

Structural Parameterization of Steiner Tree Packing

Niko Hastrich  

Saarland University and Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Kirill Simonov  

Department of Informatics, University of Bergen, Norway

Abstract

STEINER TREE PACKING (STP) is a notoriously hard problem in classical complexity theory, which is of practical relevance to VLSI circuit design. Previous research has approached this problem by providing heuristic or approximate algorithms. In this paper, we show the first FPT algorithms for STP parameterized by structural parameters of the input graph. In particular, we show that STP is fixed-parameter tractable by the tree-cut width as well as the fracture number of the input graph.

To achieve our results, we generalize techniques from EDGE-DISJOINT PATHS (EDP) to GENERALIZED STEINER TREE PACKING (GSTP), which generalizes both STP and EDP. First, we derive the notion of the augmented graph for GSTP analogous to EDP. We then show that GSTP is FPT by

- the tree-cut width of the augmented graph,
- the fracture number of the augmented graph,
- the slim tree-cut width of the input graph.

The latter two results were previously known for EDP; our results generalize these to GSTP and improve the running time for the parameter fracture number. On the other hand, it was open whether EDP is FPT parameterized by the tree-cut width of the augmented graph, despite extensive research on the structural complexity of the problem. We settle this question affirmatively.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Graph algorithms analysis

Keywords and phrases Steiner tree packing, structural parameters, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.51

Related Version *Full Version*: <https://arxiv.org/abs/2505.09250> [18]

Funding This work was conducted in part at Hasso Plattner Institute, University of Potsdam.

Niko Hastrich: This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 850979).

1 Introduction

In STEINER TREE PACKING (STP), we are given a triple (G, T, d) , where G is a graph, $T \subseteq V(G)$ is the *terminal set*, and $d \in \mathbb{N}^+$ is the *demand*. The goal is to decide whether there are d edge-disjoint trees F_1, F_2, \dots, F_d in G which all contain the vertices T . It is easy to see that the problem is polynomial-time solvable if $|T| \leq 2$ or $d = 1$. However, even for $|T| \geq 3$ or $d \geq 2$, STP is already NP-hard [19, 1]. Despite this obstacle, STP – and the related problems EDP and GSTP, which we introduce shortly – has extensive applications. The problem has been closely studied in the context of VLSI circuit design [22, 5, 7, 23, 25, 17], designing computer networks for multicasting [24, 6, 15], and video-conferencing [28].

From a more theoretical perspective, STEINER TREE PACKING has seen a long line of research focusing on the optimal gap between the edge connectivity of the terminal set and the size of the tree packing. On the one hand, if there are d edge-disjoint subgraphs



© Niko Hastrich and Kirill Simonov;

licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng

Article No. 51; pp. 51:1–51:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



connecting T , T is clearly d -edge-connected. On the other, Kriesell [21] conjectured that if T is $2d$ -edge-connected, then G contains d edge-disjoint trees connecting T . While the conjecture remains open, we know that $(5d + 4)$ -edge-connectivity is sufficient [8]. These results are constructive, providing algorithms that approximate the size of a maximum Steiner tree packing within a constant factor in polynomial time. However, even by proving the original conjecture of Kriesell one could only obtain a 2-approximate algorithm for STP.

The situation with exact polynomial-time algorithms for STP is considerably more restrictive. As discussed above, in terms of the size $|T|$ of the terminal set and the demand d , we can only expect polynomial time when both are bounded. In fact, it follows from the celebrated result of Robertson and Seymour [26], that STP is fixed-parameter tractable by $|T| + d$. However, this result is non-constructive and the running time as a function of $|T| + d$ is enormously fast-growing; in particular, this result has very little practical relevance.

Therefore, it is natural to turn to structural properties of the input graph, in order to identify cases where the problem can be solved efficiently. Specifically, we are interested in FPT algorithms for STP under the respective parameter. Prior to this work, no such FPT algorithms were known. In fact, for treewidth this is highly unlikely. STP has INTEGER 2-COMMODITY FLOW as a special case [1], and this problem has recently been shown to be W[1]-hard parameterized by treewidth [2]. Thus, STP is also W[1]-hard parameterized by treewidth, even on instances where the terminal set T contains just three vertices.

To identify regimes where FPT algorithms for STP are feasible, we first turn our attention to the closely related EDGE-DISJOINT PATHS (EDP) problem. An EDP instance is a tuple (G, \mathcal{T}) , where G is a graph and $\mathcal{T} \subseteq \binom{V(G)}{2}$ is a set of terminal pairs in $V(G)$. The task is to decide whether for each $\{u, v\} \in \mathcal{T}$ there is an uv -path P_{uv} in G such that all $\{P_{uv}\}_{\{u,v\} \in \mathcal{T}}$ are pairwise edge-disjoint. This problem is FPT when parameterized by $|\mathcal{T}|$ [26]. However, it is notoriously hard with respect to common structural parameters. In fact, EDP is NP-hard even on complete bipartite graphs where one side contains 3 vertices [10]. These graphs have a vertex cover of size 3, which rules out FPT algorithms for EDP parameterized by any of the following classical structural parameters unless $P = NP$: treewidth, fracture number¹, size of the smallest feedback vertex set, size of the smallest vertex cover.

Still, there are several FPT algorithms known for EDP. They fall in two categories. The first category are FPT algorithms with respect to structural parameters that are based on edge-cuts, like slim tree-cut width, rather than vertex-cuts, like treewidth [13, 12, 3, 14]. The second category of FPT algorithms are based on structural parameters, considered not with respect to the host graph G , but rather to the augmented graph $G + \mathcal{T}$, where an edge is inserted between every terminal pair [14]. Intuitively, this captures the relations between the terminal pairs directly in the graph structure. This allows for positive results. For example, EDP is NP-hard for graphs of fracture number 3, but fixed-parameter tractable by fracture number of the augmented graph [14]. However, these results for EDP carry little direct implication for STP, as there is no simple reduction known between instances of one problem to the other.

In order to provide a unified perspective on both STP and EDP, we consider the following GENERALIZED STEINER TREE PACKING (GSTP) problem, which can be seen as a version of STP with multiple terminal sets. Formally, in GSTP we are given a triple (G, \mathcal{T}, d) , where G is the underlying graph, $\mathcal{T} \subseteq 2^{V(G)}$ is the set of terminal sets, and $d: \mathcal{T} \rightarrow \mathbb{N}^+$ gives the demand for each terminal set. Our task is to decide whether there is a set of pairwise edge-disjoint, connected subgraphs \mathcal{F} of G and an assignment function $\pi: \mathcal{F} \rightarrow \mathcal{T}$ such that

¹ Fracture number is equivalent to another parameter called vertex integrity.

■ **Table 1** The parameterized complexity of STP, EDP, and GSTP with respect to structural parameters of the host graph or the augmented graph. See Section 2 for parameter definitions. The value D is the sum of demands. New results are highlighted with the symbol \star , results where we improve the running time are highlighted with \star . We abbreviate -hard with -h.

	STP		EDP		GSTP	
	Host	Aug.	Host	Aug.	Host	Aug.
tw	W[1]-h [1, 2]		paraNP-h [10]	W[1]-h [14]	paraNP-h [10]	W[1]-h [14]
tw + D	FPT \star Thm. 30		FPT \star Thm. 30		FPT \star Thm. 30	
fn	FPT \star Thm. 13		paraNP-h [10]	FPT \star Thm. 13	paraNP-h [10]	FPT \star Thm. 13
tcw	FPT \star Cor. 31		W[1]-h [13]	FPT \star Cor. 28	W[1]-h [13]	FPT \star Cor. 28
stcw	FPT \star Cor. 21		FPT [12, 3]		FPT \star Cor. 21	

every solution subgraph $F \in \mathcal{F}$ is assigned to a terminal set $\pi(F)$, which is contained in F (i.e., $\pi(F) \subseteq V(F)$). Additionally, for every terminal set $T \in \mathcal{T}$, the assignment function needs to assign $d(T)$ many solution subgraphs to T . The GSTP problem is not only a natural generalization of both STP and EDP, but was also studied directly, e.g., in the area of VLSI design [17, 22, 7]. Note that in the literature both STP and GSTP are often referred to as “Steiner tree packing”; throughout this work, we stick to the formal definitions of STP and GSTP as above, in order to avoid ambiguity.

Our contribution. In this paper, we generalize all known FPT algorithms for EDP parameterized by structural parameters to GSTP, which greatly extends the applicability of the underlying techniques. In particular, this allows us to apply them to STP. Moreover, in doing so, we discover new FPT algorithms for EDP. We positively settle the open question, whether EDP is FPT with respect to the tree-cut width of the augmented graph in the affirmative. Finally, we also improve the running time of several known FPT algorithms for EDP. See an overview of our results in comparison to previously known results in Table 1.

Overview and structure. This paper is structured as follows. We start in Section 2 by introducing relevant definitions and notation. In Section 3, we extend the notion of augmentation from EDP to GSTP. To the best of our knowledge, such a generalization was not studied before. We argue that among two natural approaches, one is strictly more general, and we settle on it for the remainder of this paper.

Building on this definition, we show in Section 4 that GSTP is FPT parameterized by the fracture number of the augmented graph. This result directly applies to STP parameterized by the fracture number of the host graph. This is in stark contrast to the fact, that EDP, and therefore GSTP, is paraNP-hard parameterized by the fracture number of the host graph [10]. The running time we obtain is doubly exponential in the parameter. This improves upon the triply exponential running time obtained by Ganian et al. [14] for EDP parameterized by the fracture number of the augmented graph.

We then focus on results using tree-cut decompositions in Section 5. First, we define an additional property for tree-cut decompositions, which we call being *simple*. We use this property to streamline our algorithms on tree-cut decompositions, which we believe

might be of independent interest. Then, we show how to decide an instance of GSTP given a simple tree-cut decomposition. For tree-cut decompositions, we are interested in two measures of the decomposition, its width and slim width, where the former is bounded by a function of the latter. We then show how the latter result can be applied to obtain an FPT algorithm parameterized by the slim tree-cut width of the host graph and the tree-cut width of the augmented graph. This directly implies that EDP is FPT by the tree-cut width of the augmented graph, which was not known before.

Augmentation of a single terminal set can increase the tree-cut width arbitrarily. So, the previous result does not directly translate to an FPT algorithm for STP. In Section 6, we examine this further. We show that we can avoid this hurdle, and provide an FPT algorithm for STP parameterized by tree-cut width of the host graph. This result combines most of the other results obtained in this paper. To achieve this, we use a win-win strategy by the size of the terminal set. If the size of the terminal set is small, then augmentation does not increase the tree-cut width much, and we apply the result obtained in Section 5. Otherwise, we show that we can assume the demand to be small. We then develop an FPT algorithm for GSTP parameterized by the sum of treewidth and demand. As treewidth dominates tree-cut width, this FPT algorithm can be used to solve the remaining case. Additionally, this FPT algorithm directly applies to EDP parameterized by the sum of treewidth and the number of terminal pairs improving the previously known running time [29].

Omitted proofs. Due to space constraints, proofs are omitted from this extended abstract. They were provided to the reviewers and can be found in the full version of this paper [18].

2 Preliminaries

We denote the natural numbers by $\mathbb{N} = \{0, 1, 2, \dots\}$ and the positive natural numbers by \mathbb{N}^+ . Let $k \in \mathbb{N}$, let $[k] := \{1, 2, \dots, k\}$ and $[k]_0 = \{0\} \cup [k]$. For any set S , we denote with 2^S its power-set. Let G be a graph. Unless explicitly stated otherwise, we only consider simple graphs. Let $S \subseteq V(G)$, we refer to the edges with exactly one endpoint in S by $\delta_G(S)$. Let $uv \in E(G)$, contracting uv merges u and v into a single vertex, while keeping vertices adjacent to u or v adjacent to the combined vertex. Similarly, contracting a vertex set $S \subseteq V(G)$ merges all the vertices in S into a single vertex, while keeping vertices adjacent to at least one vertex of S adjacent to the new vertex. Let $v \in V(G)$, suppressing v is the operation of inserting edges between all neighbors of v and removing v . The set of maximally connected subgraphs of G is denoted with $\text{comp}(G)$. Let H be a hypergraph. H is minimally connected if we can remove no hyperedge while preserving connectivity.

Now, we give the definitions of the structural parameters that we consider. For any parameter that is based on a *decomposition*, which contains a graph G' on vertex-sets of G , we refer to the vertices of G' as *nodes* or *bags* and to its edges by *links*.

Fracture Number. Let $S \subseteq V(G)$. We call S a *fracture modulator* if every component in $G - S$ contains at most $|S|$ vertices. The size of the smallest fracture modulator is called the fracture number $\text{fn}(G)$ of G . Dvorák et al. [9] claim that for any graph H , one can find a fracture modulator of minimal size in time $\mathcal{O}((\text{fn}(H) + 1)^{\text{fn}(H)} |E(H)|)$. We note that their proof suffers from a minor inaccuracy, and present a corrected version that runs in time $\mathcal{O}((2\text{fn}(H) - 1)^{\text{fn}(H)} |V(H)|)$, see Appendix A of the full version for details. Fracture number is always at most a factor of two away from another parameter called vertex integrity.

Tree-Cut Width. The parameter *tree-cut width* was introduced by Wollan [27].

► **Definition 1.** A tree-cut decomposition of a graph G is a tuple $\mathcal{D} := (T, \mathcal{X})$, where T is a rooted tree and $\{X_s \subseteq V(G) \mid s \in V(T)\} := \mathcal{X}$ a near-partition of $V(G)$.

Here, subsets in a *near-partition* may be empty, as opposed to a partition. For a node $t \in V(T)$, let $T_t^{\mathcal{D}}$ be the sub-tree rooted at t and we set $Y_t^{\mathcal{D}} := \bigcup_{s \in T_t^{\mathcal{D}}} X_s$ to be the vertices contained in the bags of $T_t^{\mathcal{D}}$. Let $\partial_t := G[\delta(Y_t^{\mathcal{D}})]$ be the graph induced by all edges crossing the link between t and its parent, which we refer to as the *boundary* at t . The *adhesion* $\text{adh}_{\mathcal{D}}(t)$ of t is defined as $|E(\partial_t)|$.

The *torso* of a tree-cut decomposition at node t , denoted as $H_t^{\mathcal{D}}$, is the graph obtained from G as follows. Consider the connected components of $T - t$ and for each $C \in \text{comp}(T - t)$, define $Z_C := \bigcup_{s \in V(C)} X_s$. The torso at t is obtained from G by contracting for all $C \in \text{comp}(T - t)$ the sets Z_C into a single vertex z_C . Note that this may create parallel edges. Consider the unique graph $\tilde{H}_t^{\mathcal{D}}$, obtained from $H_t^{\mathcal{D}}$, by repeatedly suppressing vertices not in X_t of degree at most 2 and removing loops. $\tilde{H}_t^{\mathcal{D}}$ is called the *3-center* of $H_t^{\mathcal{D}}$ with respect to X_t .

► **Definition 2.** The *width* of the tree-cut decomposition is

$$\max \left(\{\text{adh}(s)\}_{s \in V(T)} \cup \{|V(\tilde{H}_s^{\mathcal{D}})|\}_{s \in V(T)} \right).$$

The *tree-cut width* $\text{tcw}(G)$ of G is the smallest width of a tree-cut decomposition. We call t *empty* if $X_t = \emptyset$ and we denote with $\text{chil}(t)$ the set of children of t in T . We call t *thin* if $\text{adh}_{\mathcal{D}}(t) \leq 2$ and *bold* otherwise. The set of thin children is denoted by $\text{t-chil}_{\mathcal{D}}(t)$ and the set of bold children as $\text{b-chil}_{\mathcal{D}}(t)$. A tree-cut decomposition is *nice*, if for all thin nodes t the sets $N_G(Y_t)$ and $\bigcup_{s \text{ is a sibling of } t} Y_s^{\mathcal{D}}$ are disjoint. Any tree-cut decomposition can be transformed into a nice tree-cut decomposition in cubic time [11]. In all notation we leave off the tree-cut decomposition, if it is clear from context. For every graph G , we can compute a tree-cut decomposition of width at most $2\text{tcw}(G)$ in time $2^{\mathcal{O}(\text{tcw}(G)^2 \log \text{tcw}(G))} |V(G)|^2$ [20].

Slim Tree-Cut Width. The *slim tree-cut width* is defined similarly to tree-cut width. Consider a tree-cut decomposition $\mathcal{D} = (T, \mathcal{X})$ and $t \in V(T)$. Define the *2-center* $\tilde{H}_t^{2;\mathcal{D}}$ of $H_t^{\mathcal{D}}$ with respect to X_t analogous to the 3-center, while only suppressing vertices of degree at most 1.

► **Definition 3.** The *slim width* of \mathcal{D} is defined to be

$$\max \left(\{\text{adh}(t)\}_{t \in V(T)} \cup \{|V(\tilde{H}_t^{2;\mathcal{D}})|\}_{t \in V(T)} \right).$$

The *slim tree-cut width* $\text{stcw}(G)$ of G is the smallest slim width of any tree-cut decomposition of G . Ganian et al. [12] provide an algorithm that, given a graph G , computes a tree-cut decomposition of slim width at most $6(\text{stcw}(G) + 1)^3$ in time $2^{\mathcal{O}(\text{stcw}(G)^2 \log \text{stcw}(G))} |V(G)|^4$.

To give perspective on the different parameters considered in this work, we note that treewidth dominates tree-cut width (i.e., bounded tree-cut width implies bounded treewidth) and fracture number, and tree-cut width dominates slim tree-cut width, while (slim) tree-cut width is incomparable with fracture number. We remark that all these parameters model sparse graphs; formally, n -vertex graphs where the parameter is bounded have $\mathcal{O}(n)$ edges.

3 Augmentation for GSTP

The augmented graph for an instance (G, \mathcal{T}) of EDP is defined to be G with all terminal pairs connected. That is, the augmented graph $G^{\mathcal{T}}$ is exactly $G + \mathcal{T}$. To extend this notion to an instance (G, \mathcal{T}, d) of GSTP, we consider two different approaches. The *clique-augmented graph* $G^{\text{cliq}(\mathcal{T})}$ is obtained from G by adding for all $T \in \mathcal{T}$ an edge between every pair of vertices in T . If an edge, that added this way was already present, we obtain parallel edges. As there is a reduction for EDP ensuring that all terminals are unique degree one vertices [10], this definition matches the definition for EDP very closely. On the other hand, the *vertex-augmented graph* $G^{\text{vert}(\mathcal{T})}$ is obtained from G by considering each $T \in \mathcal{T}$ and adding a new vertex $\text{aug}(T)$ to the graph, which is adjacent to all vertices of T .

We compare these two definitions for the parameters treewidth, feedback vertex set number, fracture number, vertex cover number, tree-cut width, slim tree-cut width, feedback edge number, and the sum of treewidth and maximum degree. Let κ be any of these parameters except vertex cover number. Then, we can bound $\kappa(G^{\text{vert}(\mathcal{T})})$ by a function of $\kappa(G^{\text{cliq}(\mathcal{T})})$. For the sum of treewidth and maximum degree, the reverse is possible as well. For the other parameters, the reverse is not possible. For the vertex cover number, there is no function bounding in either direction. In other words, FPT algorithms considering the vertex-augmented graph are almost always more general than those considering the clique-augmented graph, with respect to the parameters considered in this paper. Therefore, we focus on the vertex augmented graph, which, from now on, we call simply the *augmented graph*, and denote this graph by $G^{\mathcal{T}}$.

4 GSTP by Augmented Fracture Number

In this section, we lay out our result that GSTP is FPT when parameterized by the fracture number of the augmented graph. Let X be a fracture modulator of $G^{\mathcal{T}}$. We achieve this result by showing an equivalence characterization of the components of $G^{\mathcal{T}} - X$, which allows us to construct an equivalent ILP instance where number of variables is bounded by a function of $|X|$. A similar result is known for EDP [14]. We generalize this significantly, while improving the running time from triply exponential to doubly exponential.

Consider any $U \subseteq V(G^{\mathcal{T}})$, we call the set of all $T \in \mathcal{T}$ with $\text{aug}(T) \in U$ by \mathcal{T}_U . Let $C \in \text{comp}(G^{\mathcal{T}} - X)$. To better assess, which terminal sets might need special attention, we only consider fracture modulators with a particular structure.

- **Definition 4.** Let $X \subseteq V(G^{\mathcal{T}})$ be a fracture modulator of $G^{\mathcal{T}}$. We call X nice, if
1. $G[X]$ is edgeless,
 2. for each terminal set $T \in \mathcal{T}_X$, there exist two distinct components $C, C' \in \text{comp}(G^{\mathcal{T}} - X)$ with $V(C) \cap T \neq \emptyset$ and $V(C') \cap T \neq \emptyset$.

Note that even though $G[X]$ is edgeless, $G^{\mathcal{T}}[X]$ may contain edges. We show that we can always turn a fracture modulator into a nice fracture modulator of similar size.

- **Lemma 5.** Let $\mathcal{P} := (G, \mathcal{T}, d)$ be an instance of GSTP and X be a fracture modulator of $G^{\mathcal{T}}$. In linear time, one can construct an equivalent instance $\mathcal{P}' = (G', \mathcal{T}, d)$ and a nice fracture modulator S of $G'^{\mathcal{T}}$ with $|S| \leq 2|X|$ and $|V(G')| \leq |V(G)| + \binom{|X|}{2} + 2|X|$.

From now on, we assume that S is a nice fracture modulator for (G, \mathcal{T}, d) . Each terminal set T has its augmented vertex $\text{aug}(T)$ either in S , meaning $T \in \mathcal{T}_S$, or in some component $C \in \text{comp}(G^{\mathcal{T}} - S)$, in which case $T \in \mathcal{T}_C$. We also denote with \mathcal{T}^* the set

of all $T \in \mathcal{T}$ satisfying $T \subseteq S$, note that $\mathcal{T}^* \cap \mathcal{T}_S = \emptyset$ by Definition 4. After removing terminal sets T with $|T| < 2$, we reject any instance for which there is a $v \in V(G)$ such that $\deg_G(v) < \sum_{T \in \mathcal{T}: v \in T} d(T)$. This implies for each $T \in \mathcal{T} \setminus \mathcal{T}^*$ that $d(T) < 2|S|$, since the degree of any vertex outside of S is less than $2|S|$.

Component Configurations. In order to fulfill the connectivity requirements, each component might need to use edges from other components, or other components might need to use edges from this component. Denote with $C^+ := G^{\mathcal{T}}[V(C) \cup S]$ the subgraph induced on the component and the fracture modulator. Based on this, we introduce the concept of a *component-configuration* characterizing how a component $C \in \text{comp}(G^{\mathcal{T}} - S)$ can interact with the remaining instance. Let $u := \binom{2|S|}{2}$, so C^+ has at most u edges, and define a configuration γ of C as a tuple $(\text{dem}_\gamma, \text{supl}_\gamma, \text{assign}_\gamma)$ with $\text{dem}_\gamma, \text{supl}_\gamma : 2^{S \cap V(G)} \rightarrow [u]_0$ and $\text{assign}_\gamma : \mathcal{T}_S \times [2|S|] \times [|S|] \rightarrow 2^{S \cap V(G)}$ (if the configuration is clear from context, we omit the index γ).

Intuitively, the first part (i.e., dem) signifies how often each subset of the fracture modulator gets connected by other components and used for the connection requirements of \mathcal{T}_C – what is the additional demand for each subset? The second component (i.e., supl) signifies how often each subset of the fracture modulator gets connected inside this component, but these connections are *not* used to satisfy connection requirements of terminal sets in \mathcal{T}_C – what is the additional supply for each subset?

For the final part, consider a terminal set $T \in \mathcal{T}_S$. We have $T \notin \mathcal{T}^*$; so, $d(T) \leq 2|S|$. This allows us to explicitly store the contribution of C to every tree in the solution assigned to such a terminal set. This is necessary since T can span an arbitrary number of components in $G^{\mathcal{T}} - S$. For each $i \in [d(T)]$, the information required for the i -th tree assigned to T is stored in $\text{assign}(T, i, \cdot)$. Call this tree F^i . Let $\sigma : [|S|] \rightarrow V(C)$ be a surjection. Then, we can imagine for all $j \in [|S|]$ that $\text{assign}(T, i, j)$ is a set of vertices in S which $\sigma(j)$ can reach via $F^i[V(C^+)]$. Note that it might not be all such vertices.

Admitted Configurations. Not every component can be in any configuration in a valid solution. For example, a component with k edges can not supply more than k connections to other components. To capture this concept, we say a component *admits* a configuration if it can locally satisfy all the requirements of the configuration and of the terminal sets in \mathcal{T}_C . In order to make this rigorous, we construct an auxiliary instance of GSTP to help us define when C admits a configuration. For this, let $\sigma : [|S|] \rightarrow V(C)$ be a surjection, which we use to encode the assign mapping as explained above. The instance we construct is denoted by $\text{confInst}(C, \gamma, \sigma)$.

First, we define the host-graph of the instance $\text{confInst}(C, \gamma, \sigma)$. For this, we start with the graph C^+ and for each $Q \subseteq S \cap V(G)$, we add $\text{dem}(Q)$ -many vertices v to the graph with $N(v) = Q$. Denote this graph with H .

Now, we define the terminal sets, that need to be connected. Denote for all $T \in \mathcal{T}_S$, $i \in [d(T)]$ and $U \subseteq S \cap V(G)$ the set $A(T, i, U, \sigma) := \{\sigma(j) \mid j \in [|S|]; \text{assign}(T, i, j) = U\}$ the set of vertices in C that is assigned to U to satisfy connections for the i -th tree of T . Now let $\text{assignSets}(T, i, \sigma) := \bigcup_{\emptyset \neq U \subseteq S \cap V(G): A(T, i, U, \sigma) \neq \emptyset} \{U \cup A(T, i, U, \sigma)\}$ be the subsets of $S \cap V(G)$ that get assigned some vertex unioned with the assigned vertices. Let

$$\begin{aligned} \mathcal{Q} &:= \{T \in \mathcal{T}_C \mid T \cap C \neq \emptyset\}, \\ \mathcal{S} &:= \{X \subseteq S \cap V(G) \mid \text{supl}(X) > 0\}, \\ \mathcal{A} &:= \bigcup_{T \in \mathcal{T}_S, i \in [d(T)]} \text{assignSets}(T, i, \sigma). \end{aligned}$$

Note that if C consists of exactly one augmented vertex, \mathcal{Q} is empty; otherwise $\mathcal{Q} = \mathcal{T}_C$. Further, the set \mathcal{A} might intersect \mathcal{Q} , but one can verify that \mathcal{S} is disjoint from \mathcal{A} and \mathcal{Q} . The complete set of terminal sets, for which we need connections, is $\mathcal{U} := \mathcal{Q} \cup \mathcal{S} \cup \mathcal{A}$.

Finally, we specify the number of required connections $d' : \mathcal{U} \rightarrow \mathbb{N}^+$. For this extend d , supl and assign to yield 0 or \emptyset on arguments not in their original domain. For all $U \in \mathcal{U}$ let

$$d'(U) := d(U) + \text{supl}(U) + \sum_{T \in \mathcal{T}_S} |\{i \in [d(T)] \mid U \in \text{assignSets}(T, i, \sigma)\}|,$$

and define $\text{confInst}(C, \gamma, \sigma) = (H, \mathcal{U}, d')$.

► **Definition 6.** We say a component $C \in \text{comp}(G^T - S)$ admits a configuration γ , if there is a surjection $\sigma : [|S|] \rightarrow V(C)$ such that

1. for all $T \in \mathcal{T}_S$, $i \in [d(T)]$, and $j \in \sigma^{-1}(T \cap V(C))$, we have $\text{assign}(T, i, j) \neq \emptyset$,
2. there is a solution (\mathcal{F}, π) to $\text{confInst}(C, \gamma, \sigma)$ such that
 - a. for all $F \in \pi^{-1}(\mathcal{S})$, we have that $V(F) \subseteq V(C^+)$,
 - b. for all $v \in V(H) \setminus V(C^+)$ where H is the host-graph of $\text{confInst}(C, \gamma, \sigma)$, there is exactly one $F \in \mathcal{F}$ with $v \in V(F)$ and for this F , we have $\deg_F(v) \geq 2$,
 - c. for all $F \in \mathcal{F}$, we have $E(C^+) \cap E(F) \neq \emptyset$ and F is cycle-free.

We say that $(\sigma, \mathcal{F}, \pi)$ gives rise to γ on C . With Item 1 we ensure for all $T \in \mathcal{T}_S$ that every $v \in T \cap V(C)$ is connected to a vertex in $S \cap V(G)$. Item 2a ensures that the supply claimed by this configuration is satisfied completely inside C^+ while Item 2b ensures that connections required from the outside are only used for a single solution tree. Item 2c forbids redundant configurations ensuring that the number of admitted configurations stays singly exponential in $|S|$.

We call the set of configurations a component admits its *signature* $\text{sig}(C)$. We now characterize, whether an instance is solvable solely based on the signatures of its components in $G^T - S$. For this, let Γ be a function that assigns each component of $C \in \text{comp}(G^T - S)$ a configuration in $\text{sig}(C)$. We call Γ a *configuration selector*.

► **Definition 7.** We call Γ valid, if there is a function $\rho : 2^{S \cap V(G)} \times \mathbb{N} \rightarrow 2^{2^{S \cap V(G)}}$ such that

1. for all $U \subseteq S \cap V(G)$, with $r_U := |\{(T, i) \in \mathcal{T}^* \times \mathbb{N} \mid U \in \rho(T, i)\}|$ we have

$$r_U + \sum_{C \in \text{comp}(G^T - S)} \text{dem}_{\Gamma(C)}(U) \leq \sum_{C \in \text{comp}(G^T - S)} \text{supl}_{\Gamma(C)}(U),$$

2. for all $T \in \mathcal{T}^*$ and $i \in [d(T)]$, the hypergraph $(T \cup \bigcup \rho(T, i), \rho(T, i))$ is minimally connected,
3. for all $T \in \mathcal{T}_S$ and $i \in [d(T)]$, denote with

$$\mathcal{H} := \{\text{assign}_{\Gamma(C)}(T, i, j) \mid C \in \text{comp}(G^T - S), j \in [|S|]\}$$

all subsets of S that are connected for this terminal set. The hypergraph $((T \cap S) \cup \bigcup \mathcal{H}, \mathcal{H})$ is connected.

In the definition above, the function ρ is used to capture how the requirements of the terminal sets \mathcal{T}^* are fulfilled. For a $T \in \mathcal{T}^*$ and $i \in [d(T)]$, $\rho(T, i)$ gives all connections that are needed for the i -th tree assigned to T and Item 2 ensures that these connections actually connect T . With Item 1 we ensure that there is enough supply to meet the demand. Finally, Item 3 ensures that the stored solutions for the terminal sets \mathcal{T}_S are actually connected.

We require minimal connectivity in Item 2 to limit the number of needed variables in our ILP. Note that we cannot require this in Item 3 as this would prevent for any $T \in \mathcal{T}_S$, $i \in d(T)$, and $j \in [|S|]$ that $|\text{assign}(T, i, j)| = 1$, which is required in some instances. We show, that a valid configuration selector for \mathcal{P} exists if and only if \mathcal{P} is a positive instance.

► **Lemma 8.** *Let S be a nice fracture modulator of G^T . Assume that for all $v \in V$ we have $\sum_{T \in \mathcal{T}: v \in T} d(T) \leq \deg_G(v)$. Then, there is a valid configuration selector with respect to S if and only if \mathcal{P} is positive.*

Equivalent Configurations. When building our ILP, we want to treat components with the same signature equally. We call these components equivalent. For each equivalence class, we want to represent all its components by a common set of variables. Now, consider two components $C_1, C_2 \in \text{comp}(G^T - S)$ such that there exists a graph isomorphism between C_1^+ and C_2^+ preserving S , whether a vertex is an augmented vertex, and the demand of such vertices. Then, C_1 and C_2 are equivalent. Based on this sufficient condition for equivalence, we can calculate that there are no more than $2^{\mathcal{O}(|S|^2)}$ non-empty equivalence classes.

► **Corollary 9.** *On $\text{comp}(G^T - S)$ there are at most $2^{\mathcal{O}(|S|^2)}$ non-empty equivalence classes.*

Additionally, we want to use a variable for each configuration in our ILP. So, we need to bound the number of component configurations that we actually need to consider.

► **Definition 10.** *Let γ be a component-configuration. We call γ viable, if*

$$\sum_{U \subseteq S \cap V(G)} \text{dem}(U) \leq u|S| \text{ and } \sum_{U \subseteq S \cap V(G)} \text{supl}(U) \leq u.$$

The set of all viable configurations is denoted by \mathcal{V} .

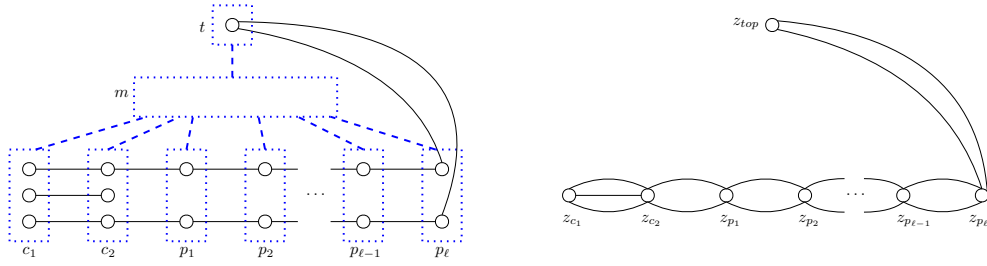
We can prove that any configuration, that is admitted by a component is indeed viable.

► **Lemma 11.** *For all $C \in \text{comp}(G^T - S)$, we have $\text{sig}(C) \subseteq \mathcal{V}$.*

By counting the number of viable configurations, we get a bound on the number of variables we will need to encode each configuration. Allowing us to compute the signature of a component by brute-force testing whether this component admits a configuration.

► **Lemma 12.** *For all $C \in \text{comp}(G^T - S)$, we have $|\text{sig}(C)| \leq |\mathcal{V}| \leq 2^{\mathcal{O}(|S|^4)}$ and we can compute $\text{sig}(C)$ in running time $2^{\mathcal{O}(|S|^4 \log |S|)}$.*

ILP Construction. To construct our ILP, denote with \mathfrak{C} the set of non-empty equivalence classes. For all $\mathcal{X} \in \mathfrak{C}$, denote the signature of any component in \mathcal{X} with $\text{sig}(\mathcal{X})$. For all $\gamma \in \text{sig}(\mathcal{X})$, we create a variable $\mathbf{d}_{\mathcal{X}, \gamma}$ signifying how many components in \mathcal{X} use the configuration γ . We ensure for all $\mathcal{X} \in \mathfrak{C}$ that $\sum_{\gamma \in \text{sig}(\mathcal{X})} \mathbf{d}_{\mathcal{X}, \gamma} = |\mathcal{X}|$. According to Corollary 9 and Lemma 12, This uses $2^{\mathcal{O}(|S|^4)}$ variables. Denote with Γ a configuration selector that is encoded by these values $\mathbf{d}_{\mathcal{X}, \gamma}$. To check Items 1 and 2 of Definition 7, we create for each $U \subseteq S \cap V(G)$ a variable \mathbf{s}_U representing the final value of r_U , amounting to at most $2^{|S|}$ many additional variables. First, we ensure that \mathbf{s}_U is at most the difference between supply and demand for U . Second, let $T \in \mathcal{T}^*$, $T \subseteq W \subseteq S \cap V(G)$, and \mathcal{M}_W be the set of minimally-connected hypergraphs on the vertex set W . We use for each $H \in \mathcal{M}_W$ a variable $\mathbf{p}_{T, H}$ that signifies how often the hypergraph H is used to fulfill the demand of T . Using this, we can check that each hyper edge $R \subseteq S \cap V(G)$ is used at most \mathbf{s}_R times overall while the demand of T is being fulfilled. We see that $|\mathcal{M}_W| \leq 2^{\mathcal{O}(|W|^2)}$; meaning, we only introduce $2^{\mathcal{O}(|S|^2)}$ additional variables.



(a) A graph family where the bags and links of the tree-cut decomposition are indicated in blue. (b) The torso at m . The 3-center is the graph induced by z_{c_1} and z_{c_2} .

■ **Figure 1** A family of graphs with tree-cut width at most 5. In the depicted nice tree-cut decomposition the node m has $\ell + 2$ bold children, where ℓ can be chosen freely.

To check Item 3 of Definition 7, assume that for all $T \in \mathcal{T}_S$, $i \in [d(T)]$, as well as $U \subseteq S \cap V(G)$, the variable $\mathbf{a}_{T,i,U}$ corresponds to the binary indicator whether there is a $C \in \text{comp}(G^T - S)$, and $j \in [|S|]$ with $U = \text{assign}_{\Gamma(C)}(T, i, j)$. If $T \cap S \subseteq U \subseteq S \cap V(G)$, we create the binary variable $\mathbf{b}_{T,i,U}$. The variable $\mathbf{b}_{T,i,U}$ is an indicator for whether U is the vertex set of the hypergraph H in Item 3 of Definition 7. So, we ensure that there is at most one $T \cap S \subseteq Z \subseteq S \cap V(G)$ with $\mathbf{b}_{T,i,Z} = 1$ and that in any valid assignment $Z = \bigcup_{C \in \text{comp}(G-S), j \in [|S|]} \text{assign}(T, i, j)$. We ensure that H is connected using a cut based approach.

Overall, we need $2^{\mathcal{O}(|S|)}$ variables of type $\mathbf{a}_{T,i,U}$ and $\mathbf{b}_{T,i,U}$. This shows, that we can represent the instance as an ILP with $2^{\mathcal{O}(|S|^4)}$ variables. As we can find a fracture modulator of minimal size in FPT time, GSTP is FPT by the fracture number of the augmented graph. We get thus the main result of this section, stated next.

► **Theorem 13.** *Let $\mathcal{P} = (G, \mathcal{T}, d)$ be an instance of GSTP. We can decide whether \mathcal{P} is a positive instance in running time $2^{2^{\mathcal{O}(\text{tn}(G^T)^4)}} |G| + \mathcal{O}(|\mathcal{P}|)$.*

5 GSTP by Augmented/Slim Tree-Cut Width

In this section, we present an outline of our proof that GSTP is FPT by the tree-cut width of the augmented graph as well as the slim tree-cut width of the host graph. Both boil down to solving a dynamic program for a tree-cut decomposition that fulfills some additional assumptions. Note that GSTP is W[1]-hard by tree-cut width of the host graph, so these additional assumptions are necessary to achieve an FPT algorithm.

Our dynamic program relies on the fact that the number of bold children of any node in our tree-cut decomposition is bounded by a function of its width. Ganian et al. [11] claimed that in a nice tree-cut decomposition the number of bold children of any node is bounded by $w + 1$. In Figure 1, we provide a counter example to this. This shows that in a nice tree-cut decomposition, the number of bold children is actually not bounded by a function of its width. Still, any tree-cut decomposition can be modified in polynomial time to achieve a similar result. The main insight behind this result is that the bold nodes that are not present in the 3-center form a path like the one seen in Figure 1b.

► **Corollary 14.** *Any tree-cut decomposition (S, \mathcal{X}) can be altered in polynomial time to ensure that it becomes nice and for all $s \in S$, we have $|\text{b-chil}(s)| + |X_s| \leq w + 2$. We call such a tree-cut decomposition friendly.*

Based on this, we give the definition of a *simple* tree-cut decomposition.

► **Definition 15.** Consider a tree-cut decomposition $\mathcal{D} := (S, \mathcal{X})$ of G and let $s \in V(S)$. Denote with $\text{cross}(s) := \{T \in \mathcal{T} \mid T \cap Y_s \neq \emptyset \wedge T \setminus Y_s \neq \emptyset\}$ the set of terminal sets crossing the link between s and its parent. We call s *simple* if it is thin, $|Y_s| = 1$, $\text{adh}(s) = 2$, and $\text{cross}(s) = \emptyset$. We call \mathcal{D} *simple*, if it is friendly and all thin nodes are simple.

5.1 GSTP by a Simple Tree-Cut Decomposition

Assume we are given a simple tree-cut decomposition $\mathcal{D} := (S, \mathcal{X})$ for G of width w . For all $s \in V(S)$, the thin children of s mostly act like subdivided edges inside $G[X_s]$. To limit the number of trees that we need to consider crossing ∂_s , define $\text{d-cross}(s) := \sum_{T \in \text{cross}(s)} d(T)$.

► **Reduction Rule 16.** If there is a node $s \in V(S)$ with $\text{d-cross}(s) > \text{adh}(s)$, output a trivial negative instance.

If we apply Reduction Rule 16, we can now choose for each $s \in V(S)$ a function $\eta_s: [\text{d-cross}(s)] \rightarrow \text{cross}(s)$ such that for all $T \in \text{cross}(s)$ we have $|\eta_s^{-1}(T)| = d(T)$. This function gives us the ability to identify the different trees crossing the link between this node and its parent by a number in the set $[\text{d-cross}(s)] \subseteq [w]$. Based on this, we obtain a dynamic program for deciding the instance in FPT time with respect to w .

The Data-Table. The data-table $D(s)$ at s , is a set of tuples τ , each consisting of three parts: pastPart_τ , pastAssign_τ , and futPart_τ with $\text{pastAssign}_\tau: \text{pastPart}_\tau \rightarrow [w]$. Each of pastPart_τ and futPart_τ are partitions of a (not necessarily proper) subset of $E(\partial_s)$. If $\bigcup \text{pastPart}_\tau$ and $\bigcup \text{futPart}_\tau$ are disjoint, we call τ *syntactically valid*. Note that we do not assign the partitions in futPart_τ to indices, since at this point we do not care, whether they are eventually used for the same terminal set.

Intuitively, a tuple τ gives almost complete information about the part of the final solution that crosses ∂_s . Consider a solution (\mathcal{F}, π) to the whole instance and denote with $\mathcal{F}' \subseteq \mathcal{F}$ the trees containing an edge of $E(\partial_s)$. Set $\mathcal{F}^\downarrow := \{F \in \mathcal{F}' \mid \pi(F) \cap Y_s \neq \emptyset\}$ and $\mathcal{F}^\uparrow := \mathcal{F}' \setminus \mathcal{F}^\downarrow$ as the subgraphs crossing the link between s and its parent, which are assigned to terminal set starting below this link and starting above this link, respectively. Let $G_s := G[Y_s] \cup \partial_s$ be the graph edge-induced by all edges with at least one endpoint in Y_s . This solution corresponds to a tuple $\tau \in D(s)$ with $\text{pastPart}_\tau := \{\{E(K) \cap E(\partial_s) \mid K \in \text{comp}(F[E(G_s)])\}\}_{F \in \mathcal{F}^\downarrow}$ and $\text{futPart}_\tau := \{\{E(K) \cap E(\partial_s) \mid K \in \text{comp}(F[E(G_s)])\}\}_{F \in \mathcal{F}^\uparrow}$ where we consider the edges in $E(\partial_s)$ per connected component of the trees in \mathcal{F}' restricted to the edges incident to Y_s . For each $F \in \mathcal{F}^\downarrow$, we choose a distinct $\lambda_F \in [w]$ such that if $\pi(F) \in \text{cross}(s)$, we have $\lambda_F \in [\text{d-cross}(s)]$ and $\eta_s(\lambda_F) = \pi(F)$ and if $\pi(F) \notin \text{cross}(s)$, we have $\lambda_F \in [w] \setminus [\text{d-cross}(s)]$. Now, we set pastAssign_τ to λ_F for each set induced by F .

Consider a syntactically valid tuple τ . To formally define what $\tau \in D(s)$ is supposed to mean, let $\mathcal{U}_s := \{T \in \mathcal{T} \mid T \subseteq Y_s\}$. Note that \mathcal{U}_s and $\text{cross}(s)$ are disjoint and their union is the set of all terminals $T \in \mathcal{T}$ that are not disjoint from Y_s . We add w vertices $\{q_{s,i}\}_{i \in [w]}$ to G_s such that each has the neighborhood $V(\partial_s) \setminus Y_s$. Call this graph G_s^* . These additional vertices can be used to simulate that a subgraph gets connected outside G_s . Let $i \in [\text{d-cross}(s)]$ and denote with $Q_{s,i} := (\eta_s(i) \cap Y_s) \cup \{q_{s,i}\}$ the vertices in Y_s of the terminal set assigned to the i -th subgraph crossing ∂_s combined with $q_{s,i}$. Additionally, define for each $P \in \text{futPart}_\tau$ the set $R_{s,P} := V(G[P]) \setminus Y_s$ to be all vertices of edges contained in P that are not in Y_s . Finally, define the instance $\mathcal{D}_{s,\tau} := (G_s^*, \mathcal{U}_s \cup \{Q_{s,i}\}_{i \in [\text{d-cross}(s)]} \cup \{R_{s,P}\}_{P \in \text{futPart}_\tau}, d')$, where for all $T \in \mathcal{U}_s$ we have $d'(T) = d(T)$, for all $i \in [\text{d-cross}(s)]$, we have $d'(Q_{s,i}) = 1$, and for all $P \in \text{futPart}_\tau$, we have $d'(R_{s,P}) = |\{P' \in \text{futPart}_\tau \mid R_{s,P'} = R_{s,P}\}|$.

► **Definition 17.** For each $s \in V(S)$ the data-table $D(s)$ is the set of syntactically valid tuples τ at s where the instance $\mathcal{D}_{s,\tau}$ has a solution (\mathcal{F}, π) such that

1. we have $E(\partial_s) \cap \bigcup E(\mathcal{F}) \subseteq \bigcup \text{pastPart}_\tau \cup \bigcup \text{futPart}_\tau$,
2. for all $P \in \text{futPart}_\tau$ and $F \in \pi^{-1}(R_{s,P})$, the set $V(F)$ is disjoint from $\{q_{s,i}\}_{i \in [w]}$,
3. for all $P \in \text{futPart}_\tau$, there is a $F \in \pi^{-1}(R_{s,P})$ with $E(F) \cap E(\partial_s) = P$,
4. for all $i \in [w]$, let $\mathcal{P}_i := \text{pastAssign}_\tau^{-1}(i)$, then
 - if $\mathcal{P}_i = \emptyset$, we have $q_{s,i} \notin \bigcup V(\mathcal{F})$,
 - otherwise, there is exactly one $F \in \mathcal{F}$ with $q_{s,i} \in V(F)$ and this F additionally satisfies that $E(F - q_{s,i})$ can be partitioned into $\{E_P\}_{P \in \mathcal{P}_i}$ such that for all $P \in \mathcal{P}_i$, the graph $F[E_P]$ is connected and $E_P \cap E(\partial_s) = P$.

With Item 1 we ensure that the edges used in the solution are accounted for in pastPart_τ and futPart_τ . We want to be able to assume that for every $P \in \text{futPart}_\tau$ there is a subgraph completely contained in G_s connecting all edges of P . So, in Item 2 we ensure that the vertices $\{q_{s,i}\}_{i \in [w]}$ – that are used to simulate that a subgraph gets connected outside of this graph – are not included in the subgraphs connecting the edges in P . These subgraphs should also use exactly the edge-set P in $E(\partial_s)$, which we ensure with Item 3. Finally, consider Item 4 and $i \in [w]$. If $\mathcal{P}_i = \emptyset$, that is, no edges are assigned to the i -th subgraph, the vertex $q_{s,i}$, which is used to mark the i -th subgraph crossing ∂_s , is not used in any subgraph. Otherwise, we again ensure that for each $P \in \mathcal{P}_i$ there is a subgraph in this solution that connects the edges of P inside G_s . Notice that we can combine Items 1, 3, and 4 to show, that for any $F \in \mathcal{F}$ with $E(F) \cap E(\partial_s) \neq \emptyset$ there is a $P \in \text{pastPart}_\tau \cup \text{futPart}_\tau$ with $P \subseteq E(F)$.

Immediately, we observe that this dynamic program can indeed be used to determine whether \mathcal{P} is a positive instance, by checking for the root r of S whether $D(r) \neq \emptyset$.

Computing the Data-Table. To compute this dynamic program, we assume that for all bold children $b \in \text{b-chil}(s)$ of s , we have already computed the data-table $D(b)$. We now outline how to compute $D(s)$ in FPT-time. As the number of syntactically valid tuples is in $2^{\mathcal{O}(w \log w)}$, it is enough to decide in FPT-time for a given syntactically valid tuple τ whether $\tau \in D(s)$. For this we iterate over all simultaneous choices of $\tau_b \in D(b)$ for $b \in \text{b-chil}(s)$ and check whether the solutions witnessing $\tau_b \in D(b)$ can be extended to solutions witnessing $\tau \in D(s)$.

Assume for all $b \in \text{b-chil}(s)$ a $\tau_b \in D(b)$ is fixed. To combine the subgraphs of the sub-solutions witnessing $\tau_b \in D(b)$ to a solution witnessing $\tau \in D(s)$, we need to translate the local numberings of the solution subgraphs into a numbering shared across all solution subgraphs not fully contained in one connected component of $S - s$. When dealing with a shared mapping, we need to avoid that we map solution subgraphs assigned to different terminal sets to the same index. Let $A := \{s\} \cup \text{b-chil}(s)$ and denote with $\mathcal{T}_s^* := \{T \in \mathcal{T} \mid T \subseteq X_s\}$ the set of terminal sets completely contained in X_s , and with $\mathcal{X} := \mathcal{T}_s^* \cup \bigcup_{a \in A} \text{cross}(a)$ the set of all terminal sets, which are not completely contained in one sub-tree of $S - s$. Note that this claim holds for \mathcal{T}_s^* as none of their terminal vertices occur in any sub-tree of $S - s$. As \mathcal{D} is simple, s has at most $w + 2$ bold children. So, $w(w + 3)$ is an upper bound on the total demand of all terminal sets crossing the links between s and its parent or any of its bold children. It is also an upper bound on the number of edges crossing bold links adjacent to s . So, we need at most $w(w + 3)$ shared indices for the crossing subgraphs. Based on these observations, we can carefully construct a mapping from the local numberings to a shared numbering. Given such a mapping, we can construct $2^{\mathcal{O}(w^3)}$ GSTP instances with augmented fracture number at most $\mathcal{O}(w^2)$ to decide whether this mapping gives a viable extension.

► **Theorem 18.** *Given a simple tree-cut decomposition of width w , we can decide whether the instance is positive in time $2^{2^{\mathcal{O}(w^8)}} |V(G)|$.*

However, we can not assume, in general, that \mathcal{D} is simple. If \mathcal{D} is already close to simple everywhere, we can make \mathcal{D} simple without increasing its width too much. Let $s \in V(S)$ and denote with $N_s \subseteq \text{t-chil}(s)$ the set of thin and non-simple children of s . If for all s both $|N_s|$ and $|\text{b-chil}(s)| + |X_s| - w$ are small, we obtain a simple tree-cut decomposition of only slightly increased width. The main idea is to treat non-simple thin children like bold children.

► **Lemma 19.** *Let $\Delta_s := |N_s| + |\text{b-chil}_{\mathcal{D}}(s)| + |X_s| - w - 1$. We can compute in linear time an equivalent instance (G', \mathcal{T}, d) and a simple tree-cut decomposition \mathcal{C} of G' with width*

$$w + 4 + \max(0, \max_{s \in V(S)} \Delta_s).$$

5.2 GSTP by Slim Tree-Cut Width

We now show that GSTP is FPT by the slim tree-cut width of G . This is a direct application of Lemma 19. For this, denote with $\bar{w} \geq w$ the slim width of \mathcal{D} . We mostly need to take care of thin links with adhesion one.

► **Reduction Rule 20.** *Assume that Reduction Rule 16 was applied. Let $s \in V(S)$ with $\text{adh}(s) = 1$ and consider $\{uv\} := \delta(Y_s)$ with $u \in Y_s$ and $v \notin Y_s$. Remove uv from G and if there is a $T \in \text{cross}(s)$, increase the demand of $(T \cap Y_s) \cup \{u\}$ and $(T \setminus Y_s) \cup \{v\}$ by 1 while removing T from \mathcal{T} (if necessary, add $(T \cap Y_s) \cup \{u\}$ and $(T \setminus Y_s) \cup \{v\}$ to \mathcal{T}).*

After applying Reduction Rules 16 and 20 exhaustively and considering connected components separately, all links have adhesion at least two. This can be used to prove that for all $s \in V(S)$, we have $|\text{chil}(s)| + |X_s| \leq \bar{w}$. As $N_s \subseteq \text{t-chil}(s)$, we have $\Delta_s \leq |\text{chil}(s)| + |X_s| - w - 1 \leq \bar{w} - w - 1$. This allows us to apply Lemma 19 and obtain a simple tree-cut decomposition of width at most $\bar{w} + 4$. There is an algorithm that computes a tree-cut decomposition of slim width at most $6(\text{stcw}(G) + 1)^3$ in time $2^{\mathcal{O}(\text{stcw}(G)^2 \log \text{stcw}(G))} |V(G)|^4$ [12]. Combined with Theorem 18, we obtain that GSTP is FPT by the slim tree-cut width of the host graph.

► **Corollary 21.** *Let $\mathcal{P} := (G, \mathcal{T}, d)$ be an instance of GSTP. We can decide whether \mathcal{P} is positive in time $2^{2^{\mathcal{O}(\text{stcw}(G)^{24})}} \text{poly}(|\mathcal{P}|)$.*

5.3 GSTP by Tree-Cut Width of the Augmented Graph

Solving GSTP parameterized by the tree-cut width of the augmented graph also boils down to solving instances of GSTP with a simple tree-cut decomposition. From now on, assume that $\mathcal{D} = (S, \mathcal{X})$ is a tree-cut decomposition of $G^{\mathcal{T}}$ with width w . Set $X'_s := X_s \cap V(G)$ and let $\mathcal{X}' := \{X'_s\}_{s \in V(S)}$ for all $s \in V(S)$. Then, $\mathcal{D}' := (S, \mathcal{X}')$ is a tree-cut decomposition of G with width at most w . When we refer to Reduction Rules 16 and 20, we mean that they are applied with respect to \mathcal{D}' while keeping \mathcal{D} in sync. These reduction rules already bring us quite close to \mathcal{D}' being simple with respect to the nodes that are thin in \mathcal{D} .

► **Lemma 22.** *After exhaustively applying Reduction Rules 16 and 20 with respect to \mathcal{D}' , removing nodes that are empty in \mathcal{D} and \mathcal{D}' and splitting the instance by connected components, we have for all $s \in V(S) \setminus \{r\}$ that are thin in \mathcal{D} that $\text{adh}_{\mathcal{D}'}(s) = 2$ and $\text{cross}_{\mathcal{D}'}(s) = \emptyset$. In particular, $\delta_{G^{\mathcal{T}}}(Y_s^{\mathcal{D}'})$ does not contain an augmented and a non-augmented edge. Additionally, the tree-cut decompositions can be maintained efficiently while not increasing the width of \mathcal{D} and keeping \mathcal{D} friendly if it was friendly before.*

There are two obstacles remaining before we can show that GSTP is FPT by the tree-cut width of the augmented graph. First, we need to ensure for all thin nodes $s \in V(S) \setminus \{r\}$ that $|Y_s^{\mathcal{D}'}| \leq 1$. We call nodes $s \in V(S) \setminus \{r\}$ that are thin in \mathcal{D} , but have $|Y_s^{\mathcal{D}'}| \geq 2$ *cluttered*. Second, we need to take care of all nodes that are bold in \mathcal{D} , but thin in \mathcal{D}' . As \mathcal{D} is friendly, for each $s \in V(S)$ the number of such children is bounded by the number of bold children. So, this task can be taken care of rather quickly by applying Lemma 19.

► **Lemma 23.** *Assume \mathcal{D} is friendly, has no cluttered nodes, that Reduction Rule 20 was applied exhaustively, and that the instance is split by connected components. We can compute in linear time an equivalent instance (G', \mathcal{T}, d) and a simple tree-cut decomposition \mathcal{C} of G' of width at most $w + 5$.*

To tackle the cluttered nodes, we solve sub-instances of GSTP. The reduction rules we present now, are no reduction rules in the classical sense (i.e., they do not run in polynomial time), but rather recursion rules. We later show, how to apply these rules in a way, that we only solve simple sub-instances and mostly preserve the running time obtained in Theorem 18.

The crux of why this problem is FPT by the tree-cut width of the augmented graph, but W[1]-hard by the tree-cut width of the host-graph [13] lies in the fact, that for a cluttered node $s \in V(S)$, we have for all $T \in \mathcal{T}$ that either $T \cap Y_s^{\mathcal{D}'}$ or $T \setminus Y_s^{\mathcal{D}'}$ is empty. This means that no terminal set crosses $Y_s^{\mathcal{D}'}$. Therefore, we can mostly disregard how the instance looks on $V(G) \setminus Y_s^{\mathcal{D}'}$ for deciding how terminal sets contained in $Y_s^{\mathcal{D}'}$ are solved in a solution of the whole instance. Let $u, x \in Y_s$ and $v, y \in V(G) \setminus Y_s$ be such, that $\{uv, xy\} = \delta_{G\mathcal{T}}(Y_s^{\mathcal{D}'})$, and let $\mathcal{U} := \{T \in \mathcal{T} \mid T \subseteq Y_s\}$ be the terminal sets contained in $Y_s^{\mathcal{D}'}$. Note that $uv, xy \in E(G)$. First, we consider the case, where we can satisfy the requirements of \mathcal{U} , while supplying an additional connection for vy to $V(G) \setminus Y_s^{\mathcal{D}'}$, while solving the requirements of \mathcal{U} .

► **Reduction Rule 24.** *Consider the instance $\mathcal{X}_s := (G[Y_s^{\mathcal{D}'}], \mathcal{U} \cup \{u, x\}, d')$ where d' is $d|_{\mathcal{U} \cup \{u, x\}}$ increased by one for the argument $\{u, x\}$. If \mathcal{X}_s is positive, remove all \mathcal{U} from \mathcal{T} and contract $Y_s^{\mathcal{D}'}$ in the original instance \mathcal{P} .*

If Reduction Rule 24 is not applicable, we know that we cannot use uv and xy in a tree for terminals contained in $V(G) \setminus Y_s^{\mathcal{D}'}$. So, we check, whether the terminal sets \mathcal{U} can be solved only using edges of $G[Y_s^{\mathcal{D}'}]$.

► **Reduction Rule 25.** *Assume that Reduction Rule 24 is not applicable to s . Consider the instance $\mathcal{Y}_s := (G[Y_s^{\mathcal{D}'}], \mathcal{U}, d|_{\mathcal{U}})$. If \mathcal{Y}_s is positive, remove \mathcal{U} from \mathcal{T} and $Y_s^{\mathcal{D}'}$ from G in the original instance \mathcal{P} .*

Finally, we need to take care of the case, where the terminal sets \mathcal{U} cannot be solved using only edges of $G[Y_s^{\mathcal{D}'}]$.

► **Reduction Rule 26.** *Assume both Reduction Rules 24 and 25 are not applicable to s . Consider the instance $\mathcal{Z}_s := (G/(V(G) \setminus Y_s^{\mathcal{D}'}), \mathcal{U}, d|_{\mathcal{U}})$. If \mathcal{Z}_s is positive, we remove \mathcal{U} from \mathcal{T} , $Y_s^{\mathcal{D}'}$ from G , and add 1 demand to the terminal set $\{v, y\}$ in the remaining instance (if necessary, add $\{v, y\}$ to \mathcal{T}). Otherwise, output a trivial negative instance.*

The tree-cut width of the augmented graphs of \mathcal{X}_s , \mathcal{Y}_s , and \mathcal{Z}_s is at most w and applying any of Reduction Rules 24–26 does not increase the tree-cut width of the augmented graph. Together, these reduction rules can be used to remove a cluttered node – or at least make it non-cluttered. To solve GSTP parameterized by the tree-cut width of the augmented graph, we solve multiple sub-instances of GSTP with respect to simple tree-cut decompositions. The basic idea is to consider a cluttered node $s \in V(S)$ with $|Y_s^{\mathcal{D}'}| < \frac{|V(G)|}{2}$. If such a node does

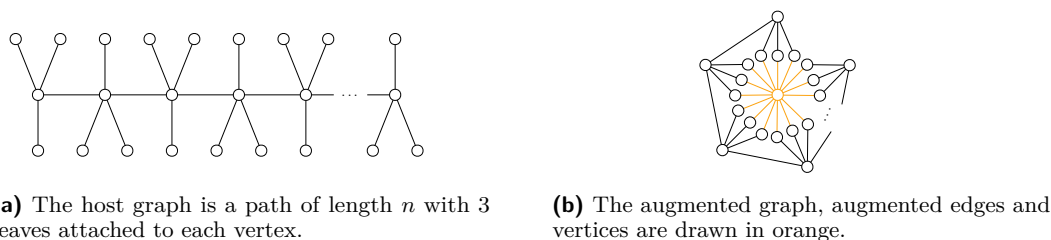


Figure 2 A family of instances where the tree-cut width of the augmented graph is unbounded.

not exist, we either re-root, or we are able to directly find a simple tree-cut decomposition of width $w + 12$. Now, we check recursively which of Reduction Rules 24–26 is applicable. By choice of s , we can ensure that overall at most $\mathcal{O}(|V(G)|^2)$ many simple instances are solved.

► **Theorem 27.** *Assume GSTP can be solved in time $r(g, k)$, given a graph of size at most g and a simple tree-cut decomposition of width at most k . Let $\mathcal{P} := (G, \mathcal{T}, d)$ be an instance of GSTP. Given a tree-cut decomposition of width w for $G^{\mathcal{T}}$, we can decide whether \mathcal{P} is positive in time $r(|G|, w + 12) \cdot \text{poly}(|\mathcal{P}|)$.*

Kim et al. [20] proved that for all graphs H we can compute a tree-cut decomposition of width $2\text{tcw}(H)$ in time $2^{\mathcal{O}(\text{tcw}(H)^2 \log \text{tcw}(H))} |V(H)|^2$. Combined with Theorem 18, we know that GSTP is FPT by the tree-cut width of the augmented graph.

► **Corollary 28.** *Let $\mathcal{P} := (G, \mathcal{T}, d)$ be an instance of GSTP. We can decide whether this instance is positive in time $2^{2^{\mathcal{O}(\text{tcw}(G^{\mathcal{T}})^8)}} \text{poly}(|\mathcal{P}|)$.*

6 STP by Tree-Cut Width

To show, how to decide an instance $\mathcal{P} := (G, T, d)$ of STP parameterized by $\text{tcw}(G)$, let $\mathcal{D} = (S, \mathcal{X})$ be a friendly tree-cut decomposition of width w . If $|T| \leq w$, we interpret \mathcal{P} as an instance of GSTP. Consider the tree-cut decomposition $\mathcal{D}' = (S', \mathcal{X}')$ for $G^{\{T\}}$, obtained by adding a new root r' containing $\text{aug}(T)$ to S , and making the old root a child of r' . The adhesion of \mathcal{D}' is bounded by $2w$. Let $s \in V(S)$. Any $t \in \text{t-chil}_{\mathcal{D}}(s)$ with $Y_t^{\mathcal{D}} \cap T \neq \emptyset$ is also a thin child of s' in \mathcal{D}' . Thus, $|\tilde{H}_s^{\mathcal{D}'}| \leq 1 + |T| + |\text{b-chil}(s)| + |X_s| \leq 2w + 3$. Note that the torso at r' in \mathcal{D}' consists of 2 vertices. Thus, the width of \mathcal{D}' is bounded by $2w + 3$ and we can use Corollary 28 to decide whether the instance is positive in FPT-time.

The same approach does not work for the case $|T| > w$, as in this case, the tree-cut width of the augmented graph is not necessarily bounded by a function of the tree-cut width of the host graph. For this consider as the host graph a path of length n where we attach to each vertex of the path 3 leaves. This graph is a tree, has tree-cut width 1, and is depicted in Figure 2a. If we take all leaves to be the terminal set, the augmented graph, depicted in Figure 2b, has tree-cut width at least $\Omega(\sqrt[4]{n})$ [27].

► **Remark 29.** There exists a family of STP instances such that the host-graph of every instance has tree-cut width 1, but the tree-cut width of the augmented graph of the instances is not bounded.

In this case (i.e., $|T| > w$), we observe that T is not contained in a single bag. By Reduction Rule 16, we can assume that $d(T) \leq w$. As $\text{tw}(G) = \mathcal{O}(w^2)$, the parameter $\text{tw}(G) + d(T)$ is bounded by a function of w . So, if we can solve STP parameterized by this

parameter, we obtain an FPT algorithm for STP parameterized by tree-cut width of the original graph. As it turns out, we can even solve GSTP by a similar parameter using a dynamic program on the tree decomposition.

► **Theorem 30.** *Let (G, \mathcal{T}, d) be an instance of the GSTP problem and set $\Sigma_D := \sum_{T \in \mathcal{T}} d(T)$. In time $2^{\mathcal{O}(\Sigma_D \cdot \text{tw}(G) \log \text{tw}(G))} |V(G)|$, we can decide whether this instance is positive.*

We remark that this algorithm directly improves upon the known algorithm for EDP with respect to the parameter $\text{tw}(G) + |T|$ [29].

Combined with the case distinction for STP, that in time $2^{\mathcal{O}(\text{tcw}(G)^2 \log \text{tcw}(G))} |V(G)|^2$ we can find a tree-cut decomposition of width $2\text{tcw}(G)$ [20], and Corollary 28 we get that STP is FPT by the parameter $\text{tcw}(G)$.

► **Corollary 31.** *Let (G, T, d) be an instance of STP. In time $2^{2^{\mathcal{O}(\text{tcw}(G)^8)}} \text{poly}(|\mathcal{G}|)$, we can decide whether this instance is positive.*

7 Conclusion and Outlook

In this paper, we provide the first fixed-parameter tractable algorithm for STEINER TREE PACKING (STP) parameterized by a structural parameter. Concretely, we show that STP is FPT when parameterized by fracture number as well as tree-cut width. This significantly extends the number of instances for which we know an exact polynomial time algorithm. Previously known polynomial time algorithms are typically based on heuristics or approximations. In case of the result that STP is FPT by $|T| + d$, we do not even know a concrete algorithm, but only that one exists [26].

To achieve this goal, we generalize the notion of the augmented graph from EDGE-DISJOINT PATHS (EDP) to GENERALIZED STEINER TREE PACKING (GSTP) and STP. This is the first result utilizing this tool on a problem where the terminals are arbitrary sets and not pairs of vertices. The notion of augmentation has been used extensively for EDP, but was originally introduced for MULTICUT [16]. Despite the fact that many parameterized complexity results for the generalized version of this problem (STEINER MULTICUT) are known [4], the augmented graph has not yet been considered in this setting. We think that augmentation will also prove to be a valuable tool for STEINER MULTICUT and other similar problems in future research.

Further, we extend all known FPT algorithms for EDP parameterized by a structural parameter to GSTP. In addition, we provide a novel FPT algorithm for GSTP parameterized by the tree-cut width of the augmented graph. This settles whether GSTP is FPT or $W[1]$ -hard parameterized by all eight commonly used structural parameters described in Section 2 with respect to the augmented graph as well as the host graph. As all these results coincide between EDP and GSTP, this also completes such a complexity classification for EDP, where previously the result that EDP is FPT by the tree-cut width of the augmented graph was not known.

For STP the established results are almost as complete. We prove for six of these eight parameters that STP is FPT. It is known, that STP is $W[1]$ -hard parameterized by treewidth, even if $|T| = 3$ [2, 1]. So, the only question remaining here is whether STP is FPT parameterized by feedback vertex set number. As EDP is $W[1]$ -hard parameterized by the feedback vertex set number of the augmented graph [14], the approach employed in this paper – generalizing results from EDP to GSTP and applying them to STP – is not suited to decide this questions. Also, the techniques used by Bodlaender et al. [2] to obtain the $W[1]$ -hardness result for INTEGER 2-COMMODITY FLOW parameterized by treewidth, which generalizes to STP with $|T| = 3$ [1], do not easily apply with respect to the feedback vertex set number. We leave answering this question to future research.

References

- 1 Ashkan Aazami, Joseph Cheriyan, and Krishnam Raju Jampani. Approximation algorithms and hardness results for packing element-disjoint steiner trees in planar graphs. *Algorithmica*, 63(1-2):425–456, 2012. doi:10.1007/S00453-011-9540-3.
- 2 Hans L. Bodlaender, Isja Mannens, Jelle J. Oostveen, Sukanya Pandey, and Erik Jan van Leeuwen. The parameterised complexity of integer multicommodity flow. In *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, pages 6:1–6:19, 2023. doi:10.4230/LIPIcs.IPEC.2023.6.
- 3 Cornelius Brand, Esra Ceylan, Robert Ganian, Christian Hatschka, and Viktoriia Korchemna. Edge-cut width: An algorithmically driven analogue of treewidth based on edge cuts. In *Graph-Theoretic Concepts in Computer Science - 48th International Workshop, WG 2022, Tübingen, Germany, June 22-24, 2022, Revised Selected Papers*, pages 98–113, 2022. doi:10.1007/978-3-031-15914-5_8.
- 4 Karl Bringmann, Danny Hermelin, Matthias Mnich, and Erik Jan van Leeuwen. Parameterized complexity dichotomy for steiner multicut. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 157–170, 2015. doi:10.4230/LIPIcs.STACS.2015.157.
- 5 Michael Burstein and Richard N. Pelavin. Hierarchical wire routing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 2(4):223–234, 1983. doi:10.1109/TCAD.1983.1270040.
- 6 S. Chen, O. Gunluk, and B. Yener. Optimal packing of group multicasting. In *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No.98, volume 3, pages 980–987 vol.3, 1998*. doi:10.1109/INFCOM.1998.662907.
- 7 James P. Cohoon and Patrick L. Heck. BEAVER: a computational-geometry-based tool for switchbox routing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 7(6):684–697, 1988. doi:10.1109/43.3208.
- 8 Matt DeVos, Jessica McDonald, and Irene Pivotto. Packing steiner trees. *J. Comb. Theory B*, 119:178–213, 2016. doi:10.1016/J.JCTB.2016.02.002.
- 9 Pavel Dvorák, Eduard Eiben, Robert Ganian, Dusan Knop, and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP: programs with few global variables and constraints. *Artif. Intell.*, 300:103561, 2021. doi:10.1016/J.ARTINT.2021.103561.
- 10 Krzysztof Fleszar, Matthias Mnich, and Joachim Spoerhase. New algorithms for maximum disjoint paths based on tree-likeness. *Math. Program.*, 171(1):433–461, September 2018. doi:10.1007/s10107-017-1199-3.
- 11 Robert Ganian, Eun Jung Kim, and Stefan Szeider. Algorithmic applications of tree-cut width. *SIAM J. Discret. Math.*, 36(4):2635–2666, 2022. doi:10.1137/20M137478X.
- 12 Robert Ganian and Viktoriia Korchemna. Slim tree-cut width. In *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, pages 15:1–15:18, 2022. doi:10.4230/LIPIcs.IPEC.2022.15.
- 13 Robert Ganian and Sebastian Ordyniak. The power of cut-based parameters for computing edge-disjoint paths. *Algorithmica*, 83(2):726–752, October 2020. doi:10.1007/s00453-020-00772-w.
- 14 Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. On structural parameterizations of the edge disjoint paths problem. *Algorithmica*, 83(6):1605–1637, 2021. doi:10.1007/s00453-020-00795-3.
- 15 Naveen Garg, Rohit Khandekar, Keshav Kunal, and Vinayaka Pandit. Bandwidth maximization in multicasting. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, pages 242–253, 2003. doi:10.1007/978-3-540-39658-1_24.
- 16 Georg Gottlob and Stephanie Tien Lee. A logical approach to multicut problems. *Inf. Process. Lett.*, 103(4):136–141, 2007. doi:10.1016/J.IPL.2007.03.005.

- 17 Martin Grötschel, Alexander Martin, and Robert Weismantel. The steiner tree packing problem in VLSI design. *Math. Program.*, 77:265–281, 1997. doi:10.1007/BF02614374.
- 18 Niko Hastrich and Kirill Simonov. Structural parameterization of steiner tree packing. *CoRR*, abs/2505.09250, 2025. doi:10.48550/arXiv.2505.09250.
- 19 Petteri Kaski. Packing steiner trees with identical terminal sets. *Inf. Process. Lett.*, 91(1):1–5, 2004. doi:10.1016/j.ipl.2004.03.006.
- 20 Eun Jung Kim, Sang-il Oum, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. An FPT 2-approximation for tree-cut decomposition. *Algorithmica*, 80(1):116–135, 2018. doi:10.1007/S00453-016-0245-5.
- 21 Matthias Kriesell. Edge-disjoint trees containing some given vertices in a graph. *J. Comb. Theory B*, 88(1):53–65, 2003. doi:10.1016/S0095-8956(02)00013-8.
- 22 W. K. Luk. A greedy switch-box router. *Integr.*, 3(2):129–149, 1985. doi:10.1016/0167-9260(85)90029-X.
- 23 A. Martin and R. Weismantel. Packing paths and steiner trees: routing of electronic circuits. *CWI Quarterly*, 6(3):185–204, September 1993.
- 24 Yoram Ofek and Bülent Yener. Reliable concurrent multicast from bursty sources. *IEEE J. Sel. Areas Commun.*, 15(3):434–444, 1997. doi:10.1109/49.564140.
- 25 William R. Pulleyblank. Two steiner tree packing problems (extended abstract). In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 383–387, 1995. doi:10.1145/225058.225163.
- 26 Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/JCTB.1995.1006.
- 27 Paul Wollan. The structure of graphs not admitting a fixed immersion. *J. Comb. Theory, Ser. B*, 110:47–66, 2015. doi:10.1016/J.JCTB.2014.07.003.
- 28 Miao Zhao, Bin Jia, Mingquan Wu, Heather Yu, and Yang Xu. Software defined network-enabled multicast for multi-party video conferencing systems. In *2014 IEEE International Conference on Communications (ICC)*, pages 1729–1735, 2014. doi:10.1109/ICC.2014.6883572.
- 29 Xiao Zhou, Syurei Tamura, and Takao Nishizeki. Finding edge-disjoint paths in partial k -trees. *Algorithmica*, 26(1):3–30, 2000. doi:10.1007/s004539910002.