{\omega}$ .
- 23         Create a virtual edge pair  $(f, g)$  between  $\mu$  and  $\omega$  such that  $f$  has  $a_1$  and  $a_2$  as end-vertices.
- 24         **if**  $e \in T_{\mu}$  **then** Update  $T_{\mu} := (T_{\mu} \cup \{e\}) \setminus \{f\}$ .
- 25         **return** EXTEND\_SERIES( $\omega$ )
- 26     **end**
- 27 **end**

---

**Lemma 37.** Let  $G = (V, E)$  be a 3-connected graph with spanning tree  $T \subseteq E$ . Consider a non-empty set  $Y \subseteq E \setminus T$  and let  $Q := \bigcap_{y \in Y} P_y(T)$  be the intersection of the fundamental paths of all  $Y$ -edges. Then all inner vertices of  $Q$  that are articulation vertices of  $G \setminus Y$  must be adjacent on  $Q$ . In particular, there are at most two such vertices.

*Proof.* First note that, due to  $Y \neq \emptyset$ ,  $Q$  is a path. Let  $a_1, a_2 \in V(Q)$  be two inner vertices of  $Q$  that are both articulation vertices of  $G \setminus Y$ , and let  $u_1$  and  $u_2$  be the end-vertices of  $Q$  such that  $a_1$  comes first when traversing  $Q$  from  $u_1$  to  $u_2$ . Suppose that  $a_1$  and  $a_2$  are not adjacent on  $Q$ , which means that the sub-path of  $Q$  from  $a_1$  to  $a_2$  has an inner vertex  $m$ .

Since  $G$  is 3-connected,  $G - \{a_1, a_2\}$  is connected, which implies that each connected component of  $(G \setminus Y) - \{a_1, a_2\}$  is connected to some other component via an edge  $y \in Y$ . However,  $Q \subseteq P_y(T)$  and the fact that  $a_1, a_2$  lie on  $Q$  imply that each of these components contains one end-vertex of  $Q$ . It follows that  $(G \setminus Y) - \{a_1, a_2\}$  has at most two connected components. For  $i = 1, 2$ , let  $C_i$  be the component that contains  $u_i$ . Assume, without loss of generality, that  $m \in C_1$  holds. When adding back  $a_2$ , that is, considering  $(G \setminus Y) - \{a_1\}$ , the components  $C_1$  and  $C_2$  are merged into one. This contradicts the assumption that  $a_1$  is an articulation vertex of  $G \setminus Y$ , which concludes the proof.  $\square$

**Corollary 38.** Let  $G = (V, E)$  be a 3-connected graph with spanning tree  $T \subseteq E$ . Consider a non-empty set  $Y \subseteq E \setminus T$  and let  $Q := \bigcap_{y \in Y} P_y(T)$  be the intersection of the fundamental paths of all  $Y$ -edges. There are at most four vertices on  $Q$  that are articulation vertices of  $G \setminus Y$  or the center of a  $Y$ -star. Moreover, this upper bound is tight.

*Proof.* By definition, only end-vertices of  $Q$  can be the center vertex of a  $Y$ -star. Then considering the two end-vertices of  $Q$  and the two internal vertices of  $Q$  as given by Lemma 37, the result follows.

An example with exactly four vertices is shown in Fig. 8, which proves tightness.  $\square$

In [32], subroutine TEST C, Truemper claims that a 3-connected graph has a graphic row update if and only if it has a single articulation vertex or a star vertex. However, Fig. 8 is a direct contradiction to this claim, since it contains 4 articulation vertices, of which the two internal to  $Q$  are both  $Y$ -splittable. In the following results, we explore the structure corresponding to a pair of  $Y$ -splittable vertices.

**Lemma 39.** Let  $G$  be a connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be such that  $u$  is  $Y$ -splittable with respect to  $G$ . Let  $v \in \delta(u)$  be an adjacent vertex. If the neighborhood split of  $u$  is such that one part of it is non-empty and contains only edges between  $u$  and  $v$ , then  $v$  is  $Y$ -splittable with respect to  $G$ .

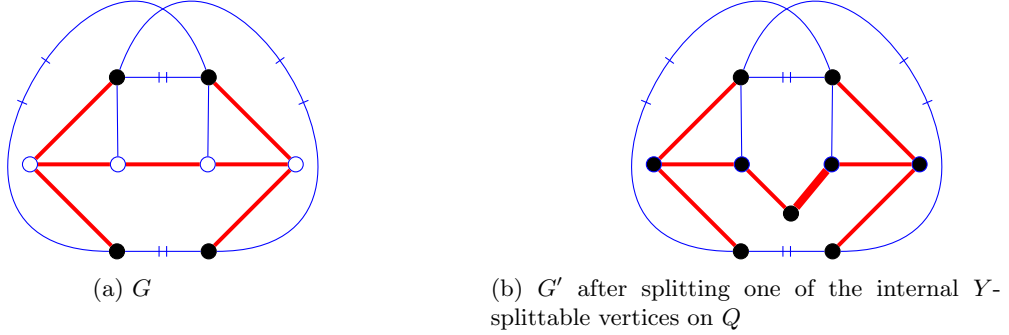


Figure 8: A graph-tree pair where  $G \setminus Y$  has four articulation vertices on  $Q$ . Edges in  $T$  are marked red and bold, all other edges are marked blue. Edges in  $Y$  are marked by two stripes. The articulation vertices are marked in white in a). The newly added row edge is marked extra bold.

*Proof.* Let  $I$  and  $J$  denote the bipartition of  $H_Y^v$  such that  $\delta^I(u)$  is non-empty and only consists of edges between  $u$  and  $v$ . We apply the construction of Lemma 11 to  $u$  and obtain the graph  $G'$  by splitting  $u$  into  $i$  and  $j$  with new edge  $r^* = \{i, j\}$ , updated tree  $T' := T \cup \{r^*\}$  and reassigned edges  $\delta(u)$ . Note that all edges from  $\delta(u)$  are reassigned to  $j$ , except for those in  $\delta^I(u)$ . By the assumptions of the lemma, vertex  $i$  only has the two neighbors  $j$  and  $v$ . We now create the graph  $\overline{G}'$  from  $G'$  by reassigning the end-vertex of each edge incident to  $i$  to the respective other vertex from  $\{j, v\}$ . Thereby, edge  $r^* = \{i, j\}$  is turned into  $\overline{r}^* = \{i, v\}$  and every edge  $e \in \{i, v\}$  is turned into  $\overline{e} = \{j, v\}$ . Let  $\overline{T}' := T \cup \{\overline{r}^*\}$ . First, note that the sets of fundamental cycles of  $G'$  with respect to  $T'$  and of  $\overline{G}'$  with respect to  $\overline{T}'$  are identical. Second, contracting  $\overline{r}^*$  of  $\overline{G}'$  also yields  $G$ . This implies that also  $v$  must be  $Y$ -splittable, where application of Lemma 11 yields the graph  $\overline{G}'$ .  $\square$

Note that the statement of Lemma 39 is only about sufficiency. If we additionally require that  $G$  is 3-connected and that  $Y$  is nonempty, we can show that the condition is also necessary. Although this result is not crucial to our main argument, it can be used in practice to speed up the algorithm. Thus, the proof can be found in Appendix A.

In the next few results, we show that a pair of  $Y$ -splittable vertices must be adjacent in 3-connected graphs, which we will need to prove correctness.

**Lemma 40.** *Let  $G$  be a 3-connected graph with spanning tree  $T$  and a non-empty set  $Y \subseteq E(G) \setminus T$ . Let  $v_1$  and  $v_2$  be distinct  $Y$ -splittable vertices, where  $v_1$  is an articulation vertex of  $G \setminus Y$ . Let  $Q := \bigcap_{y \in Y} P_y(T)$  be the intersection of the fundamental paths of all  $Y$ -edges. Then  $v_1$  and  $v_2$  must be adjacent on  $Q$ .*

*Proof.* First, note that by Lemma 9, both  $v_1$  and  $v_2$  must lie on  $Q$ . We distinguish two cases, depending on the structure of  $Y$ .

**Case 1:**  $Y \subseteq \delta(v_2)$ . Since  $G$  is 3-connected,  $G - \{v_1, v_2\} = (G \setminus Y) - \{v_1, v_2\}$  must be connected. Since  $v_1$  is an articulation vertex of  $G \setminus Y$ ,  $G_Y^{v_1} = (G \setminus Y) - v_1$  is disconnected. The connectivity of  $(G \setminus Y) - \{v_1, v_2\}$  implies that  $v_2$  must be disconnected from all other vertices in  $G_Y^{v_1}$ . Thus, we must have in particular that  $\delta(v_2) \subseteq \delta(v_1) \cup Y$ . However, since  $T \cap Y = \emptyset$  and  $T$  is a spanning tree, there must exist a spanning tree edge in  $\delta(v_1) \cap \delta(v_2)$ , which clearly belongs to  $Q$  since  $v_1$  and  $v_2$  are on  $Q$ . Thus,  $v_1$  and  $v_2$  are adjacent on  $Q$ .

**Case 2:**  $Y \not\subseteq \delta(v_2)$ . For the sake of contradiction, assume that  $v_1$  and  $v_2$  are not adjacent on  $Q$ . Then there exists a vertex  $m$  on  $Q$  that lies between  $v_1$  and  $v_2$ . Since  $v_2$  is  $Y$ -splittable,  $v_2$  must be an articulation vertex of  $G \setminus Y$  by Lemma 8. Since  $G$  is 3-connected,  $G - \{v_1, v_2\}$  is connected, which implies that each connected component of  $(G \setminus Y) - \{v_1, v_2\}$  is connected to some other component via an edge  $y \in Y$ . Let  $y^* \in Y$  be an edge that connects the component  $C_1$  containing  $m$  to another component  $C_2$ . Because  $v_1$  and  $v_2$  and  $m$  lie on  $Q$ , there can be no edge in  $Y$  that connects to the nodes of the component of  $T \setminus (\delta(v_1) \cup \delta(v_2))$  containing  $m$ .

Then, by symmetry of  $v_1$  and  $v_2$ , we can assume without loss of generality that the unique vertex  $u \in C_1 \cap y^*$  has the property that the  $u$ - $m$ -path within  $T$  traverses through  $v_1$ . The other end node  $w \in C_2 \cap y^*$  must have a unique  $w$ - $m$  path in  $T$ . Because we defined  $u$  is in the same component as  $M$ , there exists a  $u$ - $m$  path  $P$  in  $(G \setminus Y) - \{v_1, v_2\}$ . Clearly,  $P$  also exists in  $G \setminus Y - \{v_1\}$ . When adding back  $v_2$  along with its incident edges, that is, considering  $G_Y^{v_1} = (G \setminus Y) - \{v_1\}$ , the  $w$ - $m$  path in  $T$  connects  $w$  with  $m$  again. However, then  $y^*$  induces a loop in  $H_Y^{v_1}$  that is given by the  $w$ - $m$  path,  $P$  and  $y^*$  itself, contradicting that  $v_1$  was  $Y$ -splittable. This concludes the proof.  $\square$



**Lemma 41.** *Let  $G$  be a 3-connected graph with at least four edges, a spanning tree  $T$  and a non-empty set  $Y \subseteq E(G) \setminus T$ . Then  $G$  has at most two  $Y$ -splittable vertices. Additionally, if  $G$  has two  $Y$ -splittable vertices then these must be adjacent.*

*Proof.* Consider two  $Y$ -splittable vertices  $v_1$  and  $v_2$ . If  $Y \subseteq \delta(v_1) \cap \delta(v_2)$  then  $v_1$  and  $v_2$  are connected by some edge from  $Y \neq \emptyset$ , showing that they are adjacent. Otherwise, Lemma 8 shows that one of them is an articulation vertex of  $G_Y$  in which case Lemma 40 shows that  $v_1$  and  $v_2$  are connected by an edge from  $T$ . This proves the second statement.

Assume, for the sake of contradiction, that there exist three  $Y$ -splittable vertices  $v_1, v_2$  and  $v_3$ . Since  $T$  is a tree, they cannot be pair-wise connected via edges from  $T$ , which implies that we have  $Y \subseteq \delta(v_i) \cap \delta(v_j)$  for some  $i, j \in \{1, 2, 3\}$ . Let  $k \in \{1, 2, 3\} \setminus \{i, j\}$  be the third index. Note that this implies  $Y \not\subseteq \delta(v_k)$ , and hence  $v_k$  is connected to  $v_i$  and  $v_j$  by an edge from  $T$ . Since  $G$  has at least four edges, none of which are parallel due to its 3-connectivity, there must exist a fourth vertex  $v_4 \in V(G) \setminus \{v_1, v_2, v_3\}$ . Since  $G$  is 3-connected,  $G - \{v_i, v_k\}$  and  $G - \{v_j, v_k\}$  are both connected, which implies that there exist paths  $P_i$  and  $P_j$  from  $v_4$  to  $v_i$  and to  $v_j$  that do not traverse any of the other two vertices  $\{v_j, v_k\}$  or  $\{v_i, v_k\}$ , respectively. Hence, the path  $P_i \cup P_j$  connects  $v_i$  with  $v_j$  (via  $v_4$ ) within  $G_Y^{v_3}$ , which induces a self-loop in  $H_Y^{v_3}$ , which in turn contradicts our assumption that  $v_3$  is  $Y$ -splittable.  $\square$

In Lemma 41, we fully explored the cases where a 3-connected graph  $G$  can have multiple  $Y$ -splittable points. Now, we are ready to prove correctness of the proposed algorithms.

**Lemma 42.** *Let  $\mathcal{T}$  be a  $Y$ -reduced tree that contains a node  $\mu$  of type (R). Then algorithm `FINDSPLITTABLEVERTICES`( $\mathcal{T}, \mu, Y$ ) correctly identifies all  $Y$ -splittable vertices of  $G_\mu$ .*

*Proof.* For the first step which checks if  $Y = \emptyset$ , Corollary 7 shows that if  $Y = \emptyset$  holds, that the algorithm functions correctly. In the next steps, we check if  $Y$  is a star for Proposition 6, and we find all articulation vertices of  $G_\mu \setminus Y$  that lie on  $Q$  (to check Lemma 9). Then we simply check if the auxiliary graph  $H_Y^\alpha$  is bipartite for each articulation vertex  $\alpha$  of  $G \setminus Y$  that lies on  $Q$ . By Lemma 8, this is an exhaustive procedure for finding all  $Y$ -splittable vertices of  $G_\mu$ . Corollary 38 shows that we need to construct the auxiliary graph for at most four vertices, and Lemma 41 guarantees that we can find at most two  $Y$ -splittable vertices.  $\square$

**Lemma 43.** *Let  $\mathcal{T}$  be a  $Y$ -reduced tree with  $\mathcal{V}(\mathcal{T}) = \{\mu\}$  and let  $\mu$  be of type (R). Let  $\mathcal{T}', X := \text{SPLITSKELTON}(\mathcal{T}, \mu, Y)$ . If  $X = \emptyset$ , then  $\mathcal{T}$  does not represent a graph  $G$  that contains a  $Y$ -splittable vertex  $v$ . Otherwise,  $X = \{v_1, v_2\}$  and  $\mathcal{T}'$  is a  $Y$ -processed SPQR tree with respect to  $v_1$  and  $v_2$ .*

*Proof.* First, note that  $E^{\text{virt}}(\mathcal{T}) = \emptyset$  implies that `FINDTREESPLITTABLEVERTICES`( $\mathcal{T}, \mu, Y$ ) simply returns `FINDSPLITTABLEVERTICES`( $\mathcal{T}, \mu, Y$ ). By Lemma 42, `FINDSPLITTABLEVERTICES`( $\mathcal{T}, \mu, Y$ ) finds all  $Y$ -splittable vertices of  $G_\mu$ . By Lemma 41, there are at most two such  $Y$ -splittable vertices, and if there are two, they are adjacent to each other. Let us make a case distinction on  $|A|$ , as in the algorithm.

If  $|A| = 0$ , i.e.,  $A = \emptyset$ , then  $G$  has no  $Y$ -splittable vertex. Hence,  $\mathcal{T}$  does not represent any graph  $G$  that contains a  $Y$ -splittable vertex.

If  $|A| = 1$ , then there is a unique  $Y$ -splittable vertex  $a$  for  $G$ . In this case, Theorem 51 shows that the application of `BIPARTITESPLIT`, together with the newly added row edge, gives a new graph  $G'$  which is still of type (R). Clearly, the updated SPQR tree  $\overline{\mathcal{T}'}$  obtained by adding this edge is minimal, as it contains a single (R)-node.

Finally, if  $|A| = 2$ , we know by Lemma 41 that exactly one edge  $e$  (since  $G$  is simple) is incident to both vertices in  $A$ . Additionally, by combining Lemma 39 and Lemma 11, we observe that  $\{e\}$  forms one side of the neighborhood partition for both  $Y$ -splittable vertices in  $A$ . If we then apply the construction of Lemma 11 to  $G$ , we end up with a graph  $G'$  where  $e$  is replaced by  $e$  and the new row edge  $r$ , put in series. The order of  $e$  and  $r$  depends on the vertex in  $A$  to which Definition 10 was applied. Then we observe that the SPQR tree of  $G$  then consists of  $\mu$ , where  $e$  is replaced by a virtual edge which connects to a new node  $\nu$  of type (S). Here,  $E_\nu$  consists of a virtual edge connecting to  $\mu$ ,  $e$ , and the new row edge. Since the structure of  $G_\mu$  did not change,  $G_\mu$  is still 3-connected and simple. Moreover,  $|E_\nu| = 3$  holds, so that  $\nu$  is indeed of type (S). Clearly,  $\overline{\mathcal{T}'}$  is minimal since it contains only a single (R)–(S) connection.

For  $|A| = 1$  and  $|A| = 2$ , we obtain a graphic realization  $G'$  by splitting a  $Y$ -splittable vertex in  $G$  by Lemma 11.

This shows that Definition 33 (b) is satisfied. Additionally, we have shown that in both cases,  $\overline{\mathcal{T}'}$  is a minimal SPQR tree. Thus,  $\mathcal{T}'$  is a  $Y$ -processed SPQR tree with respect to  $v_1$  and  $v_2$  if  $X \neq \emptyset$ .  $\square$

**Theorem 44.** *Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with  $\mathcal{V}(\mathcal{T}) = \{\mu\}$  and consider the pair  $(\mathcal{T}', X) := \text{SPLITSKELTON}(\mathcal{T}, \mu, Y)$ . If  $X = \emptyset$ , then  $\mathcal{T}$  does not represent a graph  $G$  that contains a  $Y$ -splittable vertex  $v$ . Otherwise,  $X = \{v_1, v_2\}$  holds and  $\mathcal{T}'$  is a  $Y$ -processed SPQR tree with respect to  $v_1$  and  $v_2$ .*

*Proof.* By Proposition 30,  $\mu$  must be of type (Q), (S) or (R). If  $\mu$  is of type (Q) or (S), then Lemma 36 shows that  $X \neq \emptyset$  and that  $\mathcal{T}'$  is a  $Y$ -processed tree with respect to  $v_1$  and  $v_2$ . If  $\mu$  is of type (R), then Lemma 43 shows the result.  $\square$

Note that the proof of Lemma 43 refers to Theorem 51, which we have not yet explored. Theorem 51 shows that if  $G$  has a unique  $Y$ -splittable node, that it remains 3-connected. The theorem provides a more general statement than needed for Lemma 43, and will be explored in the next section, where we will discuss the more complicated case in which  $|\mathcal{V}(\mathcal{T})| \geq 2$  holds.

### 5.3 Updating multiple skeletons by merging

In this section, we will consider what happens when the  $Y$ -reduced tree  $\mathcal{T}$  has  $|\mathcal{V}(\mathcal{T})| \geq 2$ . We start by showing that it suffices to test  $Y$ -splittability of the individual skeleton nodes of the SPQR tree. To do so, we first consider a 2-separation of any graph represented by  $\mathcal{T}$ . Then we show that any  $Y$ -splittable vertex must be one of the two separating vertices of the 2-separation.

**Lemma 45.** *Let  $G$  be a 2-connected multigraph that has a 2-separation  $(E_1, E_2)$  with separating vertices  $\{u, w\}$ . For  $i = 1, 2$ , let  $G_i$  denote the graph with vertex set  $V(E_i)$  and edge set  $E_i \cup \{e_i\}$  for a new edge  $e_i := \{u, w\}$ , and let  $Y_i := Y \cap E_i$ . Then  $v \in \{u, w\}$  is  $Y$ -splittable in  $G$  if and only if  $v$  is  $Y_i$ -splittable in  $G_i$  for  $i \in \{1, 2\}$ .*

*Proof.* Since the statement of the lemma is symmetric in  $u$  and  $w$ , we only show it for  $v = u$ . First, observe that  $G - v$  has the articulation vertex  $w$  and that  $G - v$  can be obtained from  $G_1 - v$  and  $G_2 - v$  by taking their union and identifying the vertex  $w$  that belongs to both graphs. Let  $C, C_1$  and  $C_2$  denote the sets of connected components of  $G_Y^v = (G \setminus Y) - v, (G_1 \setminus Y_1) - v$  and  $(G_2 \setminus Y_2) - v$ , respectively. Moreover, let  $h_w, h_w^1$  and  $h_w^2$  denote the component of  $C, C_1$  and  $C_2$ , respectively, that contains the vertex  $w$ . From  $V(G_1 - v) \cap V(G_2 - v) = \{w\}$  we obtain that  $C \setminus \{h_w\}$  is the disjoint union of  $C_1 \setminus \{h_w^1\}$  and  $C_2 \setminus \{h_w^2\}$ . Finally, each  $y \in Y$  belongs to some  $E_i$  for some  $i \in \{1, 2\}$  and thus connects only components that both belong to  $C_i$ . This shows that in the auxiliary graph  $H_Y^v(G)$ , the component  $h_w$  is also an articulation vertex (or contains  $V(G_i)$  for some  $i \in \{1, 2\}$ ). We conclude that  $H_Y^v(G)$  is bipartite if and only if  $H_{Y_i}^v(G_i)$  is bipartite for both  $i = 1, 2$ , which yields the claimed result.  $\square$

**Lemma 46.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T$  and  $Y \subseteq E(G) \setminus T$ . Let  $(E_1, E_2)$  be a 2-separation of  $G$  with separating vertices  $u$  and  $w$  such that  $P_{u,w}(T) \subseteq E_1$ , such that Lemmas 22, 23 and 26 are not applicable. Let  $(G_1, T_1)$  be the graph-tree pair given by  $E_1$  augmented by an edge  $e = \{u, w\}$  with  $T_1 = E_1 \cap T$  and let  $(G_2, T_2)$  be the graph-tree pair formed by  $E_2$  augmented by an edge  $e_2 = \{u, w\}$  with  $T_2 = (E_2 \cap T) \cup \{e\}$ . Then no vertex  $v \in V(G) \setminus \{u, w\}$  is  $Y$ -splittable, and  $v \in \{u, w\}$  is  $Y$ -splittable if and only if  $v$  is  $(Y \cap E_i)$ -splittable in  $(G_i, T_i)$  for  $i = 1, 2$ .*

*Proof.* First, let us show that no vertex in  $V(G) \setminus \{u, w\}$  can be  $Y$ -splittable. Without loss of generality, assume that  $P_{u,w}(T) \subseteq E_1$ . As Lemmas 22 and 23 are not applicable, there exists  $y_i \in Y \cap E_i$  for  $i = 1, 2$ . Then by Lemma 21,  $P_{y_1}(T) \subseteq E_1$  implies that no vertex in  $V(G_2) \setminus \{u, w\}$  can be  $Y$ -splittable. Next, consider the case  $v \in V(G_1) \setminus \{u, w\}$ . If there exists a  $y \in Y \cap E_2$  such that  $v \notin P_y(T)$  then by Lemma 9  $v$  is not  $Y$ -splittable. Otherwise, we must have that  $v \in P_{u,w}(T)$  and  $Y \cap E_2 \subseteq P_{u,w}^{-1}(G, T) \cap E_2$ , which shows that  $y_2 \in P_{u,w}^{-1}(G, T) \cap E_2$ . Since equality does not hold as Lemma 26 is not applicable, there exists an edge  $c \in P_{u,w}^{-1}(G, T) \cap E_2$ . However, then  $H_Y^v$  has a self-loop given by  $c, T_u$  and  $T_w$  and  $y_2$  as  $y_2$  and  $c$  both connect  $T_u$  to  $T_w$  which implies that  $v$  is not  $Y$ -splittable.

Finally, application of Lemma 45 to  $v \in \{u, w\}$  shows the desired result.  $\square$

In Lemma 47 we show that it is sufficient to check the  $Y$ -splittability for each skeleton separately, which we prove by repeatedly applying Lemma 45.

**Lemma 47.** *Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with at least two nodes, and let  $G$  be represented by  $\mathcal{T}$  using the mapping  $\Phi_G$  in Definition 15. Then  $v \in V(G)$  is  $Y$ -splittable if and only if for each  $\mu \in \mathcal{V}(\mathcal{T})$ , the skeleton  $G_\mu$  contains a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  such that  $\Phi_G(v_\mu) = v$  holds.*

*Proof.* We define  $\mathcal{T}_0$  as the SPQR tree  $\mathcal{T}$  in which the skeletons  $\mu$  of type (S) are reordered as in step 1 of Definition 15. For  $i = 1, 2, \dots, k$ , we define  $\mathcal{T}_i$  as the tree obtained from  $\mathcal{T}_{i-1}$  by carrying a single step 2 from Definition 15 using the mapping  $\Phi_G$  of vertices for an arbitrary edge of  $\mathcal{T}_{i-1}$ . Consequently,  $\mathcal{T}_k$  consists of a single node whose skeleton is equal to  $G$ . We consider the following statement for all  $i = 0, 1, \dots, k$ :

$$\text{for each } \mu \in \mathcal{V}(\mathcal{T}_i), G_\mu \text{ contains a } (Y \cap E_\mu)\text{-splittable vertex } v_\mu \text{ with } \Phi_G(v_\mu) = v. \quad (3)_i$$

We now show that  $(3)_i$  implies  $(3)_{i+1}$ , where  $i \in \{0, 1, \dots, k-1\}$ . Let  $\{\mu, \nu\} \in \mathcal{E}(\mathcal{T}_i)$  be the unique edge of  $\mathcal{T}_i$  that does not belong to  $\mathcal{T}_{i+1}$ , and whose skeletons  $G_\mu$  and  $G_\nu$  are merged into a new skeleton  $G_\xi$ . Let  $v_\mu$  be the  $(Y \cap E_\mu)$ -splittable vertex with  $\Phi_G(v_\mu) = v$  and let  $v_\nu$  be the  $(Y \cap E_\nu)$ -splittable vertex with  $\Phi_G(v_\nu) = v$ . Since for each skeleton,  $\Phi$  is injective,  $v_\mu$  and  $v_\nu$  must be identified to the same vertex in  $G_\xi$ , which we call  $v_\xi$ . This means that  $v_\xi$  is one of the separating vertices of the 2-separation of  $G_\xi$  that corresponds to the SPQR tree edge  $\{\mu, \nu\}$ . Lemma 45 implies that also  $v_\xi$  is  $(Y \cap E_\xi)$ -splittable. The claim follows from  $\Phi(v_\xi) = \Phi(v_\mu) = v$ .

By induction, we obtain that  $(3)_0$  implies  $(3)_k$ , which proves sufficiency for  $v \in V(G)$  being  $Y$ -splittable.

For necessity, let  $v \in V(G)$  be  $Y$ -splittable and consider any skeleton  $\mu \in \mathcal{V}(\mathcal{T})$ . Due to  $|\mathcal{V}(\mathcal{T}_0)| \geq 2$  there exists a virtual edge  $\{u, w\} \in E_\mu \cap E^{\text{virt}}(\mathcal{T}_0)$ , which corresponds to a 2-separation of  $G$ , having  $\Phi_G(u)$  and  $\Phi_G(w)$  as separating vertices. Since  $\mathcal{T}$  is  $Y$ -reduced, so is  $\mathcal{T}_0$ , and Lemma 46 is applicable to this 2-separation of  $G$ , which implies  $v \in \{\Phi_G(u), \Phi_G(w)\}$ . This shows that every skeleton  $G_\mu$  of  $\mathcal{T}_0$  contains a unique vertex  $v_\mu$  with  $\Phi_G(v_\mu) = v$ . When applying step 1 of Definition 15, two such vertices are

merged into one of the resulting skeleton. Hence, by induction on  $i$ , also every skeleton  $G_\mu$  of  $\mathcal{T}_i$  contains a unique vertex  $v_\mu$  with  $\Phi_G(v_\mu) = v$ .

We now show that  $(3)_{i+1}$  implies  $(3)_i$ , where  $i \in \{0, 1, \dots, k-1\}$ . Let again  $\{\mu, \nu\} \in \mathcal{E}(\mathcal{T}_i)$  be the unique edge of  $\mathcal{T}_i$  that does not belong to  $\mathcal{T}_{i+1}$ , and whose skeletons  $G_\mu$  and  $G_\nu$  are merged into a new skeleton  $G_\xi$ . From the previous paragraph we know that  $v_\xi$  must be one of the separating vertices of the 2-separation of  $G_\xi$  that corresponds to the SPQR tree edge  $\{\mu, \nu\}$ . Again, Lemma 45 implies that also  $v_\mu$  is  $(Y \cap E_\mu)$ -splittable and that  $v_\nu$  is  $(Y \cap E_\nu)$ -splittable. The claim follows.

By induction, we obtain that  $(3)_k$  implies  $(3)_0$ . The latter statement, i.e., that each skeleton  $G_\mu$  of  $\mathcal{T}_0$  has a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  with  $\Phi_G(v_\mu) = v$  is clearly equivalent to the statement for the original SPQR tree  $\mathcal{T}$ , i.e., that each skeleton  $G_\mu$  of  $\mathcal{T}$  has a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  with  $\Phi_G(v_\mu) = v$ . This concludes the proof.  $\square$

Lemma 47 shows that the structure of represented graphs  $G$  that contain a  $Y$ -splittable vertex  $v$  is quite restricted. In particular,  $v$  must be obtained by mapping all individual  $Y \cap E_\mu$ -splittable vertices into one. Next, we show that this implies that each  $v_\mu$  must be incident to all virtual edges within its skeleton. The immediate algorithmic consequence of this characterization is that it can be checked locally in each skeleton.

**Lemma 48.** *Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with at least two nodes. Then  $\mathcal{T}$  represents a graph  $G$  with a  $Y$ -splittable vertex  $v$  if and only if each skeleton  $\mu \in \mathcal{V}(\mathcal{T})$  contains a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu \in V(G_\mu)$  such that  $v_\mu$  is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ .*

*Proof.* For necessity, let  $G$  be any graph represented by  $\mathcal{T}$  with a  $Y$ -splittable vertex  $v$ . Applying Lemma 47, each  $\mu \in \mathcal{V}(\mathcal{T})$  contains a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  such that  $\Phi_G(v_\mu) = v$ . Since Definition 15 only alters  $\Phi_G$  for vertices that are merged, which are exactly the vertices that are incident to virtual edges, this implies that each  $v_\mu$  must be incident to each virtual edge of  $E_\mu^{\text{virt}}(\mathcal{T})$ .

For sufficiency, assume that each  $\mu \in \mathcal{V}(\mathcal{T})$  contains a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  such that  $v_\mu$  is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ . For a node  $\mu$  of type (S), any vertex  $v_\mu$  has two incident edges, which implies  $|E_\mu^{\text{virt}}(\mathcal{T})| \leq 2$ . Since  $\mu$  is of type (S),  $|E_\mu| \geq 3$  holds, and thus Proposition 30 implies  $|E_\mu^{\text{virt}}(\mathcal{T})| = 2$  and  $|E_\mu| = 3$ . We create a realization  $G$  of  $\mathcal{T}$  by mapping  $v_\mu$  to  $v_\nu$  for each 2-separation given by an edge  $\{\mu, \nu\}$  of the SPQR tree. We can ignore the permutation of series members in Definition 15 since reorienting any series member with  $|E_\mu| = 3$  does not change the realized graph. Since we map all the  $Y \cap E_\mu$ -splittable vertices into one vertex, it follows that  $\Phi_G(v_\mu) = v$  holds for some vertex  $v \in V(G)$  and for all  $\mu \in \mathcal{V}(\mathcal{T})$ . The result follows by applying Lemma 47.  $\square$

When proving sufficiency, we observed that all nodes of type (S) had degree 2. We show that this is indeed necessary and that in this case the  $Y$ -splittable vertex must be unique.

**Corollary 49.** *Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with a node  $\mu \in \mathcal{V}(\mathcal{T})$  of type (S). If  $\mu$  has degree 2 in  $\mathcal{T}$ , any graph  $G$  represented by  $\mathcal{T}$  has at most one  $Y$ -splittable vertex. Additionally, if  $\mu$  has degree at least 3, then no graph  $G$  represented by  $\mathcal{T}$  has a  $Y$ -splittable vertex.*

*Proof.* First, consider the case where  $\mu$  has degree at least 3. Since every node of  $G_\mu$  has degree 2, but  $|E_\mu^{\text{virt}}(\mathcal{T})| \geq 3$  holds, there cannot exist a vertex  $v$  that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ . Thus, Lemma 48 implies that there does not exist a graph represented by  $\mathcal{T}$  that contains a  $Y$ -splittable vertex  $v$ .

Second, consider the case where  $\mu$  has degree 2 and let  $G$  be any graph represented by  $\mathcal{T}$ . Since  $|E_\mu^{\text{virt}}(\mathcal{T})| = 2$  holds, it follows from  $|E_\mu| \geq 3$  that  $|E_\mu| = 3$  holds by Proposition 30. Thus, there is exactly one vertex  $v_\mu$  in  $G_\mu$  that is incident to both virtual edges  $E_\mu^{\text{virt}}(\mathcal{T})$ . Let  $v$  be a  $Y$ -splittable node in  $G$ . By Lemma 47, we have that  $\Phi_G(v_\mu) = v$  holds. Since  $v_\mu$  is the unique  $(Y \cap E_\mu)$ -splittable vertex in  $G_\mu$  that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$  and since  $\Phi_G$  is surjective,  $v$  must be unique as well.  $\square$

Next, we will argue that the previous results imply that minimal SPQR trees  $\mathcal{T}$  with  $|\mathcal{V}(\mathcal{T})| \geq 2$  admit at most one  $Y$ -splittable vertex.

**Lemma 50.** *Let  $\mathcal{T}$  be a minimal  $Y$ -reduced SPQR tree with at least two nodes. Then every graph  $G$  represented by  $\mathcal{T}$  contains at most one  $Y$ -splittable vertex.*

*Proof.* Let  $G$  be a graph represented by  $\mathcal{T}$ . If  $\mathcal{T}$  contains a (S)-node, it has degree 2 or greater by Proposition 30. Then,  $G$  contains at most one  $Y$ -splittable vertex by Corollary 49. Thus, we can assume that  $\mathcal{T}$  does not have an (S)-node.

By minimality of  $\mathcal{T}$ , it does not contain two adjacent (P)-nodes, and must thus contain some node of type (R). Now, suppose that  $\mathcal{T}$  has a leaf node  $\mu$  of type (R). Let  $\{u, w\}$  be the vertices of the 2-separation in  $G$  given by the edges  $E_\mu^{\text{reg}}(\mathcal{T})$  and  $E^{\text{reg}}(\mathcal{T}) \setminus E_\mu^{\text{reg}}(\mathcal{T})$ . Since Lemma 26 was not applicable,  $Y \cap E_\mu \neq P_{u,w}^{-1}(G, \mathcal{T}) \cap E_\mu$  holds. Then Lemma 27 implies that at most one of  $u$  and  $w$  can be  $Y$ -splittable. Consequently, Lemma 46 implies that at most one vertex of  $G$  can be  $Y$ -splittable, as  $u$  and  $w$  are the only possible  $Y$ -splittable vertices of  $G$ .

It remains to show the statement for the case in which all leaves are of type (P). Since  $\mathcal{T}$  is minimal, (P)-nodes are not adjacent in  $\mathcal{T}$ . Hence, there must exist a non-leaf node  $\nu$  of type (R) with degree at

least 2. This implies that  $\nu$  has at least two virtual edges. Since  $\nu$  is of type **(R)**,  $G_\nu$  is simple, and hence there is at most one vertex of  $G_\nu$  that is incident to every virtual edge, since  $G_\nu$  must have at least two virtual edges. By applying Lemma 46 to each 2-separation of  $G$  that corresponds to a virtual edge in  $G_\nu$ , only such a vertex may be  $Y$ -splittable. This concludes the proof.  $\square$

We now utilize the previous uniqueness results. In particular, we show that if a graph  $G$  represented by a  $Y$ -reduced SPQR tree has a unique  $Y$ -splittable vertex, that its updated graph  $G'$  is simple and 3-connected. This implies that in the SPQR tree  $\mathcal{T}$  is merged into a single node of type **(R)** when we perform the neighborhood split.

**Theorem 51.** *Let  $G$  be a 2-connected multigraph with spanning tree  $T \subseteq E(G)$  and a non-empty set of edges  $Y \subseteq E(G) \setminus T$  with a unique  $Y$ -splittable vertex  $v$ . Let  $G'$  be the graph constructed by splitting  $v$  into  $v_1$  and  $v_2$ , reassigning the edges according to Lemma 11 and adding the new edge  $r = \{v_1, v_2\}$ . Then  $G'$  has at least four edges. Additionally, if no 2-separation  $(E_1, E_2)$  of  $G$  satisfies the conditions of Lemmas 22, 23 or 26, then  $G'$  is simple and 3-connected.*

*Proof.* Assume that  $|E(G')| < 4$ , which implies that  $|E(G)| < 3$ . Since  $Y$  is non-empty and  $G$  contains a spanning tree  $T$  with  $T \cap Y = \emptyset$ , we must have  $|E(G)| = 2$ . Since  $G$  is 2-connected,  $G$  consists of two parallel edges. Because both vertices in  $G$  are symmetric,  $G$  cannot have a unique  $Y$ -splittable vertex, which is a contradiction. We conclude that  $|E(G)| \geq 3$  and thus  $|E(G')| \geq 4$  holds.

Assume, for the sake of contradiction, that  $G'$  is not 3-connected or not simple. In either case,  $G'$  has a  $k$ -separation  $(E'_1, E'_2)$  for  $k \in \{1, 2\}$  with separating vertices  $\{u', w'\}$  and the corresponding graphs  $G'_1$  and  $G'_2$ , where we allow  $u' = w'$  in case  $k = 1$ . Without loss of generality we can assume  $r \in E'_1$  and denote by  $E_1 := E'_1 \setminus \{r\}$  and  $E_2 := E'_2$  a partition of the edge set of  $G := G'/r$ . These edge sets belong to the graphs  $G_1 := G'_1/r$  and  $G_2 := G'_2$ , respectively. We denote by  $u$  and  $w$  the vertices  $u'$  and  $w'$  after contraction of  $r$ , respectively. Let  $T' := T \cup \{r\}$  denote the spanning tree of  $G'$  that is obtained from  $T$  via Lemma 11.

First, observe that  $G_2 = G'_2$  and  $G_1 = G'_1/r$  together imply that  $V(G_1) \cap V(G_2) = \{u, w\}$  holds, i.e., contraction of  $r$  cannot enlarge the set of vertices shared by the two subgraphs.

Second, since  $(E'_1, E'_2)$  is a  $k$ -separation of  $G'$  we have  $|E_2| = |E'_2| \geq k$ . Similarly,  $|E_1| = |E'_1| - 1 \geq k - 1$  follows. For the case  $k = 1$ ,  $|E_1| = k - 1 = 0$  would imply  $E'_1 = \{r\}$ . Together with the fact that  $u' = w'$  is an articulation vertex of  $G'$  this would induce a degree 1 vertex in  $G$ , which contradicts 2-connectivity. For the case  $k = 2$ ,  $|E_1| = k - 1 = 1$  would imply  $E_1 = \{e\}$  for some edge  $e$ . Since  $u, w, v \in V(G_1)$  hold, these vertices cannot be distinct, say,  $v = u$  as well as  $e = \{u, w\}$ , would hold. Moreover, we would have that  $G'_1$  is a path of length 2 from  $u' = v_1$  to  $w'$  via edges  $r$  and  $e$  with inner vertex  $v_2$ . However, the neighborhood split of  $v$  (in  $G$ ) must have been of the form  $\delta^I(v) = \{e\}$  and  $\delta^J(v) = \delta(v) \setminus \{e\}$ , implying  $Y$ -splittability of  $w$  due to Lemma 39, which contradicts uniqueness of the  $Y$ -splittable vertex  $v \neq w$ . We conclude that  $|E_1| = k - 1$  is not possible, and thus that  $(E_1, E_2)$  form a  $k$ -separation of  $G$ . Clearly, 2-connectivity of  $G$  yields that  $k = 1$  is impossible, and hence  $k = 2$  must hold.

Third, our assumptions imply that Lemmas 22 and 23 were not applicable to the 2-separation  $(E_1, E_2)$ , which implies  $Y \cap E_2 \neq \emptyset$ . Let  $y \in Y \cap E_2$  and note that  $r \in P_y(T')$  holds. If  $P_y(T) \subseteq E_2$  would hold, then  $v_1$  and  $v_2$  would both belong to  $V(G'_2)$  and to  $V(G'_1)$  (due to  $r \in E'_1$ ), which would imply  $\{v_1, v_2\} = \{u', w'\}$ . However, the contraction of  $r$  would yield an articulation vertex  $u = w$  in this case, contradicting that  $\{u, w\}$  is a 2-separation. We conclude that  $P_y(T) \not\subseteq E_2$ , i.e., that  $P_{u,w}(T) \subseteq E_1$ . Since this holds for any choice of  $y \in Y \cap E_2$ , we also have  $Y \cap E_2 \subseteq P_{u,w}^{-1}(G, T) \cap E_2$ .

Fourth, our assumptions imply that Lemma 26 is not applicable to the 2-separation  $(E_1, E_2)$ , which implies that  $Y \cap E_2 \not\subseteq P_{u,w}^{-1}(G, T) \cap E_2$  holds, i.e., there exists an edge  $e \in (P_{u,w}^{-1}(G, T) \setminus Y) \cap E_2$ . If  $v \notin \{u, w\}$  would hold, then the fact that  $(T \cap E_2) \cup \{e\}$  forms a spanning tree of  $G_2$  implies that all vertices  $V(G_2)$  would belong to the same connected component of  $G_Y^v$ . Moreover, each of the edges  $y \in Y \cap E_2$  would induce a self-loop in the bipartite graph  $H_Y^v$ , contradicting that  $v$  is  $Y$ -splittable. We conclude that  $v \in \{u, w\}$  holds.

Without loss of generality, we can assume that  $v_1 = u'$  and that  $v_2 \in V(G'_1) \setminus V(G'_2)$  holds. Let  $f_1, f_2 \in \delta(v)$  be the unique edges of the cycle  $P_e(T) \cup \{e\}$  that are incident to  $v$ . Since  $P_{u,w}(T) \subseteq P_e(T)$  holds, exactly one of these two edges belongs to  $P_{u,w}(T)$  and the other belongs to  $E_2$ . We assume without loss of generality that  $f_1 \in P_{u,w}(T)$  and  $f_2 \in E_2$ . Note that by Lemma 11 we have  $r \notin P_e(T')$  and hence  $f_1$  and  $f_2$  have  $v_1$  in common in  $G'$ , so that  $f_1$  is reassigned to  $v_1$ . This implies that  $P_{u',w'}(T') = P_{u,w}(T)$  holds.

Additionally, there must also exist an edge  $y \in P_{u,w}^{-1}(G, T) \cap E_2$  since  $Y \cap E_2$  is non-empty. However, since we have  $r \in E_1$  and  $r \notin P_{u',w'}(T')$  (since  $P_{u,w}(T) = P_{u',w'}(T')$ ), it follows that  $r \notin P_y(T')$ , which contradicts that  $v$  was a  $Y$ -splittable node which was split using Lemma 11.

We conclude that  $G'$  is indeed 3-connected and simple.  $\square$

From the previous results we obtain a full characterization for the case where  $|\mathcal{V}(\mathcal{T})| \geq 2$ . Lemma 47 shows that  $\mathcal{T}$  represents a graph  $G$  that contains a  $Y$ -splittable vertex  $v$  if and only if we can find a  $Y$ -splittable vertex  $v_\mu$  in the skeleton  $G_\mu$  for each node  $\mu \in \mathcal{V}(\mathcal{T})$  such that we can map all  $v_\mu$  into one vertex  $v$ . In Lemma 48 we show that this is equivalent to finding a  $Y$ -splittable vertex that is incident to all virtual edges  $E_\mu^{\text{virt}}(\mathcal{T})$  for each  $\mu \in \mathcal{V}(\mathcal{T})$ . Lemma 50 shows that  $v$  must be unique, and Theorem 51

then shows that the  $Y$ -processed SPQR tree must consist of a single node of type (R). This implies that all skeletons of  $\mathcal{T}$  are merged into one big skeleton graph  $G'$  in the  $Y$ -processed tree. We can find the specific realization  $G$  where splitting a  $Y$ -splittable vertex leads to a 3-connected graph by mapping all splittable vertices into one node  $v$ , as in Lemma 47, and then applying the neighborhood split to  $v$ .

Although we could apply the splitting to the unique  $Y$ -splittable vertex  $v$  of the merged graph  $G$  to obtain  $G'$ , this may be cumbersome as  $G$  may be quite big. Instead, we can already split  $v_\mu$  locally in each skeleton, and map the resulting two vertices to the two vertices of  $G'$  when performing the contraction outlined in Lemma 47. We prove this claim in Corollary 52.

**Corollary 52.** *Let  $G$  be a 2-connected multigraph that has a 2-separation  $(E_1, E_2)$  with separating vertices  $\{u, w\}$ , where  $u$  is  $Y$ -splittable. For  $i = 1, 2$ , let  $G_i$  denote the graph with vertex set  $V(E_i)$  and edge set  $E_i \cup \{e_i\}$  for a new edge  $e_i := \{u, w\}$ , and let  $Y_i := Y \cap E_i$ . Let  $G'_i$  be the graph created by splitting  $u$  into  $u_1^i$  and  $u_2^i$  according to the neighborhood split of  $u$ , which is given by  $I_i, J_i \subseteq V(H_{Y_i}^u(G_i))$ . We distinguish between  $I_i$  and  $J_i$  by assuming that the connected component of  $H_{Y_i}^u(G_i)$  that contains  $w$  is in  $I_i$ . Let  $G'$  be the graph formed by merging  $G'_1$  with  $G'_2$ , identifying  $u_1^1$  with  $u_1^2$ ,  $u_2^1$  with  $u_2^2$  and  $w$  with itself and removing  $e_1$  and  $e_2$ . Then  $G'$  is equal to the graph obtained from  $G$  by performing the neighborhood split on  $u$ .*

*Proof.* Lemma 45 shows that  $H_Y^u(G)$  is bipartite if and only if  $H_{Y_i}^u(G_i)$  is bipartite for  $i = 1, 2$ . Let  $h_w$  be the component of  $H_Y^u(G)$  containing vertex  $w$  and let, for  $i = 1, 2$ ,  $h_w^i$  be the component of  $H_{Y_i}^u(G_i)$  containing vertex  $w$ . Then the proof of Lemma 45 shows that  $I = I_1 \cup I_2 \cup \{h_w\} \setminus \{h_w^1, h_w^2\}$  and  $J = J_1 \cup J_2$  hold.

In particular, we observe that

$$\delta^I(u) := \{\{u, v\} \in \delta(u) \mid \text{either } \{u, v\} \in Y \text{ or there exists } h \in I \text{ with } u \in h\}$$

can be partitioned as  $\delta^I(u) = \bigcup_{i=1,2} \delta^{I_i}(u) \setminus \{e_i\}$ , where

$$\delta^{I_i}(u) = \{\{u, v\} \in \delta(u) \cap E_i \mid \text{either } \{u, v\} \in Y_i \text{ or there exists } h \in I_i \text{ with } u \in h\}.$$

Similarly, we find that  $\delta^J(u) = \delta^{J_1}(u) \cup \delta^{J_2}(u)$ . Since  $\delta^{I_i}$  and  $\delta^{J_i}$  describe the neighborhood split of  $G_i$ , this shows that the neighborhood split of  $G$  can be found by performing the neighborhood split on  $G_1$  and  $G_2$ , identifying  $u_i^1$  with  $u_i^2$  for  $i = 1, 2$  and  $w$  with itself and removing  $e_1$  and  $e_2$ . This concludes the proof.  $\square$

First, we show that **SPLITSKELETON** correctly finds and splits the  $Y$ -splittable vertices that are incident to each virtual edge.

**Lemma 53.** *Let  $\mathcal{T}$  be a  $Y$ -reduced tree with  $|\mathcal{V}(\mathcal{T})| \geq 2$ . For any  $\mu \in \mathcal{V}(\mathcal{T})$ , let  $(\mathcal{T}', X) := \text{SPLITSKELETON}(\mathcal{T}, \mu, Y)$  and let  $G_\mu$  and  $G'_\mu$  be the skeleton of  $\mu$  in  $\mathcal{T}$  and  $\mathcal{T}'$ , respectively. Then the following hold:*

1.  $X = \emptyset$  holds if and only if  $G_\mu$  does not contain a  $(Y \cap E_\mu)$ -splittable vertex that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ .
2. If  $X \neq \emptyset$  holds, then  $X = \{v_1, v_2\}$  and  $G'_\mu$  is the graph obtained from the neighborhood split of a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ .

*Proof.* In **SPLITSKELETON**, the sub-routine **FINDTREESPLITTABLEVERTICES** first finds all  $Y$ -splittable vertices by Lemma 42, and then checks if any intersects all virtual edges  $E_\mu^{\text{virt}}(\mathcal{T})$ . Note that  $A \neq \emptyset$  holds if  $\mu$  is of type (P), since the parallel structure implies that both end-vertices are always  $Y$ -splittable and incident to each (virtual) edge. Since  $|\mathcal{V}(\mathcal{T})| \geq 2$  holds,  $\mu$  cannot be of type (Q). If  $\mu$  is of type (R) or (S), then we return  $X = \emptyset$  when  $A = \emptyset$  and return  $X \neq \emptyset$  otherwise. This shows the first statement.

For the second point, let us assume  $X \neq \emptyset$ . If  $\mu$  is of type (S), then it has at least one virtual edge, which implies that the case  $A = V(G_\mu)$  does not occur due to  $|E_\mu| \geq 3$ . If  $\mu$  is of type (R), then the case  $|A| = 2$  cannot occur; if  $\mu$  is a leaf of  $\mathcal{T}$ , then Lemma 26 was not applicable, and Lemma 27 then shows that at most one vertex incident to a virtual edge can be  $Y$ -splittable, as otherwise  $\mu$  could have been reduced. If  $\mu$  is not a leaf of  $\mathcal{T}$ , then there is at most one vertex in  $G_\mu$  that is incident to all virtual edges  $E_\mu^{\text{virt}}(\mathcal{T})$  since  $G_\mu$  is simple. Since  $A \neq \emptyset$ , we must have  $|A| = 1$ . Finally, if  $\mu$  is of type (P), then both vertices in  $G_\mu$  are trivially  $Y$ -splittable and incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ . Hence, in all cases **SPLITSKELETON** runs **BIPARTITESPLIT** $(\mathcal{T}, \mu, Y, v_\mu)$  on some  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ , which shows the second statement.  $\square$

We are ready to state the full algorithm for the case where the  $Y$ -reduced tree  $\mathcal{T}$  has  $|\mathcal{V}(\mathcal{T})| > 1$ . **MERGETREE** contains the proposed merging algorithm. Fig. 9 contains an example of the execution of **MERGETREE** on the  $Y$ -reduced tree from Fig. 7f.

---

**Algorithm 9:** Merging  $\mathcal{T}_R$  into a single node of type (R)
 

---

**Algorithm:** MERGETREE( $\mathcal{T}, \mathcal{T}_R, Y_R$ )

**Input:** Reduced SPQR tree  $\mathcal{T}$ , edges  $Y$ 
**Output:**  $Y$ -processed SPQR tree  $\mathcal{T}$ , vertex set  $X'$  such that  $X' = \{v_1, v_2\}$  or  $X' = \emptyset$ 

- 1 Let  $\mu_1, \mu_2, \dots, \mu_n$  be an ordering of  $\mathcal{V}(\mathcal{T})$  such that  $\mu_1$  is a leaf and every other  $\mu_i$  is adjacent to exactly one node  $\mu_j$  with  $j < i$ .
  - 2 Let  $(\mathcal{T}', X') := \text{SPLIT SKELETON}(\mathcal{T}, \mu_1, Y)$
  - 3 **if**  $X' = \emptyset$  **then return**  $\mathcal{T}, \emptyset$
  - 4 **for**  $i = 2, \dots, n$  **do**
  - 5 Let  $(\mathcal{T}', X) := \text{SPLIT SKELETON}(\mathcal{T}', \mu_i, Y)$
  - 6 **if**  $X = \emptyset$  **then return**  $\mathcal{T}, \emptyset$
  - 7 Let  $e, f$  denote the virtual edge pair connecting  $\mu_1$  with  $\mu_i$ .
  - 8 Merge  $\mu_i$  into  $\mu_1$ , identifying  $X' \cup f$  with  $X \cup e$  such that  $f \cap X'$  is identified with  $e \cap X$ ,  $f \setminus X'$  is identified with  $e \setminus X$  and  $X' \setminus f$  is identified with  $X \setminus e$ .
  - 9 Remove the virtual edges  $e$  and  $f$  from  $\mathcal{T}'$ ,  $T_{\mu_1}$  and  $T_{\mu_i}$ .
  - 10 Update  $T_{\mu_1} := T_{\mu_1} \cup T_{\mu_i}$ .
  - 11 **end**
  - 12 Change  $\mu_1$  to type (R).
  - 13 **return**  $(\mathcal{T}', X)$
- 

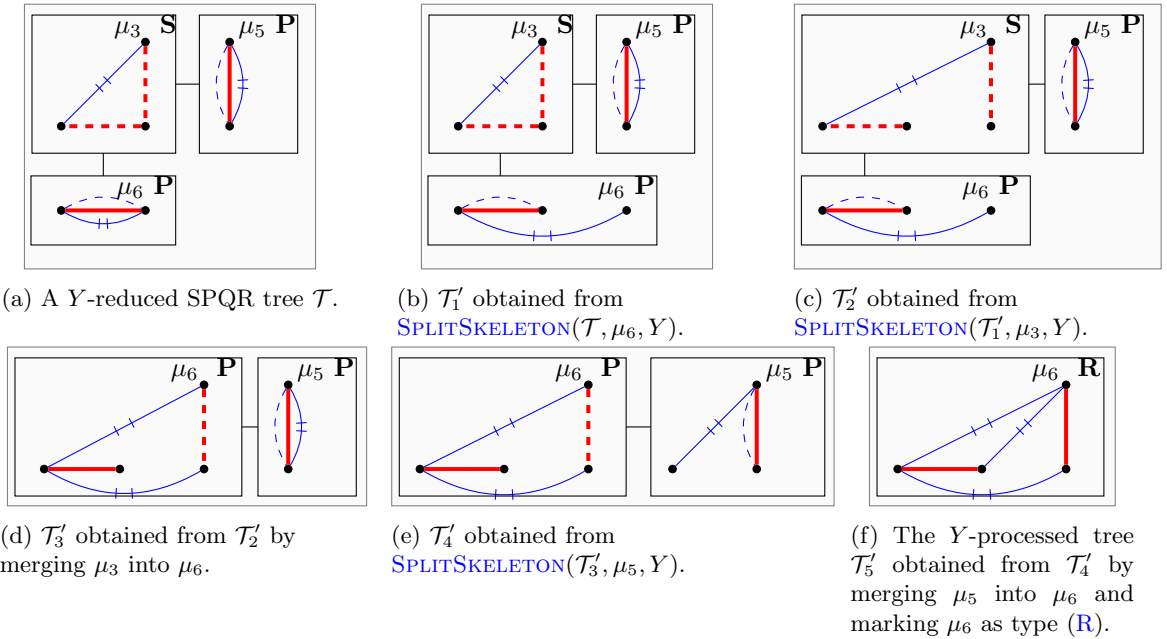


Figure 9: A run of **MERGETREE** on the  $Y$ -reduced tree from Figure 7f. Virtual edges in the SPQR tree are given by dashed edges, tree edges are marked in red and bold and all other edges are marked in blue. Edges in  $Y$  are marked by two stripes.

Finally, we show that **MERGETREE** indeed returns a  $Y$ -processed tree  $\mathcal{T}$ , if one exists.

**Theorem 54.** Let  $\mathcal{T}$  be a  $Y$ -reduced SPQR tree with  $|\mathcal{V}(\mathcal{T})| \geq 2$ , and consider the pair  $(\mathcal{T}', X') := \text{MERGETREE}(\mathcal{T}, Y)$ . If  $X' = \emptyset$ , then  $\mathcal{T}$  does not represent a graph that contains a  $Y$ -splittable vertex. Otherwise, there exist distinct vertices  $v_1$  and  $v_2$  such that  $X' = \{v_1, v_2\}$  holds, and  $\mathcal{T}'$  is a  $Y$ -processed SPQR tree of  $\mathcal{T}$  with respect to  $v_1$  and  $v_2$ .

*Proof.* Let  $\overline{\mathcal{T}'}$  denote  $\mathcal{T}'$  with additional edge  $e'$  between  $v_1$  and  $v_2$ . We note that, by the structure of **MERGETREE**, either  $X' = \emptyset$  or  $\mathcal{V}(\overline{\mathcal{T}'}) = \{\mu\}$  must hold, where  $\mu$  is of type (R) after adding  $e'$  between  $v_1$  and  $v_2$ .

If  $X' = \emptyset$ , this can only occur if  $\text{SPLIT SKELETON}(\mathcal{T}, \mu, Y)$  returns  $X = \emptyset$  for some  $\mu \in \mathcal{V}(\mathcal{T})$ . Lemma 53 implies that  $G_\mu$  does not contain a  $(Y \cap E_\mu)$ -splittable vertex that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ . Thus, Lemma 48 shows that  $\mathcal{T}$  does not represent any graph that contains a  $Y$ -splittable vertex.

If  $X' \neq \emptyset$ , then applying Lemma 53 shows that, for each  $\mu \in \mathcal{V}(\mathcal{T})$ , the skeleton  $G_\mu$  contains a  $(Y \cap E_\mu)$ -splittable vertex that is incident to each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$ . Then Lemma 48 shows that  $\mathcal{T}$  represents a graph  $G$  that contains a  $Y$ -splittable vertex  $v$ . Lemma 50 shows that this  $Y$ -splittable vertex must be unique in  $G$ , and then it follows from Theorem 51 that the graph  $G'$  obtained after performing the neighborhood split and adding the new edge must indeed be a skeleton of type (R) in  $\overline{\mathcal{T}'}$ . In particular, we have a graph-tree pair  $(G, T)$  that is represented by  $\mathcal{T}$  with a  $Y$ -splittable node  $v$ . Then, we could simply apply the split construction from Lemma 11 to obtain the updated graph-tree pair  $(G', T')$ . Then,

in order to apply Lemma 34 to show property (b) of Definition 33, we only need to show that the final graph  $G_{\mu_1}$  obtained by MERGETREE is equal to such a  $G'$ .

Lemma 53 shows that SPLITsKELETON( $\mathcal{T}, \mu, Y$ ) returns a graph  $G'_\mu$  obtained from the neighborhood split of a  $(Y \cap E_\mu)$ -splittable vertex  $v_\mu$  that is incident to each virtual edge in  $G_\mu$ . In MERGETREE we exploit Corollary 52 to combine these neighborhood splits. Note that  $X \cap e$  and  $X' \cap f$  are well-defined and have size 1 because each virtual edge in  $E_\mu^{\text{virt}}(\mathcal{T})$  must be incident to  $v_\mu$ . Since the neighborhood split of a vertex moves each virtual edge to be incident to exactly one vertex of  $\{v_1, v_2\}$ ,  $X' \setminus f$  and  $X \setminus e$  also are single-vertex sets. It follows that  $e \setminus X$  and  $f \setminus X'$  are also single-vertex sets and well-defined. By identifying  $X \cap e$  with  $X' \cap f$ , we exactly pick the realization  $G$  that maps all  $Y \cap E_\mu$ -splittable vertices  $v_\mu$  into one vertex. By using a connected ordering of  $\mathcal{V}(\mathcal{T})$ , we ensure that the conditions of Corollary 52 hold in subsequent steps, by maintaining one neighborhood split in  $\mu_1$ . Then, after MERGETREE terminates,  $G_{\mu_1}$  must contain the neighborhood split of the graph obtained by merging all  $v_\mu$  into a single vertex  $v$ . Thus,  $G' = G_{\mu_1}$  holds and  $\overline{\mathcal{T}'}$  does indeed represent  $G'$ , which shows property (b) of Definition 33.

Clearly,  $\overline{\mathcal{T}'}$  is a minimal SPQR tree since it consists of a single node of type (R) as shown by Theorem 51. Thus,  $\mathcal{T}'$  is a  $Y$ -processed tree with respect to  $v_1$  and  $v_2$ .  $\square$

## 6 Overall algorithm

In the previous sections we discussed how, given a new row  $b$  to add to the graphic matrix  $M$ , one can reduce and update an SPQR tree  $\mathcal{T}$  in order to represent the matrix  $M' = \begin{bmatrix} M \\ b^T \end{bmatrix}$ . Since this is a procedure that we would like to be able to repeat, the SPQR tree  $\mathcal{T}'$  that we find must be minimal and it must represent  $M'$ . In our algorithm, the SPQR tree  $\mathcal{T}'$  obtained by performing REDUCETREE and SPLITsKELETON or MERGETREE might not explicitly represent  $M'$ , because of the reductions performed in REDUCETREE. In particular, we have only shown that  $\mathcal{T}'$  is  $Y$ -processed with respect to  $\mathcal{T}_R$ , the reduced tree, whereas our actual goal is to obtain a  $Y$ -processed tree with respect to the original tree  $\mathcal{T}$ . In order to do so, we can reverse the reductions that we used to derive  $\mathcal{T}_R$ . We will use  $\widehat{\mathcal{T}}$  to denote the SPQR tree where these reversions have been performed.

Since reversing the reductions is equivalent to reversing the SPQR tree operations of REDUCETREE, we will not describe them in full detail. Theorem 32 shows that the reductions preserved splittability of vertices of any graph represented by  $\mathcal{T}$  in  $\mathcal{T}'$ . Thus, if we perform Lemma 11 on some  $Y$ -splittable vertex  $v$  of a graph  $G'$  represented by  $\mathcal{T}'$  and reverse the reductions, we obtain a graph  $\widehat{G}$  with a set of edges  $\widehat{Y}$  that are elongated, where  $\widehat{Y}$  is precisely the original set of marked edges  $Y$ . In particular,  $\widehat{G}$  is realized by  $\widehat{\mathcal{T}}$ . This is shown in detail by the second statements of Lemmas 22, 23 and 26, which precisely argue that the sequence of performing the reduction, splitting a  $Y$ -splittable vertex and undoing the reduction yields the same graph as when we split the  $Y$ -splittable vertex in the original graph.

We will sketch how  $\widehat{\mathcal{T}}$  can be obtained by reversing the reductions, undoing the operations from REDUCETREE. First, we reverse the local 2-separations that are removed by REDUCESERIES and REDUCEPARALLEL. In  $\widehat{\mathcal{T}}$ , these reductions are reversed by replacing the single edges by virtual edges pointing to a skeleton of type (S) and type (P) for each reduction performed in REDUCESERIES and REDUCEPARALLEL, respectively.

Secondly, we undo the reductions that remove leaf nodes from  $\mathcal{T}$  by simply making the relevant edge  $e$  virtual again and adding back the removed node and its skeleton. Since  $\mathcal{T}'$  is an SPQR tree, it is clear that  $\widehat{\mathcal{T}}$  is too. However, although we have shown that  $\mathcal{T}'$  is minimal,  $\widehat{\mathcal{T}}$  is not necessarily minimal, as reversing the reductions may create new (S)-(S) or (P)-(P) connections. If this occurs, the adjacent nodes can be merged into a single node of type (S) or (P), instead. In Lemma 55, we argue that we need at most one such merge.

**Lemma 55.** *Let  $\mathcal{T}$  be a minimal SPQR tree with a set of  $Y$ -reduced edges that represents a graph  $G$  that contains a  $Y$ -splittable vertex  $v$ . Let  $\mathcal{T}_R, Y_R := \text{REDUCETREE}(\mathcal{T}, Y)$ . If  $\mathcal{V}(\mathcal{T}) = \{\mu\}$ , let  $\mathcal{T}', X := \text{SPLITsKELETON}(\mathcal{T}_R, \mu, Y_R)$ , otherwise let  $\mathcal{T}', X := \text{MERGETREE}(\mathcal{T}_R, Y_R)$ . Let  $\widehat{\mathcal{T}}$  be the SPQR tree obtained by reversing the reductions from REDUCETREE in  $\mathcal{T}'$ . Then,  $\widehat{\mathcal{T}}$  has at most one pair of adjacent (S)-(S) or (P)-(P) nodes.*

*Proof.* First, note that the minimality of  $\mathcal{T}$  implies minimality of  $\mathcal{T}_R$  by Theorem 32, and that minimality of  $\mathcal{T}_R$  implies minimality of  $\mathcal{T}'$  by Theorems 44 and 54.

First, we consider the case where  $|\mathcal{V}(\mathcal{T}_R)| > 1$ . Then,  $\mathcal{T}'$  exists of a single node  $\mu$  of type (R). Assume that  $\widehat{\mathcal{T}}$  is not minimal, such that it contains an edge  $\{\nu, \omega\}$  where  $\nu$  and  $\omega$  are either both of type (S) or (P). First, note that neither  $\nu$  nor  $\omega$  can be equal to  $\mu$ , since they do not have type (R). In particular, this rules out that  $\nu$  or  $\omega$  was added to  $\widehat{\mathcal{T}}$  by reversing a local 2-separation, since these become leaves in  $\widehat{\mathcal{T}}$  that are connected to  $\mu$ . Thus,  $\nu$  and  $\omega$  must both have been reduced using an edge of the SPQR tree  $\mathcal{T}$ . However, this implies that the  $\{\nu, \omega\}$  edge also existed in  $\mathcal{T}$ , which contradicts that  $\mathcal{T}$  was minimal. Thus,  $\widehat{\mathcal{T}}$  must be minimal if  $|\mathcal{V}(\mathcal{T}_R)| > 1$  holds.

Second, we consider the case where  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$ . We consider different cases based on the type of  $\mu$ . Note that  $\mu$  can not be of type (P) by Lemma 31.

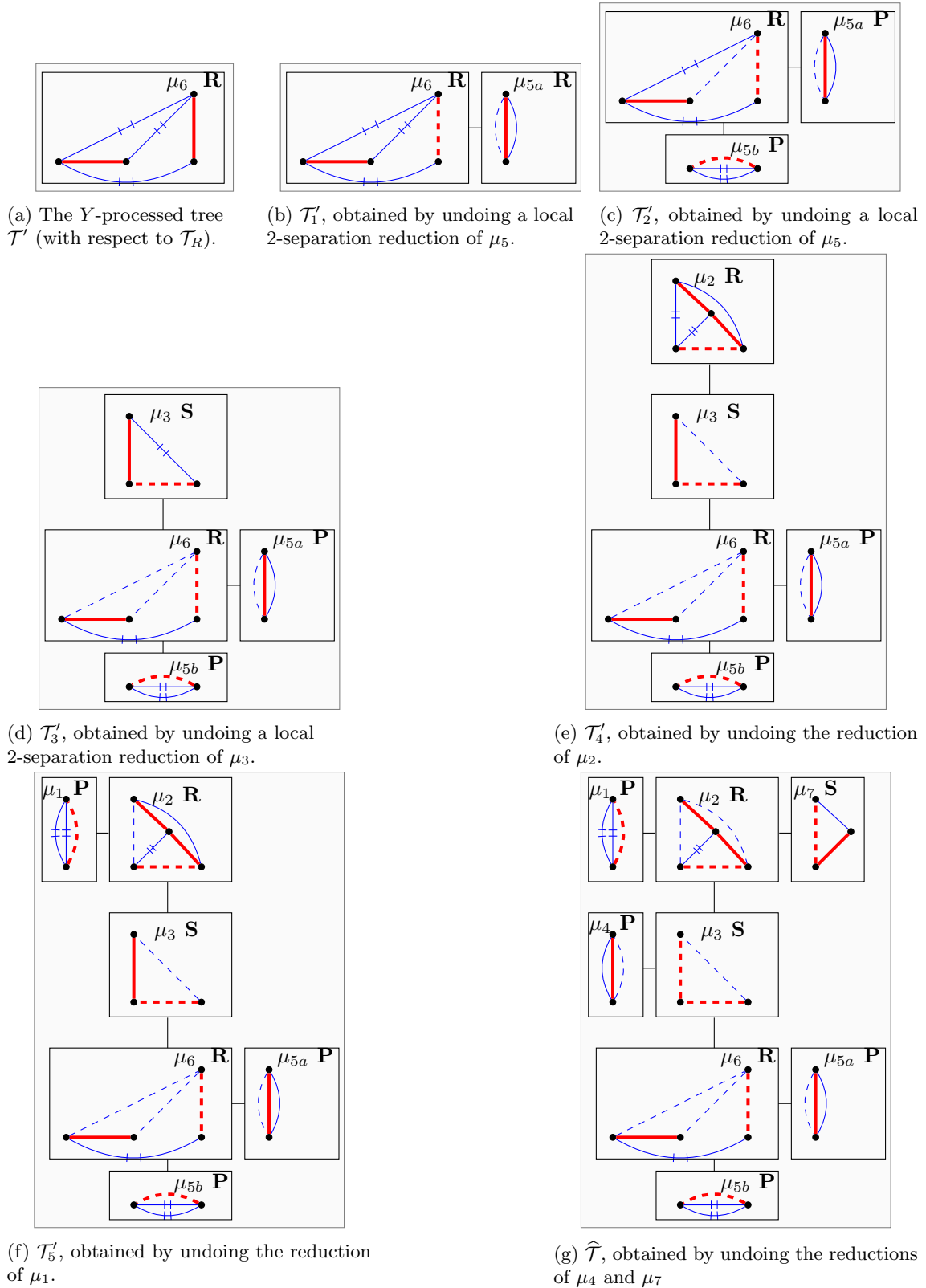


Figure 10: Reversing the reductions from Fig. 7 on the  $Y$ -processed SPQR tree from Fig. 9f. Virtual edges in the SPQR tree are given by dashed edges, tree edges are marked in red and bold and all other edges are marked in blue. Edges in  $Y$  are marked by two stripes.



First, consider the case where  $\mu$  has type (Q) in  $\mathcal{T}_R$ . By Proposition 35,  $\mu$  becomes a node of type (S) in  $\mathcal{T}'$ . If  $\mu$  is of type (Q) in  $\mathcal{T}_R$ , then  $\mu$  must be of type (Q) or (P) in  $\mathcal{T}$ . If  $\mu$  was of type (Q) in  $\mathcal{T}$ , there are no 2-separations that can be reversed, so then  $\widehat{\mathcal{T}}$  is clearly minimal, too, since it consists of a single (S) node.

If  $\mu$  was of type (P) in  $\mathcal{T}$ , at least one of two types of local 2-separations in REDUCEPARALLEL must have been performed, since we know that  $E_\mu^{\text{virt}}(\mathcal{T}_R) = \emptyset$  and  $|E_\mu^{\mathcal{T}}| \geq 3$  hold but  $|E_\mu^{\mathcal{T}_R}| = 2$ . Such a local 2-separation splits off a set of edges  $E'$  from  $\mu$  into a new (P) node  $\nu$ . Since any virtual edges in  $E'$  also exist in  $\mathcal{T}$  and  $\mathcal{T}$  was minimal, any virtual edges in  $E'$  do not connect to nodes of type (P) in  $\widehat{\mathcal{T}}$ . Because  $\mu$  is converted to a node of type (S) in  $\widehat{\mathcal{T}}$ , we create a new (S)-(P) connection this way between  $\mu$  and  $\nu$ , and  $\widehat{\mathcal{T}}$  can thus not have any (P)-(P) connections by reversing the reductions. If both local 2-separations in lines 2 and 5 of REDUCEPARALLEL were performed, then the minimality of  $\widehat{\mathcal{T}}$  follows from the minimality of  $\mathcal{T}$ . Otherwise, exactly one reduction was performed, and there can be one edge  $e' \in E_\mu^{\widehat{\mathcal{T}}}$  that could be a virtual edge to a node of type (S) after reversing the reductions. This implies that  $\widehat{\mathcal{T}}$  contains at most one (S)-(S) connection.

Secondly, we consider the case where  $\mu$  has type (S) in  $\mathcal{T}_R$ . Then, Proposition 35 shows that  $\mu$  stays a node of type (S) in  $\mathcal{T}'$  and  $\widehat{\mathcal{T}}$ . Then, since REDUCESERIES does not perform local 2-separations when  $\mathcal{T}_R$  is a single node of type (S), the node sets  $\mathcal{V}(\mathcal{T})$  and  $\mathcal{V}(\widehat{\mathcal{T}})$  are the same and have identical types. Thus,  $\widehat{\mathcal{T}}$  is minimal because  $\mathcal{T}$  is minimal.

Lastly, consider the case where  $\mu$  has type (R) in  $\mathcal{T}_R$ . If  $|A| = 1$  in SPLITSCHELETON,  $\mathcal{T}'$  consists of a single node of type (R) by Theorem 51, and we must again have that  $\mathcal{V}(\mathcal{T}) = \mathcal{V}(\widehat{\mathcal{T}})$  holds with identical types, since no local reductions were performed because  $\mu$  is of type (R). This implies that  $\widehat{\mathcal{T}}$  is minimal. In the other case, we have  $|A| = 2$ , where an edge  $e$  connects two  $Y$ -splittable vertices in  $\mathcal{T}_R$ . Then,  $\mathcal{T}'$  consists of  $\mu$  and a new node  $\nu$  of type (S), where  $e$  is put in series with the new row edge.

Since we create a new (S)-node, we may create a new (S)-(S) connection in  $\widehat{\mathcal{T}}$ . In particular, this case occurs if  $e$  is a virtual edge pointing to a node of type (S) in  $\mathcal{T}$ . Since  $e$  is the only edge in  $\nu$  that can be a virtual edge that is not pointing to  $\mu$ , at most one (S)-(S) connection can be created in this way.  $\square$

In PROCESSTREE, we present the complete algorithm for updating a single SPQR tree. In the last step of PROCESSTREE, we reverse the reductions performed in REDUCETREE and ensure that  $\widehat{\mathcal{T}}$  remains minimal, by merging two adjacent (S) or (P) nodes if necessary.

---

**Algorithm 10:** Processing a single SPQR tree  $\mathcal{T}$

---

**Algorithm:** PROCESSTREE( $\mathcal{T}, Y$ )  
**Input:** Minimal SPQR tree  $\mathcal{T}$ , edges  $Y \neq \emptyset$   
**Output:**  $Y$ -processed SPQR tree  $\widehat{\mathcal{T}}$ , vertex set  $X$  such that  $X = \{v_1, v_2\}$  or  $X = \emptyset$

- 1  $\mathcal{T}_R, Y_R := \text{REDUCETREE}(\mathcal{T}, Y)$
- 2 **if**  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$  **then**
- 3      $\mathcal{T}', X := \text{SPLITSCHELETON}(\mathcal{T}_R, \mu, Y_R)$
- 4 **else**
- 5      $\mathcal{T}', X := \text{MERGETREE}(\mathcal{T}_R, Y_R)$
- 6 **end**
- 7 **if**  $X = \emptyset$  **then**
- 8     **return**  $\mathcal{T}, \emptyset$
- 9 **end**
- 10 Let  $\widehat{\mathcal{T}}, \widehat{X}$  be obtained by reversing the reductions that reduced  $\mathcal{T}$  to  $\mathcal{T}_R$ .
- 11 **return**  $\widehat{\mathcal{T}}, \widehat{X}$

---

**Corollary 56.** *Let  $\mathcal{T}'$  be a  $Y$ -processed tree with respect to vertices  $v_1$  and  $v_2$  obtained by PROCESSTREE. Then  $v_1$  and  $v_2$  never lie in a node of type (P).*

*Proof.* First, we note that  $\mathcal{T}'$  was obtained using some  $Y$ -reduced tree  $\mathcal{T}_R$ . By Theorem 32, we cannot have  $\mathcal{V}(\mathcal{T}_R) = \{\mu\}$  where  $\mu$  is of type (P). If  $|\mathcal{V}(\mathcal{T}_R)| > 1$ , MERGETREE ensures that  $v_1$  and  $v_2$  must be in a skeleton of type (R). Otherwise, SPLITSCHELETON only places  $v_1$  and  $v_2$  in nodes of type (S) or (R).  $\square$

In Section 2 we argued that it is sufficient to consider the connected components of  $M$  separately, as we could easily connect them in a realization by identifying the new row edges of each component with one another. Since we have an SPQR tree for every connected component of  $M$ , we actually maintain an SPQR forest consisting of the individual trees for each component. Interpreting Theorem 3 in terms of SPQR trees, we observe that combining multiple connected components in  $M$  can be thought of as creating a new node  $\mu$  of type (P), that connects  $\mu$  with a virtual edges in between the vertices of  $X = \{v_1, v_2\}$  for each  $Y$ -processed SPQR tree. Additionally, it may be the case that we encounter a nonzero in a column for the first time. Clearly, the resulting edge of such a column must always be placed in parallel with the row edge. These cases are handled by our main algorithm, GRAPHICROWAUGMENTATION.

---

**Algorithm 11:** Processing a single row

---

**Algorithm:** GRAPHICROWAUGMENTATION( $M, b, \mathcal{F}$ )**Input:** Graphic Matrix  $M$ , new row  $b$ , SPQR forest  $\mathcal{F}$ **Output:** SPQR forest  $\mathcal{F}'$  representing  $M' = \begin{bmatrix} M \\ b^T \end{bmatrix}$  or FALSE if  $M'$  is not graphic

```
1 Let  $Y := \text{supp}(b) \cap E^{\text{reg}}(\mathcal{F})$  and let  $Y' := \text{supp}(b) \setminus E^{\text{reg}}(\mathcal{F})$ .
2 Let  $Y_{\mathcal{T}} := Y \cap E^{\text{reg}}(\mathcal{T})$  for each SPQR tree  $\mathcal{T} \in \mathcal{F}$ .
3 Let  $\mathcal{F}_Y := \{\mathcal{T} \in \mathcal{F} \mid Y_{\mathcal{T}} \neq \emptyset\}$ .
4 if  $\mathcal{F}_Y = \{\mathcal{T}\}$  then
5    $\mathcal{T}', X := \text{PROCESSTREE}(\mathcal{T}, Y_{\mathcal{T}})$ 
6   if  $X = \emptyset$  then return FALSE
7   if  $Y' = \emptyset$  then
8     Add  $b$  between  $v_1$  and  $v_2$ .
9   else
10    Create a new skeleton  $\mu$  of type (P) with edges  $Y' \cup \{b\}$  in  $\mathcal{T}'$ .
11    Connect  $\mu$  and  $\{v_1, v_2\}$  in  $\mathcal{T}'$  using a virtual edge pair.
12  end
13   $\mathcal{F}' := (\mathcal{F} \cup \{\mathcal{T}'\}) \setminus \{\mathcal{T}\}$ 
14 else if  $|\mathcal{F}_Y| > 1$  then
15   Let  $\mathcal{T}_{\text{new}}$  be an SPQR tree with a single member  $\mu$  of type (P) with edges  $Y' \cup \{b\}$ .
16   for  $\mathcal{T} \in \mathcal{F}_Y$  do
17      $\mathcal{T}', X := \text{PROCESSTREE}(\mathcal{T}, Y_{\mathcal{T}})$ 
18     if  $X = \emptyset$  then return FALSE
19     Update  $\mathcal{T}_{\text{new}}$  by connecting  $\mathcal{T}_{\text{new}}$  with  $\mathcal{T}'$ , by connecting  $\mu$  and  $v_1$  and  $v_2$  using a virtual edge pair.
20   end
21    $\mathcal{F}' := (\mathcal{F} \cup \{\mathcal{T}_{\text{new}}\}) \setminus \mathcal{F}_Y$ 
22 else
23   Create an SPQR tree  $\mathcal{T}$  with a single skeleton  $\mu$  of type (Q) (if  $|Y'| = 1$ ) or (P) with edges  $Y' \cup \{b\}$ .
24    $\mathcal{F}' := \mathcal{F} \cup \{\mathcal{T}\}$ 
25 end
26 return  $\mathcal{F}'$ 
```

---

**Corollary 57.** *Let  $\mathcal{F}$  be a minimal SPQR forest. Then every SPQR tree in  $\mathcal{F}'$  returned by GRAPHICROWAUGMENTATION is minimal.*

*Proof.* In the case where  $\mathcal{F}_Y \neq \emptyset$  we always obtain a new SPQR tree  $\mathcal{T}_{\text{new}}$  by from a  $Y$ -processed SPQR tree  $\mathcal{T}'$  with respect to vertices. In particular, we either connect  $v_1$  and  $v_2$  with row edge or we connect  $v_1$  and  $v_2$  using a virtual edge pointing to a new node  $\mu$  of type (P). By Corollary 56,  $v_1$  and  $v_2$  can not lie in a node of (P), and we can thus not create a new (P)-(P) connection in this manner. Then, since  $\mathcal{T}'$  is  $Y$ -processed, the SPQR tree  $\mathcal{T}_{\text{new}}$  obtained by these operations is minimal. In the case where  $|\mathcal{F}_Y| > 1$ , we identify these minimal SPQR trees in  $\mu$ , which clearly yields another minimal SPQR tree. In the case where  $\mathcal{F}_Y = \emptyset$ , we simply add a new SPQR tree to  $\mathcal{F}'$  with a single node of type (Q) or (P), which is clearly minimal.  $\square$

Although our algorithmic description both detects graphicness of adding the row and updates the SPQR tree at the same time, we note that these steps are easy to separate within an implementation. Similarly, although the SPQR tree is frequently copied throughout our pseudocode, it is easy to avoid these copies in an implementation. In particular, both the reductions from REDUCETREE and the update steps in SPLITSKELTON and MERGETREE can be performed on a single SPQR tree by editing a sub SPQR tree in-place, and the reversals of the reductions are then performed implicitly. We have intentionally omitted these details from the algorithmic description to keep it simpler.

As we aim to make our row-wise algorithm compatible with the column-wise algorithm formulated by Bixby and Wagner [6], the merging of two skeletons  $G_\mu$  and  $G_\nu$  should happen in a minimum amount of time. In particular, Bixby and Wagner use *disjoint set* data structure [29] (also referred to as *union-find*) and doubly linked-lists in order to represent the skeletons  $G_\mu$ . A sequence of  $m$  disjoint set operations on a set with  $n$  elements can be done in  $\mathcal{O}(m\alpha(n))$  time [30]. This can be seen as being ‘almost linear’ in  $m$  since in practice,  $\alpha(n) \leq 4$  holds for any realistic input that can be described on a computer.

In an implementation of the algorithms described in our paper, we use the following data structures:

- The matrix is assumed to be a compressed sparse matrix given in row-major format, so that we can obtain each row with  $k$  non-zero entries in  $\mathcal{O}(k)$  time.
- There is an explicit mapping from matrix rows and columns to edge labels
- The SPQR tree node labels  $V(\mathcal{T})$  are given by a disjoint set data structure.
- The skeleton vertex labels  $V(G_\mu)$  are given by a disjoint set data structure over all vertices of the SPQR tree  $\mathcal{T}$ .
- The neighboring edges  $\delta(v)$  of vertices  $v \in G_\mu$  link to each other using a doubly-linked list.
- Each edge stores the SPQR tree node label it is initially located in. In order to find the edge’s current skeleton, a *find*-operation on the disjoint set data structure of the SPQR tree node labels is carried out.

- Each edge stores its initial end-vertex labels. Similarly to the above, a *find*-operation is necessary to find the correct adjacent vertices.
- For each skeleton, we store its type (among (S), (P), (Q) and (R))
- Each skeleton stores its edges in  $Y$  using a doubly-linked list.
- We distinguish different SPQR trees in the SPQR forest using a disjoint set data structure over the SPQR tree node labels.

Moreover, we pick an arbitrary root node and represent  $\mathcal{T}$  as a rooted arborescence. Each SPQR tree node  $\mu$ , except for the root, stores its parent  $\nu$ , and each ‘child’ virtual edge  $e$  stores the corresponding virtual edge  $f$ . The corresponding child member can be found by using a *find*-operation on the disjoint set data structure of SPQR tree node labels.

Additionally, we note that it is not always necessary to explicitly maintain the skeleton graph. In particular, the skeletons of type (S), (P) or (Q) always have the same structure, which means that it is unnecessary to track the end-vertices of all edges. This is particularly convenient for (S)-nodes, whose edges may be arbitrarily permuted.

Let  $m$  and  $n$  be the number of rows and columns of the matrix respectively. First, we show that the size of an SPQR tree is linear with respect to the size of the graph that it represents. We can easily extend this notion to SPQR forests, and show that the SPQR forest representing a  $m \times n$  matrix always has linear size. Note that any realization of a graphic  $m \times n$  matrix is a graph  $G = (V, E)$  with  $|E| = m + n$  edges.

**Proposition 58** (Hopcroft & Tarjan[19]). *Let  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$  be a minimal SPQR tree of a 2-connected graph  $G = (V, E)$  with  $|E| \geq 3$ . Then, the total number  $\sum_{\mu \in \mathcal{V}} |E_\mu|$  of edges in the skeletons of  $\mathcal{T}$  is at most  $3|E| - 6$ .*

**Proposition 59.** *Let  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$  be a minimal SPQR tree of a 2-connected graph  $G = (V, E)$  with  $|E| \geq 3$ . Then, we have  $|\mathcal{V}| \leq |E| - 2$  for the number of skeletons of  $\mathcal{T}$ . For the total number of vertices in the skeletons of  $\mathcal{T}$ ,  $\sum_{\mu \in \mathcal{V}} |V_\mu| \leq 3|E| - 6$  holds.*

*Proof.* Since  $|E| \geq 3$  holds,  $\mathcal{T}$  consists only of nodes that are not of type (Q). Thus, each skeleton  $G_\mu = (V_\mu, E_\mu)$  then requires  $|E_\mu| \geq 3$  by definition. For the first point, Proposition 58 implies that  $|\mathcal{V}| \leq \frac{3|E| - 6}{3} = |E| - 2$ . For the second point, we have that  $|V_\mu| \leq |E_\mu|$  since  $G_\mu$  is 2-connected and  $|E_\mu| \geq 3$ . Then, Proposition 58 implies that  $\sum_{\mu \in \mathcal{V}} |V_\mu| \leq \sum_{\mu \in \mathcal{V}} |E_\mu| \leq 3|E| - 6$ .  $\square$

Using the above-mentioned data structures and bounds on the SPQR tree size, we consider basic operations and their time complexities:

- Split a vertex  $v$  into two vertices and reassign its edges in  $\mathcal{O}(|\delta(v)|)$  time.
- Merge two vertices into one (identifying them) in  $\mathcal{O}(\alpha(m+n))$  amortized time.
- Merge two SPQR tree node labels  $\mu_1$  and  $\mu_2$  in  $\mathcal{O}(\alpha(m+n))$  amortized time.
- Find (one of) the end-vertices of an edge  $e$  in  $\mathcal{O}(\alpha(m+n))$  amortized time.
- Add a vertex or edge to a skeleton in  $\mathcal{O}(1)$ .
- Add a new node with an empty skeleton to  $\mathcal{T}$  in  $\mathcal{O}(1)$  time.
- Given a virtual edge  $e \in G_\mu$  that is paired with  $f \in G_\nu$ , find  $\nu$  in  $\mathcal{O}(\alpha(m+n))$  amortized time.
- Find the SPQR tree  $\mathcal{T}$  that contains an SPQR forest node  $\mu \in V(\mathcal{F})$  in  $\mathcal{O}(\alpha(m+n))$  time.

Note that for many operations, we incur a  $\alpha(m+n)$  overhead, that depends on the size of the whole matrix ( $m+n$ ), even though the size of the SPQR tree in which they occur might be smaller, if the matrix has multiple blocks. In particular, this is because the used union-find data structures represent the whole matrix, and not just a single SPQR tree.

Let us analyze the time complexity of the given algorithm in a bottom-up manner.

**Lemma 60.** `FINDSPLITTABLEVERTICES`( $\mathcal{T}, \mu, Y$ ) runs in  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  time

*Proof.* The check in line 1 can be done in  $\mathcal{O}(1)$  time. In line 2 we find the set of incident vertices, which can be done in  $\mathcal{O}(\alpha(m+n)|Y|)$  time. Note that the  $\alpha(m+n)$  factor comes from the fact that we need to perform *find*-operations for every end-vertex of the edge.

The intersection of all paths in line 4 is more complicated. We find the intersection of all paths in  $Y$  using so-called *lowest common ancestor* (LCA) queries. First, we pick an arbitrary root vertex of  $T_\mu$  and orient  $T_\mu$  so that  $T_\mu$  becomes a rooted tree. Then we can answer LCA queries on vertices of  $T_\mu$  in  $\mathcal{O}(1)$  by doing a  $\mathcal{O}(|T_\mu|)$  preprocessing phase (see [17, 4]). Using a constant number of LCA queries, we can find the intersection of two paths  $P_y(T_\mu)$  and  $P_{y'}(T_\mu)$  in  $\mathcal{O}(1)$ . Since the intersection of two paths is again a path, we can repeat this procedure ( $|Y| - 1$ )-times in total to find the intersection of all paths. Thus, the total time complexity for this step is  $\mathcal{O}(|T_\mu| + |Y|)$ . Note that by using the LCA queries, the final path is not given explicitly, but rather it is given by its two end-vertices in  $T_\mu$ . However, we can again efficiently check in  $\mathcal{O}(1)$  time if a given vertex  $v$  is on this path using LCA queries. In the above, we assumed constant time vertex queries from the edges; since we do not have this luxury, the time complexity becomes  $\mathcal{O}(\alpha(m+n)(|T| + |Y|))$ .

The articulation vertices of  $G_\mu$  can be found in  $\mathcal{O}(|E_\mu| + |V_\mu|)$  using a depth first search [28]. However, this is assuming that the end-vertices of an edge can be queried in time  $\mathcal{O}(1)$ . Since we need to perform an additional *find*-operation for each query, we have a  $\mathcal{O}(\alpha(m+n)(|E_\mu| + |V_\mu|))$  running time. Constructing  $H_Y^a$ , which involves finding the connected components of  $G_Y^a$ , can similarly be done in  $\mathcal{O}(\alpha(m+n)(|E_\mu| + |V_\mu|))$  time, and determining if  $H_Y^a$  is bipartite can also similarly be done using a depth first search over  $G_Y^a$  in  $\mathcal{O}(\alpha(m+n)(|E_\mu| + |V_\mu|))$ . By Lemma 37, we need to compute  $H_Y^a$  for at most four candidates  $a$ , which establishes the total time complexity for this step.

Since  $Y \subseteq E_\mu$  and  $T_\mu \subseteq E_\mu$  hold, each step of **FINDSPLITTABLEVERTICES** runs in  $\mathcal{O}(\alpha(m+n)(|E_\mu| + |V_\mu|))$  time. The claimed bound follows from 2-connectivity since this implies  $|V_\mu| \leq |E_\mu|$ .  $\square$

**Proposition 61.** **SPLITSKELETON**( $\mathcal{T}, \mu, Y$ ) runs in  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  time.

*Proof.* First, we run **FINDTREESPLITTABLEVERTICES**, which calls **FINDSPLITTABLEVERTICES**. This takes  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  time by Lemma 60. Note that by 2-connectivity at most  $|V_\mu| = \mathcal{O}(|E_\mu|)$  vertices are returned. Then it intersects the resulting vertices with at most  $|E_\mu|$  virtual edges' end-vertices. The intersection of two sets can be found in time linear in their sizes. For the first edge, this takes  $\mathcal{O}(|E_\mu|)$  time. However, since each virtual edge has two vertices, the intersection with the remaining virtual edges takes  $\mathcal{O}(1)$  time for each edge, since our set to intersect will have at most two vertices. Thus, the intersection can be computed in  $\mathcal{O}(|E_\mu|)$  time, which yields a total running time of  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  for **FINDTREESPLITTABLEVERTICES**.

Then let us consider the other operations of **SPLITSKELETON**. In many cases, we execute **EXTENDSERIES**, which runs in constant time since we only split a single vertex with two edges. In all other cases, **BIPARTITESPLIT** runs in  $\mathcal{O}(|\delta(v)|)$  since it reassigns  $|\delta(v)|$  edges to new neighbors.

Clearly, all steps run in  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  time, which proves the desired result.  $\square$

**Lemma 62.** **REDUCETREE**( $\mathcal{T}, Y$ ) runs in  $\mathcal{O}(\alpha(m+n)|E|)$  time.

*Proof.* First, we find the leaves of  $\mathcal{T}$ , which can be done in  $\mathcal{O}(|\mathcal{V}(\mathcal{T})|) = \mathcal{O}(|E|)$  time by Proposition 59. Then we consider the direct requirements for the reduction of each node  $\mu \in \mathcal{V}(\mathcal{T})$ . First,  $Y_\mu = \emptyset$  can be checked in constant time. Second, for the more complicated requirement  $Y_\mu = P_e^{-1}(G_\mu, T_\mu)$ , we distinguish several cases. If  $\mu$  is of type (R) then the condition can be checked in  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  using **FINDSPLITTABLEVERTICES** by Lemma 27 and Lemma 60. If  $\mu$  is of type (P) then we simply need to check if  $|Y_\mu| = |E_\mu| - 1$  holds, i.e., that all non-tree edges are in  $Y_\mu$ , which takes  $\mathcal{O}(|E_\mu|)$  time. Nodes of type (S) are always propagated by Proposition 30, so deciding this takes  $\mathcal{O}(1)$  time. The reductions performed by **REDUCEPARALLEL** and **REDUCESERIES** run in  $\mathcal{O}(|E_\mu|)$  time since they only iterate over the edges and perform set operations. All other (sub-)steps are set operations that can be done in  $\mathcal{O}(1)$  time using the proposed data structures, and happen at most once for each node. Thus, the total time for processing each node  $\mu$  is bounded by  $\mathcal{O}(\alpha(m+n)|E_\mu|)$ . Summing up over all  $\mu \in \mathcal{V}(\mathcal{T})$ , we find that the total run time is bounded by  $\mathcal{O}(\alpha(m+n)|E|)$  using Proposition 58.  $\square$

**Lemma 63.** **MERGETREE**( $\mathcal{T}, Y$ ) runs in  $\mathcal{O}(\alpha(m+n)|E|)$  time.

*Proof.* For each skSee report.leton, we execute **SPLITSKELETON** in  $\mathcal{O}(\alpha(m+n)|E_\mu|)$  time by Proposition 61. Summing over all  $\mu \in \mathcal{V}(\mathcal{T})$ , we then find using Proposition 58 that this takes  $\mathcal{O}(\alpha(m+n)|E|)$  in total. Merging two skeletons into one, which includes identification of a constant number of vertices and removal of the virtual edges, takes  $\mathcal{O}(\alpha(m+n))$  amortized time. We do this at most  $|\mathcal{V}(\mathcal{T})| - 1$  times, so that we obtain a  $\mathcal{O}(\alpha(m+n)|\mathcal{V}(\mathcal{T})|)$  bound. By Proposition 59, this is equivalent to  $\mathcal{O}((m+n)|E|)$ . Because  $X$  has size 2, all other set operations in the merging loop have constant time complexity and thus take  $\mathcal{O}(|\mathcal{V}(\mathcal{T})|) = \mathcal{O}(|E|)$  time in total. Thus, all steps can be done in  $\mathcal{O}(\alpha(m+n)|E|)$  time, which concludes our proof.  $\square$

**Lemma 64.** **PROCESSTREE**( $\mathcal{T}, Y$ ) runs in  $\mathcal{O}(\alpha(m+n)|E|)$  time.

*Proof.* First, **REDUCETREE** runs in  $\mathcal{O}(\alpha(m+n)|E|)$  time by Lemma 62, and we obtain an SPQR tree  $\mathcal{T}_R$  that represents a graph  $G_R = (V_R, E_R)$  where  $|V_R| \leq |V|$  and  $|E_R| \leq |E|$ . In the case  $|\mathcal{V}(\mathcal{T}_R)| = 1$  holds, **SPLITSKELETON** runs in  $\mathcal{O}(\alpha(m+n)(|E_\mu|))$  time by Lemma 60, which is dominated by  $\mathcal{O}(\alpha(m+n)|E_R|)$ . Otherwise, **MERGETREE** runs in  $\mathcal{O}(\alpha(m+n)|E_R|)$  time by Lemma 63. Then finally, we can reverse the reductions from **REDUCETREE** in the same time as it took to perform them. To restore minimality, Lemma 55 shows we need at most one merge of adjacent nodes in  $\mathcal{T}'$  which takes  $\mathcal{O}(\alpha(m+n))$  time. Clearly, all steps are dominated by the given time complexity  $\mathcal{O}(\alpha(m+n)|E|)$ .  $\square$

Considering the time bounds, we can roughly distinguish three cases in which the worst case time complexity is attained. The first occurs when  $\mathcal{T}$  consists of a single large node (of type (R)), where **FINDSPLITTABLEVERTICES** is responsible for the time bound. The second case occurs when the  $Y$ -reduced tree  $\mathcal{T}$  has a large number of nodes that need to be merged into one single large node, where the time bound is given by the identification of these nodes in **MERGETREE**. Finally, if the initial SPQR tree is large and contains many reductions, **REDUCETREE** attains the worst case time complexity.

**Theorem 65.** **GRAPHICROWAUGMENTATION** runs in  $\mathcal{O}((m+n) \cdot \alpha(m+n))$  time and  $\mathcal{O}(m+n)$  space.

*Proof.* First, we distribute  $\text{supp } b$  over  $Y$  and  $Y'$ , by checking if the associated column edge exists, which takes  $\mathcal{O}(|\text{supp } b|) = \mathcal{O}(n)$  time. Then, for each  $y \in Y$ , we find the corresponding SPQR tree by first finding the corresponding SPQR node in  $\alpha(m+n)$  time, and then finding the corresponding SPQR tree in  $\alpha(m+n)$  time. Thus, this step takes  $\mathcal{O}(n\alpha(m+n))$  time.

Now, consider various cases based on the size of  $\mathcal{F}_Y$ . If  $\mathcal{F}_Y = \emptyset$ , then we simply create a new SPQR tree with  $|Y'| + 1$  edges. Clearly, this can be done in  $\mathcal{O}(n)$  time.

If  $\mathcal{F}_Y = \{\mathcal{T}\}$ , then we call **PROCESSTREE**. Since  $\mathcal{T}$  is a member of the SPQR forest that represents  $M$ , any realization of  $\mathcal{T}$  has at most  $m+n$  edges. Then, **PROCESSTREE** runs in  $\mathcal{O}((m+n) \cdot \alpha(m+n))$  time by Lemma 64. For the other steps in this branch, adding  $b$  and the edges  $Y'$ , and initializing the data structures for the new SPQR takes at most  $\mathcal{O}(n)$  time.

If  $|\mathcal{F}_Y| > 1$ , then we first create a new skeleton with  $|Y'| + 1$  edges, which can be done in  $\mathcal{O}(n)$  time. For all  $\mathcal{T} \in \mathcal{F}_Y$ , let  $G_{\mathcal{T}}$  be any graph represented by  $\mathcal{T}$ . Because  $\mathcal{F}_Y$  is a subset of the SPQR forest representing  $M$ , it must be the case that  $\sum_{\mathcal{T} \in \mathcal{F}_Y} |E(G_{\mathcal{T}})| \leq m+n$  holds. Thus, if we run **PROCESSTREE** for each  $\mathcal{T} \in \mathcal{F}_Y$ , the sum of their time complexities is of order  $\mathcal{O}((m+n)\alpha(m+n))$ . In the loop, we additionally connect  $\mathcal{T}_{new}$  with  $\mathcal{T}$ , which can be done in  $\mathcal{O}(\alpha(m+n))$  time. We do this at most  $n$  times, which gives us a time complexity of  $\mathcal{O}(n\alpha(m+n))$ , which is clearly dominated by the time complexity of running **PROCESSTREE** for each tree.

We obtain that the first 3 lines and every branch of **GRAPHICROWAUGMENTATION** can be done in  $\mathcal{O}((m+n)\alpha(m+n))$  time, which concludes the proof  $\square$

**Corollary 66.** *For a matrix  $M \in \{0,1\}^{m \times n}$ , we can determine its graphicness using repeated calls of **GRAPHICROWAUGMENTATION** in  $\mathcal{O}((m^2 + mn) \cdot \alpha(m+n))$  time.*

*Proof.* We perform  $m$  calls of **GRAPHICROWAUGMENTATION**, which gives us a total time complexity of  $\mathcal{O}((m^2 + mn) \cdot \alpha(m+n))$ .  $\square$

Note that the time complexity we obtain in Corollary 66 is strictly worse than the  $\mathcal{O}(k\alpha(k,m))$  running time that is achieved by Bixby and Wagner.

## 7 Discussion and recommendations for future research

In this paper, we formulated an algorithm for solving the graphic row augmentation problem in  $\mathcal{O}(\alpha(m+n)(m+n))$  time. By adding all rows sequentially we obtain a  $\mathcal{O}((m^2 + mn)\alpha(m+n))$  algorithm to detect whether a matrix  $M$  is graphic. Note that this is strictly worse than the  $\mathcal{O}(k\alpha(k,m))$  running time achieved by Bixby and Wagner. However, we do suspect that the  $\mathcal{O}((m+n)\alpha(m+n))$  running time can be improved. In particular, we suspect one can use (modifications of) the dynamic connectivity data structures from [21] and the dynamic LCA data structure from [27] to speed up **FINDSPLITTABLEVERTICES**, which attains the worst case bound if we have one big component of type (R). Additionally, for the purpose of detecting graphicness of the entire matrix, one can use Proposition 59 to show that at most  $\mathcal{O}(m+n)$  pairs of skeletons in  $\mathcal{T}$  are merged over multiple graphic row additions. This hints that the other worst case where the  $Y$ -reduced SPQR tree  $\mathcal{T}$  has many nodes is somewhat rare when one sequentially adds all rows of a matrix. We hypothesize that using dynamic data structures, an implementation of the described algorithms that is polynomial in  $k$  is possible. However, using dynamic data structures comes at the cost of both space complexity and a more complex algorithmic implementation, which are the reasons why we did not pursue this direction of research further in this work.

Although our method does not reach the best known running time for finding graphic matrices, it is, to the best of our knowledge, the first complete algorithm for the graphic row augmentation problem. The proposed algorithm can be combined with that of Bixby and Wagner [6] in order to find arbitrary graphic submatrices. This can be advantageous for applications in mixed-integer linear programming, where knowledge of (transposed) network submatrices in the problem may be useful to derive stronger cutting planes, primal heuristics or integrality properties. An interesting new research direction would be to investigate the presence of (transposed) network submatrices in mixed-integer linear programming problems, and to examine their properties. An implementation of the row-wise and column-wise algorithm for graphic matrices can be found in the repository: <http://github.com/rolfdhulst/matrec>. As this work primarily investigates the validity of the row-wise algorithm, we consider computational results to be outside its scope. In future work, we plan to experimentally compare both methods and investigate the presence of graphic/network submatrices in mixed-integer programming problems.

Another promising future research direction is to generalize the current results from graphic matrices and undirected graphs to network matrices and directed graphs. In particular, we suspect that there exists an SPQR tree type data structure that uniquely represents directed graphs with the same set of directed cycles. Moreover, we hypothesize that both Bixby and Wagner's column-wise algorithm and our proposed row-wise algorithm can be modified to work in this generalized setting.

Furthermore, because graphic and network matrices are closely related to regular matroids and totally unimodular matrices, this work, together with that of Bixby and Wagner in [6], can be a starting point for the development of matrix augmentation algorithms maintaining regularity or total unimodularity.

## References

- [1] Tobias Achterberg and Christian Raack. The MCF-separator: Detecting and exploiting multi-commodity flow structures in MIPs. *Mathematical Programming Computation*, 2(2):125–165, 2010. doi:10.1007/s12532-010-0015-3.
- [2] Manuel Aprile, Samuel Fiorini, Gwenaël Joret, Stefan Kober, Michał T. Seweryn, Stefan Weltge, and Yelena Yuditsky. Integer programs with nearly totally unimodular matrices: the cographic case, pre-print, 2024. URL: <https://arxiv.org/abs/2407.09477>, arXiv:2407.09477.
- [3] John J. Bartholdi. A good submatrix is hard to find. *Operations Research Letters*, 1(5):190–193, 1982. doi:10.1016/0167-6377(82)90038-4.
- [4] Michael A. Bender and Martín Farach-Colton. The LCA problem revisited. *LATIN 2000: Theoretical Informatics*, 4th Latin American Symposium(April 2000):88–94, 2000. doi:10.1007/10719839.
- [5] Robert E. Bixby and William H. Cunningham. Converting Linear Programs to Network Problems. *Mathematics of Operations Research*, 5(3):321–357, 1980. doi:10.1287/moor.5.3.321.
- [6] Robert E Bixby and Donald K Wagner. An almost linear-time algorithm for graph realization. *Mathematics of Operation research*, 13(1), 1988. doi:10.1287/moor.13.1.99.
- [7] Paul Camion. *Matrices Totalement Unimodulaires et Problemes Combinatoires*. Phd thesis, Université Libre de Bruxelles, 1963.
- [8] William H. Cunningham. Separating cocircuits in binary matroids. *Linear Algebra and Its Applications*, 43(C):69–86, 1982. doi:10.1016/0024-3795(82)90244-0.
- [9] William H. Cunningham and Jack Edmonds. A Combinatorial Decomposition Theory. *Canadian Journal of Mathematics*, 32(3):734–765, 1980. doi:10.4153/cjm-1980-057-7.
- [10] Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS ’89, pages 436–441. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63515.
- [11] Giuseppe Di Battista and Roberto Tamassia. On-line graph algorithms with spqr-trees. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 598–611, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. doi:10.1007/BFb0032061.
- [12] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate texts in Mathematics*. Springer, New York, Heidelberg, 5 edition, 2012. doi:10.1007/978-3-662-53622-3.
- [13] Satoru Fujishige. An efficient PQ-graph algorithm for solving the graph-realization problem. *Journal of Computer and System Sciences*, 21(1):63–86, 1980. doi:10.1016/0022-0000(80)90042-2.
- [14] Fănică Gavril and Robert Tamari. An algorithm for constructing edge-trees from hypergraphs. *Networks*, 13(3):377–388, 1983. doi:10.1002/net.3230130306.
- [15] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, jun 1981. doi:10.1007/BF02579273.
- [16] Carsten Gutwenger and Petra Mutzel. A Linear Time Implementation of SPQR-Trees. In *Graph Drawing*, pages 77–90, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. doi:10.1007/3-540-44541-2\_8.
- [17] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984. doi:10.1137/0213024.
- [18] Alan J. Hoffman and Joseph B. Kruskal. Integral Boundary Points of Convex Polyhedra. *Linear Inequalities and Related Systems*, 38:223–246, dec 1956. doi:10.1515/9781400881987-014.
- [19] John E. Hopcroft and Robert E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.
- [20] Masao Iri. On the Synthesis of Loop and Cutset Matrices and Related Problems. *Research Association of Applied Geometry*, 4:376–410, 1968.
- [21] Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1131–1142, 2013. doi:10.1137/1.9781611973105.81.
- [22] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC ’84, pages 302–311, New York, NY, USA, 1984. ACM. doi:10.1145/800057.808695.
- [23] Richard M. Karp and Christos H. Papadimitriou. On linear characterizations of combinatorial optimization problems. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 1–9. IEEE, 1980. doi:10.1137/0211053.
- [24] Manfred W. Padberg and Mendu R. Rao. *The Russian method for linear inequalities III: Bounded integer programming*. PhD thesis, INRIA, 1981.

- [25] Sundaram Seshu and Myril B. Reed. *Linear Graphs and Electrical Networks*. Addison-Wesley, Reading, 1961.
- [26] Paul D. Seymour. Decomposition of Regular Matroids. *Journal of combinatorial Theory, Series B*, 28(3):305–359, 1980. doi:10.1016/0095-8956(80)90075-1.
- [27] Daniel D. Sleator and Robert E. Tarjan. A data structure for dynamic trees. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 26(3):114–122, 1981. doi:10.1145/800076.802464.
- [28] Robert E. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, jun 1972. doi:10.1137/0201010.
- [29] Robert E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975. doi:10.1145/321879.321884.
- [30] Robert E. Tarjan and Jan van Leeuwen. Worst-Case Analysis of Set Union Algorithms. *Journal of the ACM (JACM)*, 31(2):245–281, 1984. doi:10.1145/62.2160.
- [31] Nobuaki Tomizawa. An  $O(m^3)$  algorithm for solving the realization problem of graphs (in Japanese). Technical report, Technical Group on Circuit and System Theory, Institute of Electronics and Communication Engineers of Japan, 1976.
- [32] Klaus Truemper. A decomposition theory for matroids. V. Testing of matrix total unimodularity. *Journal of Combinatorial Theory, Series B*, 49(2):241–281, aug 1990. doi:10.1016/0095-8956(90)90030-4.
- [33] Klaus Truemper. *Matroid decomposition*. Leibniz, 1998. URL: <https://www.emis.de/monographs/md/matroiddecompositionbook.pdf>.
- [34] William T. Tutte. An Algorithm for Determining Whether a Given Binary Matroid is Graphic. *Proceedings of the American Mathematical Society*, 11(6):905, dec 1960. doi:10.2307/2034435.
- [35] William T. Tutte. From Matrices to Graphs. *Canadian Journal of Mathematics*, 16:108–127, 1964. doi:10.4153/cjm-1964-011-0.
- [36] William T. Tutte. Lectures on Matroids. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 69B(January-June), 1965. URL: [https://nvlpubs.nist.gov/nistpubs/jres/69B/jresv69Bn1-2p1\\_A1b.pdf](https://nvlpubs.nist.gov/nistpubs/jres/69B/jresv69Bn1-2p1_A1b.pdf).
- [37] William T. Tutte. *Connectivity in Graphs*. University of Toronto Press, 1966. URL: <http://www.jstor.org/stable/10.3138/j.ctvfrxhc8>.
- [38] Hassler Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932. doi:10.2307/2371086.
- [39] Hassler Whitney. 2-isomorphic graphs. *American Journal of Mathematics*, 55(1):245–254, 1933. doi:10.2307/2371127.

## A Additional results

**Lemma 67.** *Let  $G$  be a 3-connected multigraph with spanning tree  $T$  and let  $Y \subseteq E(G) \setminus T$  be non-empty such that two adjacent vertices  $u, v \in V(G)$  are both  $Y$ -splittable. Then one side of the neighborhood split of  $u$  is given by a set of edges that connects  $u$  to  $v$ .*

*Proof.* Since  $G$  is 3-connected,  $G - \{u, v\}$  is connected. In particular, this implies that the connected components of  $G \setminus Y - \{u, v\}$  are connected to each other by edges in  $Y$ .

Let  $(I, J)$  denote the neighborhood split of  $H_Y^u$ . If  $u$  and  $v$  are both  $Y$ -stars then we have  $Y \subseteq \delta(u) \cap \delta(v)$ , and  $H_Y^u$  consists of only a single vertex. Without loss of generality, let this be in  $I$ . Then  $\delta^J(u) = Y$  holds, which together with  $Y \subseteq \delta(u) \cap \delta(v)$  shows that  $\delta^J(u)$  consists only of edges connecting  $u$  to  $v$ .

Otherwise, if not both  $u$  and  $v$  are  $Y$ -stars then Lemma 8 implies that at least one of them must be an articulation vertex of  $G \setminus Y$ . By Lemma 40,  $u$  and  $v$  are adjacent on  $Q$ , and are thus connected by a spanning tree edge  $t \in T$ . Since  $T$  is a spanning tree and  $u$  and  $v$  are connected by  $t \in T$  and  $T \cap Y = \emptyset$ , each connected component of  $G \setminus Y - \{u, v\}$  must connect to either  $u$  or  $v$  using a spanning tree edge. Let  $\mathcal{H}$  be the connected components of  $G \setminus Y - \{u, v\}$ . We partition  $\mathcal{H}$  into  $\mathcal{H}_u$  and  $\mathcal{H}_v$ , based on whether the component connects to  $u$  or  $v$ . Let  $\mathcal{H}_Y^{u,v}$  be the auxiliary graph where each vertex represents a component of  $G \setminus Y - \{u, v\}$ . Two vertices  $u, v \in \mathcal{H}_Y^{u,v}$  are connected by an edge if the components of  $G \setminus Y - \{u, v\}$  are connected by a  $Y$ -edge.

Now consider  $H_Y^v$ . Since  $G \setminus Y - \{v\}$  can be obtained by adding back  $u$  to  $G \setminus Y - \{u, v\}$ , we observe that the spanning tree edges connecting to  $u$  imply that  $\mathcal{H}_u$  is merged into one component  $h_u \in V(H_Y^v)$ . Since  $H_Y^v$  is bipartite,  $h_u$  has no self loops, and hence  $\mathcal{H}_u$  must form an independent set in  $\mathcal{H}_Y^{u,v}$ . Similarly,  $\mathcal{H}_v$  is also an independent set of  $\mathcal{H}_Y^{u,v}$ .

We argued above that  $\mathcal{H}_Y^{u,v}$  must be connected, and that  $\mathcal{H}_u$  and  $\mathcal{H}_v$  are independent sets in  $\mathcal{H}_Y^{u,v}$ . Since  $\mathcal{H}_u$  and  $\mathcal{H}_v$  partition the node set of  $\mathcal{H}_Y^{u,v}$ , this shows that  $\mathcal{H}_Y^{u,v}$  is bipartite with a unique bipartition that is given by  $\mathcal{H}_u$  and  $\mathcal{H}_v$ . In particular,  $H_Y^u$  is a star centered at  $h_u$  (the component formed by merging all  $\mathcal{H}_u$ ), and the bipartition of  $H_Y^u$  is given by  $I = \{h_u\}$  and  $J = V(H_Y^u) \setminus \{h_u\}$ .

Consider  $\delta^J(u)$ , which consists of edges  $e = \{u, w\}$  for which either  $e \in Y$  holds or there exists  $h \in J$  with  $w \in h$ . We show that for  $w \neq v$ , that existence of a such an edge leads to a contradiction. In particular, note that  $w \neq v$  implies that  $w$  must lie in one of the vertices of the components of  $\mathcal{H}_u \cup \mathcal{H}_v$ .

First, consider the case where  $w \in h$  with  $h \in I$  and  $e \in Y$ . Since  $h \in I$  holds, we have  $w \in h_u$ . Then, considering  $H_Y^v$ , existence of edge  $e$  implies that  $h_u$  has a self-loop, since there must also be a spanning tree edge connecting to  $h_u$  from  $u$ . This contradicts that  $v$  is  $Y$ -splittable.

Second, consider the case where  $w \in h$  with  $h \in J$  and  $e \notin Y$ . Then,  $w$  must lie in one of the components  $\mathcal{H}_v$ . We make a case distinction, based on the cardinality of  $\mathcal{H}_u$ .

**Case 1:**  $\mathcal{H}_u = \emptyset$ . Since  $\mathcal{H}_Y^{u,v}$  is bipartite and connected, we can conclude that  $\mathcal{H}_v$  must consist of a single component. In particular, this implies that  $u$  is a leaf node of the spanning tree  $T$ . Because we have  $T \cap Y = \emptyset$ ,  $u$  cannot be an articulation vertex of  $G \setminus Y$ , as  $T$  spans the other nodes because  $u$  is a leaf. Thus, we must have  $Y \subseteq \delta(u)$ . Since we assumed that not both  $u$  and  $v$  were articulation vertices of  $G \setminus Y$ , vertex  $v$  must be an articulation vertex of  $G \setminus Y$ . Since  $\mathcal{H}_v$  consists of a single connected component  $h$ , this implies that  $G \setminus Y - \{v\}$  can only have two components if  $h$  and  $u$  are not connected using some non- $Y$  edge. However, then the existence of  $e$  contradicts that  $v$  is an articulation vertex of  $G \setminus Y$  since it connects  $u$  to  $w$ , implying that we only have a single connected component.

**Case 2:**  $\mathcal{H}_u \neq \emptyset$ . Let  $h_2$  be the component of  $\mathcal{H}_v$  in which  $w$  lies. Since  $\mathcal{H}_Y^{u,v}$  is bipartite and connected and  $\mathcal{H}_u$  is non-empty, there must exist a member  $h' \in \mathcal{H}_u$  that is adjacent to  $h_2$  in  $\mathcal{H}_Y^{u,v}$ . Then, considering  $H_Y^v$ , we observe that  $h'$  and  $h_2$  both connect to  $u$  using an edge not in  $Y$ , and  $H_Y^v$  contains a self-loop for this component given by the  $Y$  edge connecting  $h'$  and  $h_2$ , which contradicts that  $v$  is  $Y$ -splittable.

For  $w \neq v$ , we obtained a contradiction in all cases of the case distinctions that we considered. Hence, we conclude that  $\delta^J(u)$  can only contain edges  $\{u, v\}$ , which completes our proof.  $\square$