Parameterized Spanning Tree Congestion

Michael Lampis
Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France

Valia Mitsou
Université Paris Cité, IRIF, CNRS, 75205, Paris, France

Edouard Nemery
Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France

Yota Otachi
Nagoya University, Nagoya, Japan

Manolis Vasilakis
Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France

Daniel Vaz
LIGM, Université Gustave Eiffel, CNRS, ESIEE Paris, 77454 Marne-la-Vallée, France

Abstract

In this paper we study the Spanning Tree Congestion problem, where we are given an undirected graph $G = (V, E)$ and are asked to find a spanning tree T of minimum maximum *congestion*. Here, the congestion of an edge $e \in T$ is the number of edges $uv \in E$ such that the (unique) path from u to v in T traverses e. We consider this well-studied NP-hard problem from the point of view of (structural) parameterized complexity and obtain the following results:

- We resolve a natural open problem by showing that Spanning Tree Congestion is not FPT parameterized by treewidth (under standard assumptions). More strongly, we present a generic reduction which applies to (almost) any parameter of the form "vertex-deletion distance to class $\mathcal{C}$", thus obtaining W[1]-hardness for more restricted parameters, including tree-depth plus feedback vertex set, or incomparable to treewidth, such as twin cover. Via a slight tweak of the same reduction we also show that the problem is NP-complete on graphs of modular-width 4.
- Even though it is known that Spanning Tree Congestion remains NP-hard on instances with only one vertex of unbounded degree, it is currently open whether the problem remains hard on bounded-degree graphs. We resolve this question by showing NP-hardness on graphs of maximum degree 8.
- Complementing the problem's W[1]-hardness for treewidth, we formulate an algorithm that runs in time roughly $(k + w)^{\mathcal{O}(w)}$, where k is the desired congestion and w the treewidth, improving a previous argument for parameter $k + w$ that was based on Courcelle's theorem. This explicit algorithm pays off in two ways: it allows us to obtain an FPT approximation scheme for parameter treewidth, that is, a $(1 + \varepsilon)$-approximation running in time roughly $(w/\varepsilon)^{\mathcal{O}(w)}$; and it leads to an exact FPT algorithm for parameter clique-width$+k$ via a Win/Win argument.
- Finally, motivated by the problem's hardness for most standard structural parameters, we present FPT algorithms for several more restricted cases, namely, for the parameters vertex-deletion distance to clique; vertex integrity; and feedback edge set, in the latter case also achieving a single-exponential running time dependence on the parameter.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Treewidth, Graph Width Parameters

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.65

Related Version *Full Version*: https://arxiv.org/abs/2410.08314

Funding This work is partially supported by ANR project ANR-21-CE48-0022 (S-EX-AP-PE-AL). *Yota Otachi*: JSPS KAKENHI Grant Numbers JP21K11752, JP22H00513, JP24H00697.

1 Introduction

One of the most well-studied types of problems in network optimization involves finding, for a given graph G, a spanning tree of G that optimizes a certain objective. In this paper we focus on a well-known problem of this type called Spanning Tree Congestion. The motivation of this problem can be summarized as follows: Every edge e of a spanning tree T is selected with the goal of maintaining connectivity between the two parts of the graph given by the two components of $T - e$. We can then think of every other edge e' with endpoints in both components of $T - e$ as being "simulated" by a path in T that traverses e; hence, the more such edges exist, the more e is used and "congested". Our optimization goal, then, is to find a tree where all edges have congestion as low as possible, because in such a tree each selected edge is responsible for simulating only a small number of non-selected edges and therefore the tree can be thought of as a sparse approximate representation of the original graph. Equivalently, for a spanning tree T of G, we say that the *detour* of an edge $\{u, v\} \in E(G)$ in T is the unique u–v path in T. The number of detours that traverse an edge in T constitutes its congestion, while the congestion of T is defined as the maximum over the congestion of all of its edges.[1] The *spanning tree congestion* of G, denoted by $\mathrm{stc}(G)$, is the minimum congestion over all of its spanning trees, and Spanning Tree Congestion asks, given G and an integer k, whether $\mathrm{stc}(G) \leq k$.

Spanning trees of low congestion are a natural notion that is well-studied both from the combinatorial and the algorithmic point of view. Unsurprisingly, Spanning Tree Congestion is NP-complete [57, Section 5.6]. It therefore makes sense to study the parameterized complexity of this problem, as parameterized complexity is one of the main tools for dealing with computational intractability.[2] The most natural parameter one could consider is perhaps the objective value k, but unfortunately the problem is known to be NP-hard for all fixed $k \geq 5$ [9, 59]. This motivates us to focus on *structural* graph parameters, where much less is currently known. Indeed, it is so far open whether Spanning Tree Congestion is fixed-parameter tractable for treewidth, which is the most widely studied parameter of this type (this is mentioned as an open problem in [64]). What is known, however, is that the problem is FPT when parameterized by both treewidth and k [9] and that the problem is NP-hard on graphs of clique-width at most 3 (implied by the NP-hardness on chain graphs [61]).

Our Contribution. Our aim in this paper is to present a clarified and much more detailed picture of how the complexity of Spanning Tree Congestion depends on treewidth and other notions of graph structure (see Figure 1 for a synopsis of our results).

We begin our work by considering the natural open problem we mentioned above, namely whether Spanning Tree Congestion is FPT parameterized by treewidth. We answer this question in the negative and indeed prove something much stronger: Let $\mathcal{C}$ be any class of graphs that satisfies the (very mild) requirement that for each integer i there exists a connected graph in $\mathcal{C}$ that has i vertices. Then, for any such class $\mathcal{C}$, Spanning Tree Congestion is W[1]-hard parameterized by the vertex deletion distance to a disjoint union of graphs belonging to $\mathcal{C}$. As a corollary, if we set $\mathcal{C}$ to be the class of all stars, Spanning Tree Congestion is shown to be W[1]-hard for parameter vertex-deletion distance to

[1] This has also been referred to as the *edge remember number* of G relative to T in the literature [8, Section 11].

[2] Throughout the paper we assume that the reader is familiar with the basics of parameterized complexity, as given in standard textbooks [21].

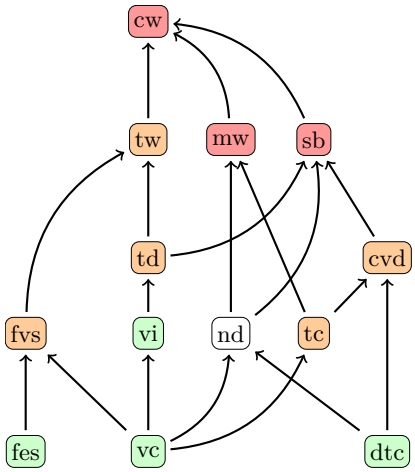

Figure 1 Our results and hierarchy of the related graph parameters (see full version for their definitions). For any graph, if the parameter at the tail of an arrow is a constant, that is also the case for the one at its head. Green indicates that the problem is FPT (Theorems 23–25), orange W[1]-hardness (Theorem 1), and red para-NP-hardness (Theorem 9). We additionally show fixed-parameter tractability by $\mathrm{cw}+k$ (Theorem 22), para-NP-hardness by maximum degree (Theorem 11), as well as develop an FPT-AS for tw (Theorem 21). Prior to this work, it was only known that the problem is FPT by $\mathrm{tw}+k$ [9] and para-NP-hard by clique-width [61].

star-forest, hence also for parameter tree-depth plus feedback vertex set (and consequently also for treewidth). Alternatively, by setting $\mathcal{C}$ to be the class of all cliques, our proof establishes W[1]-hardness parameterized by the cluster vertex deletion number, and more strongly by the twin-cover of the input graph [35]. With a couple of modifications, we then show in Theorem 9 that Spanning Tree Congestion remains NP-complete even on graphs of modular-width at most 4, linear clique-width 3, and shrub-depth 2, improving over the previously mentioned hardness result of Okamoto, Otachi, Uehara, and Uno for clique-width 3 [61].

Moving on, we consider the tractability of the problem in graphs of constant degree. All previous NP-hardness results [9, 59] require at least one vertex of unbounded degree. However, assuming that the graph has bounded degree seems potentially algorithmically useful, as recent work by Kolman [48] shows that instances of polylogarithmic maximum degree are amenable to a polynomial approximation algorithm of ratio $o(n)$ (this is non-trivial, as the best known ratio on general graphs is $n/2$, trivially achieved by any spanning tree [64]).[3] Our next result is to answer an open question posed by Kolman [48] and show that the problem in fact remains NP-hard even on graphs of degree at most 8 (Theorem 11). To this end, we make use of a novel gadget based on grids, simulating the *double-weighted edges* introduced by Luu and Chrobak [59].

Coming back to treewidth, we recall that Bodlaender, Fomin, Golovach, Otachi, and van Leeuwen [9] showed that, when k is part of the parameter, Spanning Tree Congestion is expressible in MSO_2 logic, thus due to Courcelle's theorem [20] fixed-parameter tractable by $\mathrm{tw}+k$. We improve upon this by providing an explicit FPT algorithm of running time $(\mathrm{tw}+k)^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$. In addition to providing a concrete reasonable upper-

[3] A very recent work by Kolman [49] improves over this and presents a polynomial-time approximation algorithm of ratio $\mathcal{O}(\Delta \cdot \log^{3/2} n)$, where Δ denotes the maximum degree of the graph.

bound on the running time (which cannot be done with Courcelle's theorem), this explicit algorithm allows us to obtain two further interesting extensions. First, using a technique introduced by Lampis [54], we develop an efficient FPT approximation scheme (FPT-AS) when parameterized solely by tw, that is, a $(1+\varepsilon)$-approximate algorithm running in time $(\mathrm{tw}/\varepsilon)^{\mathcal{O}(\mathrm{tw})}n^{\mathcal{O}(1)}$; notice that an efficient FPT-AS is the best we can hope for in this setting, given the W[1]-hardness following from Theorem 1. Second, using a Win/Win argument based on a result of Gurski and Wanke [42], we lift our algorithm to also show an explicit FPT algorithm for the more general parameter $\mathrm{cw}+k$, where cw denotes the clique-width of the input graph.

Finally, given all the previously mentioned hardness results, we next aim to determine which structural parameters *do* render the problem fixed-parameter tractable. As a consequence of Theorem 1, the problem remains intractable even on very restricted (dense *and* sparse) graph classes, we must therefore focus on parameters that evade this hardness result. We consider three cases: First, the parameter "distance to clique" is not covered by Theorem 1 because the graph obtained after removing the deletion set has one component; we show in Theorem 23 that Spanning Tree Congestion is FPT in this case. Second, the parameter vertex integrity is not covered by Theorem 1, as all components of the graph obtained after removing the deletion set have bounded size; we show in Theorem 24 that Spanning Tree Congestion is FPT in this case as well, via a reduction to an ILP with an FPT number of variables. Third, we consider the parameter feedback edge set, which also falls outside the scope of Theorem 1, and obtain a linear kernel, which leads to an FPT algorithm with single-exponential parameter dependence for this case.

Related Work. Spanning Tree Congestion was formally introduced by Ostrovskii [62], though it had also been previously studied under a different name [65]. There is a plethora of graph-theoretical results in the literature [16, 45, 51, 52, 56, 58, 63], as well as some algorithmic ones [9, 10, 17, 61]. See also the survey of Otachi [64]. Spanning Tree Congestion is known to be polynomial-time solvable if $k \leq 3$ [9], and NP-hard for all fixed $k \geq 5$ [59]; the case $k = 4$ remains open. Okamoto et al. [61] have presented an algorithm running in time $2^n n^{\mathcal{O}(1)}$, improving over the brute-force one. Regarding specific graph classes, it is known to be polynomial-time solvable for outerplanar graphs [10], two-dimensional Hamming graphs [51], complete k-partite graphs, and two-dimensional tori [52]. On the other hand, it is NP-hard for planar, split, and chain graphs [9, 61], with the latter result implying NP-hardness for graphs of clique-width at most 3. For fixed k, the problem is expressible in MSO_2 logic [9], thus due to standard metatheorems [20] it is FPT by $\mathrm{tw}+k$. Kozawa, Otachi, and Yamazaki [52] showed a combinatorial bound (then improved in [9]) which proves that for all graphs G, $\mathrm{tw}(G) = \mathcal{O}(\mathrm{stc}(G)\Delta(G))$, where Δ denotes the maximum degree of the input graph. Combining these, Bodlaender et al. [9] show that Spanning Tree Congestion is FPT by $\Delta + k$, as well as that it is solvable in polynomial time for fixed k on apex-minor-free graphs. There are also some results regarding the problem's approximability [9, 48, 49, 59].

Finding a spanning tree T of a connected graph such that T adheres to some constraint, i.e., $T \in \mathcal{T}$ for some family of trees $\mathcal{T}$, is an interesting combinatorial question in its own right, that oftentimes finds applications to other algorithmic problems. Examples of studied properties include trees of maximum number of branch or leaf vertices [12, 23, 30, 40, 41, 47, 53], of minimum maximum degree [66, 11], and others [2, 6, 43, 60]. One such important variant of Spanning Tree Congestion is the Tree Spanner problem [15], where one asks for a spanning tree of minimum stretch. The latter has been extensively studied [1, 13, 26, 27, 28, 29, 32], and the two problems are known to be tightly connected, especially on planar graphs [59, 64].

Lastly, a closely-related structural graph parameter is the so-called *edge-cut width* [14] or *local feedback edge number* [38]. This is, roughly speaking, the vertex variant of spanning tree congestion, where one asks to minimize the maximum congestion over the *vertices* of the spanning tree, where the congestion of a vertex is defined as the number of detours containing it.[4] Those parameters have been recently used in the setting of parameterized complexity to show various tractability and incompressibility results [14, 38], and we believe that our work might provide insights into the parameterized (in)tractability of their computation.

Organization. In Section 2 we discuss the general preliminaries. Subsequently, in Section 3 we present the various hardness results, followed by Section 4 where we present the explicit FPT algorithm when parameterized by $\mathrm{tw} + k$, as well as the two results that make use of this. Moving on, in Section 5 we present various fixed-parameter tractability results. Lastly, in Section 6 we present the conclusion as well as some directions for future research. Proofs of statements marked with ($\star$) are deferred to the appendix.

2 Preliminaries

Throughout the paper we use standard graph notation [24], and we assume familiarity with the basic notions of parameterized complexity [21]. All graphs considered are undirected without loops. For a graph $G = (V, E)$ and $S \subseteq V$, we denote the *open neighborhood of* S by $N_G(S) = (\bigcup_{s \in S} N_G(s)) \setminus S$. For $x, y \in \mathbb{Z}$, let $[x, y] = \{z \in \mathbb{Z} \mid x \leq z \leq y\}$, while $[x] = [1, x]$.

Let $G = (V, E)$ be a connected graph and T a spanning tree of G. The *detour* for $\{u, v\} \in E$ in T is the unique u–v path in T. Note that the detour of $e \in E(T)$ is e itself. The *congestion* of $e \in E(T)$, denoted $\mathrm{cng}_{G,T}(e)$, is the number of edges in G whose detours contain e. In other words, $\mathrm{cng}_{G,T}(e)$ is the size of the fundamental cutset of T defined by e, that is, $\mathrm{cng}_{G,T}(e) = |E(V(T_{e,1}), V(T_{e,2}))|$, where $E(X, Y) = \{\{x, y\} \in E \mid x \in X, y \in Y\}$ for disjoint vertex sets $X, Y \subseteq V$ and $T_{e,1}$ and $T_{e,2}$ are the two subtrees of T obtained by cutting e. The *congestion* of T, denoted $\mathrm{cng}_G(T)$, is defined as the maximum over the congestion of all edges in T, i.e., $\mathrm{cng}_G(T) = \max_{e \in E(T)} \mathrm{cng}_{G,T}(e)$. The *spanning tree congestion* of G, denoted $\mathrm{stc}(G)$, is the minimum congestion over all spanning trees of G. Given a connected graph G and an integer $k \in \mathbb{Z}^+$, Spanning Tree Congestion asks to determine whether $\mathrm{stc}(G) \leq k$.

Let $G = (V, E)$ be a graph. The *vertex integrity* of G, denoted by $\mathrm{vi}(G)$, is the minimum integer k such that there is a vertex set $S \subseteq V$ with $|S| + \max_{C \in \mathtt{cc}(G-S)} |V(C)| \leq k$, where $\mathtt{cc}(G - S)$ denotes the set of connected components in $G - S$. The *twin-cover number* of G, denoted by $\mathrm{tc}(G)$, is the size of the smallest vertex set (called *twin-cover*) whose removal results in a cluster graph, with the constraint that each clique is composed of true twins in G [35]. If we drop the constraint, this is the *cluster vertex deletion number* [25], denoted by $\mathrm{cvd}(G)$. The *modular-width* of G ([33, 34]) is the smallest integer k such that, either $|V| \leq k$, or V can be partitioned into at most $k' \leq k$ sets $V_1, \ldots, V_{k'}$, with the following two properties: (i) for all $i \in [k']$, V_i is a module of G, (ii) for all $i \in [k']$, $G[V_i]$ has modular width at most k. For the definition of the rest of the parameters appearing in Figure 1 as well as known relations between them, we refer to Graph Parameters paragraph of the full version.

[4] Analogously, this has been referred to as the *vertex remember number* of G relative to T in the literature [8, Section 11].

3 Hardness results

In this section we present various hardness results for Spanning Tree Congestion. We start with showing in Section 3.1 that the problem is W[1]-hard parameterized by the distance to the disjoint union of graphs in $\mathcal{C}$, for any family of graphs $\mathcal{C}$ that contains connected graphs of any order. Moving on, in Section 3.2 we adapt our proof and prove NP-hardness for graphs of modular-width at most 4. Finally, in Section 3.3 we introduce a novel gadget simulating the double-weighted edges previously used by Luu and Chrobak [59], of which we make use of in order to show NP-hardness for graphs of constant maximum degree.

3.1 Distance to disjoint union

We start by stating the main theorem of this subsection.

▶ **Theorem 1.** Spanning Tree Congestion *is W[1]-hard parameterized by vertex-deletion distance to disjoint union of graphs in $\mathcal{C}$, where $\mathcal{C}$ is any graph class such that, for all $p \in \mathbb{Z}^+$, $\mathcal{C}$ contains a connected p-vertex graph which can be generated in time $p^{\mathcal{O}(1)}$.*

By taking the set of stars as $\mathcal{C}$, Theorem 1 implies the W[1]-hardness parameterized by distance to star forest.

▶ **Corollary 2.** Spanning Tree Congestion *is W[1]-hard parameterized by distance to star forest (and thus by tree-depth + feedback vertex set number).*

If $\mathcal{C}$ is the set of complete graphs, Theorem 1 implies W[1]-hardness parameterized by cluster vertex deletion number [25]. In fact, as we will later see, the proof of Theorem 1 more strongly implies W[1]-hardness parameterized by twin-cover number [35].

▶ **Corollary 3.** Spanning Tree Congestion *is W[1]-hard parameterized by twin-cover number.*

For an edge-weighted graph $G = (V, E; \mathrm{w})$ with $\mathrm{w}\colon E \to \mathbb{Z}^+$, we define its spanning tree congestion by setting the congestion of an edge e in a spanning tree T of G as $\mathrm{cng}_{G,T}(e) = \mathrm{w}(E(V(T_{e,1}), V(T_{e,2})))$, where $\mathrm{w}(F) = \sum_{e \in F} \mathrm{w}(e)$ for $F \subseteq E$. The following proposition provides a connection between the weighted and the unweighted case.

▶ **Proposition 4** ([9])**.** *Let $G = (V, E; \mathrm{w})$ be an edge-weighted graph and $G' = (V', E')$ be an unweighted graph obtained from G by replacing each weighted edge $\{u, v\}$ with $\mathrm{w}(\{u, v\})$ internally vertex-disjoint u–v paths of any lengths, where one of them may be $\{u, v\}$ itself. Then,* $\mathrm{stc}(G) = \mathrm{stc}(G')$.

In the following, we prove Theorem 1 by a reduction from Unary Bin Packing. Given unary encoded integers $t, a_1, \ldots, a_n \in \mathbb{Z}^+$ with $\sum_{i \in [n]} a_i = tB$, Unary Bin Packing asks whether there is a partition $(A_1, \ldots, A_t)$ of $[n]$ such that $\sum_{i \in A_j} a_i = B$ for each $j \in [t]$. It is known that Unary Bin Packing is W[1]-hard parameterized by t [46]. We assume that $t \geq 3$ since otherwise the problem can be solved in polynomial time. We now proceed to presenting the reduction.

Construction. Let $\mathcal{I} = \langle t; a_1, \ldots, a_n \rangle$ be an instance of Unary Bin Packing with $\sum_{i \in [n]} a_i = tB$. For each $i \in [n]$, let $G_i = (V_i, E_i)$ be a connected a_i-vertex graph belonging to $\mathcal{C}$. We set $k = 5(t-1)B$ and construct an edge-weighted graph $G = (V, E; \mathrm{w})$ as follows.

1. Take the disjoint union of all G_i for $i \in [n]$.
2. Add a set of vertices $Q = \{v_j \mid j \in [t]\}$ and all possible edges between Q and $\bigcup_{i\in[n]} V_i$.
3. Add a vertex r and all possible edges between r and Q.
4. Set $\mathrm{w}(\{r, v_j\}) = 3(t-1)B$ for $j \in [n]$, and $\mathrm{w}(e) = 1$ for all other edges.

Proposition 4 implies that we can construct, in time polynomial in n and B, an unweighted graph G' from G such that $\mathrm{stc}(G) = \mathrm{stc}(G')$, where each weighted edge e of weight $\mathrm{w}(e) \geq 2$ in G is replaced by $\mathrm{w}(e)$ internally vertex-disjoint paths of length 2 between the endpoints of e. Observe that $G' - (\{r\} \cup Q)$ is the disjoint union of all G_i and the singleton components that correspond to the middle vertices of the paths replacing weighted edges. Since the single-vertex graph belongs to $\mathcal{C}$, G' has distance $|\{r\} \cup Q| = t+1$ to disjoint union of graphs in $\mathcal{C}$. We can also see that if $\mathcal{C}$ is the set of complete graphs, then $\{r\} \cup Q$ is a twin-cover. Thus, to prove Theorem 1 (together with Corollary 3), it suffices to show that $\mathrm{stc}(G) \leq k$ if and only if $\mathcal{I}$ is a yes-instance of Unary Bin Packing.

▶ **Lemma 5.** *If $\mathcal{I}$ is a yes-instance of* Unary Bin Packing, *then* $\mathrm{stc}(G) \leq k$.

Proof. Let $(A_1, \ldots, A_t)$ be a partition of $[n]$ such that $\sum_{i\in A_j} a_i = B$ for each $j \in [t]$. We construct a spanning tree T of G by setting

$$E(T) = \{\{r, v_j\} \mid j \in [t]\} \cup \bigcup_{j\in[t]} \{\{u, v_j\} \mid u \in V_i,\ i \in A_j\}.$$

Each edge $\{u, v_j\} \in E(T)$ with $u \in V_i$ has congestion $\deg_G(u) = t + \deg_{G_i}(u) \leq t + a_i - 1 < k$ since u is a leaf of T.

For each v_j, let S_j be the set of vertices of the component of $T - \{r, v_j\}$ containing v_j. By the construction, we can see that $S_j = \{v_j\} \cup \bigcup_{i\in A_j} V_i$. Thus,

$$\begin{aligned}\mathrm{cng}_{G,T}(\{r, v_j\}) &= \mathrm{w}(E(S_j, V \setminus S_j)) \\ &= \mathrm{w}(\{r, v_j\}) + |E(\{v_j\},\ \textstyle\bigcup_{i\in[n]\setminus A_j} V_i)| + |E(\bigcup_{i\in A_j} V_i,\ Q \setminus \{v_j\})| \\ &= 3(t-1)B + (t-1)B + (t-1)B = k.\end{aligned}$$

◀

▶ **Lemma 6.** *If* $\mathrm{stc}(G) \leq k$, *then $\mathcal{I}$ is a yes-instance of* Unary Bin Packing.

Proof. Let T be a spanning tree of G with congestion at most k.

We first show that $\{r, v_j\} \in E(T)$ for every $j \in [t]$. Suppose to the contrary that $\{r, v_j\} \notin E(T)$ for some $j \in [t]$. In this case, the r–v_j path P in T first visits some $v_{j'}$ with $j' \neq j$; i.e., $P = (r, v_{j'}, \ldots, v_j)$. This implies that the congestion of the edge $\{r, v_{j'}\} \in E(T)$ is at least $\mathrm{w}(\{r, v_j\}) + \mathrm{w}(\{r, v_{j'}\}) = 6(t-1)B > k$, a contradiction.

Next we show that for each $i \in [n]$, there exists exactly one index $j \in [t]$ such that at least one vertex in V_i is adjacent to v_j in T.

▷ Claim 7. For all $i \in [n]$, there exists $j \in [t]$ such that $N_T(V_i) \cap Q = \{v_j\}$.

Proof. There is at least one such $j \in [t]$ since T is a spanning tree, i.e., $N_T(V_i) \cap Q \neq \varnothing$. Suppose to the contrary that for some $h \in [n]$, there are two or more vertices in Q that have neighbors in V_h. Since V_h induces a connected subgraph of G (i.e., $G_h = (V_h, E_h)$) and each edge $\{r, v_j\}$ is included in T, there is at least one edge $e_h \in E_h$ such that the detour for e_h in T contains r. Let $R = \sum_{j\in[t]} \mathrm{cng}_{G,T}(\{r, v_j\})$. The edge e_h contributes 2 to R as its detour passes through two edges incident to r. Each edge $\{r, v_j\}$ contributes $\mathrm{w}(\{r, v_j\})$ to R. Now, for $u \in \bigcup_{i\in[n]} V_i$, let $j_u \in [t]$ be the index such that v_{j_u} appears in the u–r path $P_{u,r}$ in T. Since $\{r, v_j\} \in E(T)$ for each $j \in [t]$, such j_u is unique and v_{j_u} appears right before r

in $P_{u,r}$. Observe that for each $j \in [t] \setminus \{j_u\}$, the detour for $\{u, v_j\} \in E$ in T consists of $P_{u,r}$ and v_j, where v_j appears right after r. This detour contributes 1 to the congestion of each of the edges $\{r, v_{j_u}\}$ and $\{r, v_j\}$. The discussion so far implies that $R > tk$ as follows:

$$R \geq 2 + \sum_{j \in [t]} \mathrm{w}(\{r, v_j\}) + \sum_{u \in \bigcup_{i \in [n]} V_i} 2(t-1) = 2 + 3(t-1)Bt + 2(t-1)tB = 2 + tk.$$

This contradicts the assumption that each edge of T has congestion at most k and thus $R \leq kt$. ◁

For $j \in [t]$, let $A_j = \{i \mid \exists u \in V_i,\ \{u, v_j\} \in E(T)\}$. Claim 7 implies that $(A_1, \ldots, A_t)$ is a partition of $[n]$. In particular, the set of vertices of the component of $T - \{r, v_j\}$ containing v_j is $\{v_j\} \cup \bigcup_{i \in A_j} V_i$. This implies that

$$\begin{aligned}
\mathrm{cng}_{G,T}(\{r, v_j\}) &= \mathrm{w}(\{r, v_j\}) + |E(\{v_j\},\ \textstyle\bigcup_{i \in [n] \setminus A_j} V_i)| + |E(\textstyle\bigcup_{i \in A_j} V_i,\ Q \setminus \{v_j\})| \\
&= 3(t-1)B + \sum_{i \in [n] \setminus A_j} a_i + (t-1) \sum_{i \in A_j} a_i \\
&= 3(t-1)B + tB + (t-2) \sum_{i \in A_j} a_i.
\end{aligned}$$

Combining this with the assumption $\mathrm{cng}_{G,T}(\{r, v_j\}) \leq k = 5(t-1)B$, we obtain that $\sum_{i \in A_j} a_i \leq B$. (Recall that $t \geq 3$.) Since $\sum_{i \in [n]} a_i = tB$, we have $\sum_{i \in A_j} a_i = B$ for each $j \in [t]$. ◀

3.2 Modular-width

In this subsection we consider Spanning Tree Congestion parameterized by the modular-width of the input graph [34]. We prove that the problem remains NP-complete even on graphs of modular-width at most 4; there is no graph of modular-width 3 since there is no prime graph with three vertices, therefore our result leaves open the case of graphs of modular-width at most 2. Our result, along with the fact that graphs of modular-width at most 4 have clique-width at most 3 (Theorem 8), improves upon previous work by Okamoto et al. [61], who showed that the problem is NP-hard for graphs of clique-width at most 3. As a matter of fact, the graph we construct has linear clique-width [31] at most 3, thus directly improving over said result. Nevertheless, Theorem 8 is an interesting result in its own right which we were not able to find in the literature. Furthermore, the family of graphs constructed by our reduction has shrub-depth at most 2 [36, 37], therefore yielding para-NP-hardness for this parameterization as well.

▶ **Theorem 8** (⋆). *Any graph of modular-width at most* 4 *has clique-width at most* 3.

In the following, we mostly follow the proof of Theorem 1 by setting $\mathcal{C}$ to be the class of all complete graphs, albeit with a few adaptations. First, the starting point of our reduction is the strongly NP-complete 3-Partition problem. Second, we notice that even though the edge-weighted graph G produced in the proof of Theorem 1 has modular-width 2 when $\mathcal{C}$ is the class of complete graphs (let one module contain the vertices of Q and the other the rest), that is not the case for the unweighted graph G' produced by Proposition 4. In order to overcome this bottleneck, we emulate the weights of the edges by introducing a sufficiently large clique in our construction.

Given $3n$ unary encoded (not necessarily distinct) integers $a_i \in \mathbb{Z}^+$ for $i \in [3n]$, where $\sum_{i\in[3n]} a_i = nB$ and $B/4 < a_i < B/2$ for all $i \in [3n]$, 3-Partition asks whether there is a partition $(A_1, \ldots, A_n)$ of $[3n]$ such that $\sum_{i\in A_j} a_i = B$ for all $j \in [n]$; notice that the bounds on the values of a_i imply that if $\sum_{i\in A_j} a_i = B$, then $|A_j| = 3$. It is well-known that 3-Partition is strongly NP-complete [39, Theorem 4.4].

▶ **Theorem 9** (⋆). Spanning Tree Congestion *is NP-complete on graphs of modular-width at most* 4*, linear clique-width at most* 3*, and shrub-depth at most* 2*.*

3.3 Maximum degree

The last result of Section 3 is the NP-hardness on graphs of constant maximum degree. We first present an alternative gadget for the double-weighted edges which we subsequently use in our proof.

Double-weighted edges. For the sake of simplicity in our reductions, we will use the concept of double-weighted edges introduced by Luu and Chrobak [59]. A *double edge-weighted graph* $G = (V, E; \mathrm{w}_1, \mathrm{w}_2)$ is a graph with two edge-weight functions $\mathrm{w}_1, \mathrm{w}_2 \colon E \to \mathbb{Z}^+$. For simplicity, let $\mathrm{w}(e)$ denote the tuple $(\mathrm{w}_1(e), \mathrm{w}_2(e))$ for $e \in E$. Let T be a spanning tree of G. When considering the congestion of T, the double weights of the edges work slightly differently from the ordinal (single) edge weight considered in Section 3.1. If $e \notin E(T)$, then it contributes $\mathrm{w}_1(e)$ to the congestions of the edges in the detour for e in T; if $e \in E(T)$, it contributes $\mathrm{w}_2(e)$ to the congestion of itself. That is, for $e \in E(T)$, it holds that

$$\mathrm{cng}_{G,T}(e) = \mathrm{w}_1(E(V(T_{e,1}), V(T_{e,2})) \setminus \{e\}) + \mathrm{w}_2(e).$$

Luu and Chrobak [59] showed that for every positive integer k, a double-weighted edge e with $\mathrm{w}_1(e) \le \mathrm{w}_2(e) < k$ can be replaced with a gadget consisting of unweighted edges (i.e., edges e' with $\mathrm{w}_1(e') = \mathrm{w}_2(e') = 1$) without changing the property of having spanning tree congestion at most k. Their gadget increases the degree of one endpoint of e by $\mathrm{w}_1(e) - 1$ and the other by $\mathrm{w}_1(e) \cdot (\mathrm{w}_2(e) - \mathrm{w}_1(e) + 1) - 1$. Because of this increase of the degree, and as in the following reduction proving Theorem 11, w_2 is unbounded, we cannot use their gadget in our proof.

In the following, we present an alternative gadget for double-weighted edges that does not increase the degree too much. For an integer $n \ge 2$, the $n \times n$ grid is the Cartesian product of two n-vertex paths. We call the degree-2 vertices in a grid its *corners*. It is known that the spanning tree congestion of the $n \times n$ grid is n [16, 45].

▶ **Lemma 10** (⋆). *Let k be a positive integer and $G = (V, E; \mathrm{w}_1, \mathrm{w}_2)$ be a double edge-weighted graph with an edge $e = \{u, v\} \in E$ satisfying that $\mathrm{w}_1(e) < \mathrm{w}_2(e) < k$. Let G' be the graph obtained from G by the following modification (see Figure 2 (left)):*

1. *remove e;*
2. *add $\mathrm{w}_1(e)$ copies of the $(\mathrm{w}_2(e) - \mathrm{w}_1(e) + 1) \times (\mathrm{w}_2(e) - \mathrm{w}_1(e) + 1)$ grid;*
3. *for each grid added in the previous step, add edges $\{u, c\}$ and $\{v, c'\}$, where c is an arbitrary corner of the grid and c' is the opposite corner (i.e., the corner furthest from c);*
4. *for each new edge $f \in E(G') \setminus E(G)$, set $\mathrm{w}_1(f) = \mathrm{w}_2(f) = 1$.*

Then, $\mathrm{stc}(G) \le k$ if and only if $\mathrm{stc}(G') \le k$. The degrees of u and v increase by $\mathrm{w}_1(e) - 1$ and the maximum degree among newly added vertices is at most 4*.*

The problem $(3, \mathrm{B2})$-SAT (also appearing in the literature as 2P2N-3SAT) is a restricted version of 3-SAT: an instance of $(3, \mathrm{B2})$-SAT consists of a set X of n variables and a set C of m clauses such that each clause has exactly three literals corresponding to three different

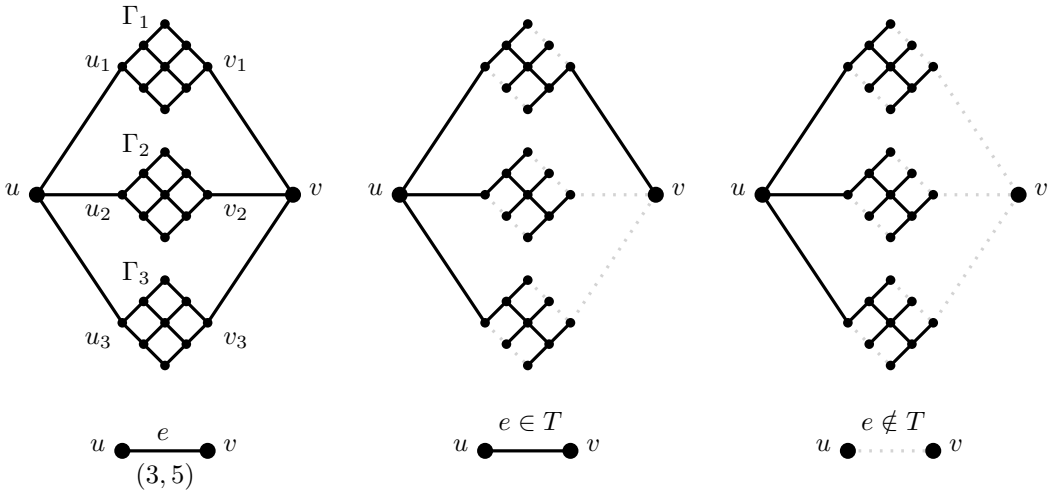

Figure 2 (Left) The gadget for a double-weighted edge $e = \{u, v\}$ with $(\mathrm{w}_1(e), \mathrm{w}_2(e)) = (3, 5)$. There are $\mathrm{w}_1(e)$ $(= 3)$ grids connected to $\{u, v\}$ and each grid is of size $(\mathrm{w}_2(e) - \mathrm{w}_1(e) + 1) \times (\mathrm{w}_2(e) - \mathrm{w}_1(e) + 1)$ $(= 3 \times 3)$. (Center & Right) The intersection of the gadget and the spanning tree T' of G' obtained from a spanning tree T of G for the cases $e \in E(T)$ and $e \notin E(T)$: in the former case, the edges $\{v_1, v\}, \{v_2, v\}, \{v_3, v\}$ and some edges in Γ_1 contribute to the congestion of the u–v path in T'; in the latter, only the edges $\{v_1, v\}, \{v_2, v\}, \{v_3, v\}$ do.

variables and each variable appears exactly twice positively and exactly twice negatively. It is known that $(3, \mathrm{B2})$-SAT is NP-complete [7], even if the formula contains only monotone clauses [22].

▶ **Theorem 11.** Spanning Tree Congestion *is NP-complete on graphs of maximum degree at most* 8.

Construction. Let (X, C) be an instance of $(3, \mathrm{B2})$-SAT with $X = \{x_1, \ldots, x_n\}$ and $C = \{c_1, \ldots, c_m\}$. We assume that $m \geq 3$ (otherwise the problem becomes trivial). Set $k = 2m+3$ (≥ 9). From (X, C), we construct a double edge-weighted graph $G = (V, E; \mathrm{w}_1, \mathrm{w}_2)$ as follows (see Figure 3).

- For $i \in [n]$, take a cycle $(x_i, y_i, \bar{x}_i, z_i)$ of four new vertices.
  - Set $\mathrm{w}(\{x_i, y_i\}) = \mathrm{w}(\{\bar{x}_i, y_i\}) = (4, k-3)$ and $\mathrm{w}(\{x_i, z_i\}) = \mathrm{w}(\{\bar{x}_i, z_i\}) = (1, 1)$.
- For $i \in [n-1]$, add the edge $\{z_i, z_{i+1}\}$, thus forming the path $(z_1, \ldots, z_n)$.
  - Set $\mathrm{w}(\{z_i, z_{i+1}\}) = (3, 3)$.
- For $j \in [m]$, take a new vertex c_j.
- For $i \in [n]$ and $j \in [m]$, add the edge $\{x_i, c_j\}$ (resp. $\{\bar{x}_i, c_j\}$) if $x_i \in c_j$ (resp. $\bar{x}_i \in c_j$).
  - Set $\mathrm{w}(\{x_i, c_j\}) = (1, k-2)$ (resp. $\mathrm{w}(\{\bar{x}_i, c_j\}) = (1, k-2)$) if the edge exists.

To prove Theorem 11, it suffices to prove the following two claims.

1. In polynomial time, we can construct an unweighted graph G' such that
   - $\mathrm{stc}(G) \leq k$ if and only if $\mathrm{stc}(G') \leq k$;
   - the maximum degree of G' is at most 8.
2. $\mathrm{stc}(G) \leq k$ if and only if (X, C) is a yes-instance of $(3, \mathrm{B2})$-SAT.

Lemma 12 shows the first claim and Lemma 13 shows the second one.

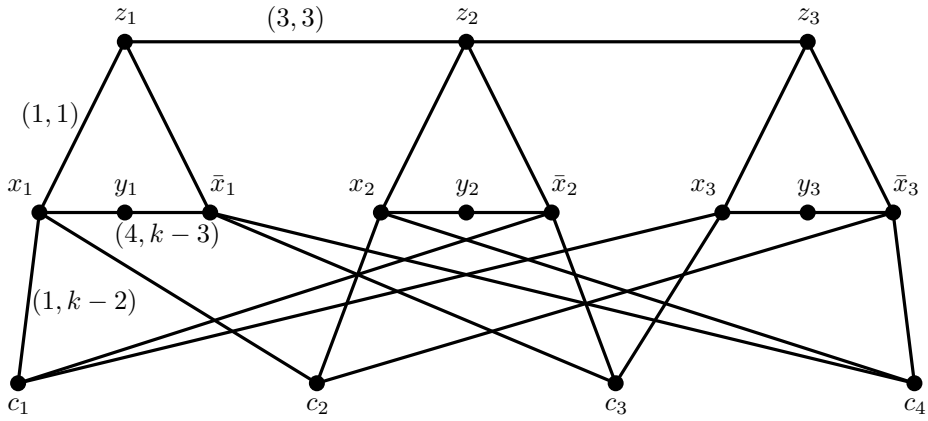

Figure 3 The construction of G, where $c_1 = \{x_1, \bar{x}_2, x_3\}$, $c_2 = \{x_1, x_2, \bar{x}_3\}$, $c_3 = \{\bar{x}_1, \bar{x}_2, x_3\}$, and $c_4 = \{\bar{x}_1, x_2, \bar{x}_3\}$.

▶ **Lemma 12** (⋆). *In time polynomial in $m + n$, one can construct an unweighted graph $G' = (V', E')$ from the double edge-weighted graph $G = (V, E; \mathrm{w}_1, \mathrm{w}_2)$ with maximum degree at most 8 such that $\mathrm{stc}(G) \leq k$ if and only if $\mathrm{stc}(G') \leq k$.*

▶ **Lemma 13** (⋆). *$\mathrm{stc}(G) \leq k$ if and only if (X, C) is a yes-instance of $(3, \mathrm{B2})$-SAT.*

4 Algorithms for Bounded Treewidth

In this section we take a second look at the complexity of Spanning Tree Congestion parameterized by treewidth, a problem shown to be W[1]-hard in Corollary 2. One way to deal with this hardness is to consider additional parameters, so we begin by presenting in Section 4.1 an FPT algorithm parameterized by treewidth plus the desired congestion k. Our algorithm follows the standard technique of performing dynamic programming over a tree decomposition, though with a few necessary tweaks (informally, we have to guess the general structure of the spanning tree, including parts of the graph that will appear "in the future").

We note here that the fact that Spanning Tree Congestion is FPT for this parameterization was already shown in [9], where it was proved that if k is a parameter, then the problem is MSO_2-expressible, hence solvable via Courcelle's theorem. Nevertheless, we are still motivated to provide an explicit algorithm for the parameter "treewidth plus k" for several reasons. First, using Courcelle's theorem does not provide any usable upper bound on the running time, while we show our algorithm to run in $(k+w)^{\mathcal{O}(w)} n^{\mathcal{O}(1)}$, where w is the treewidth; this implies, for instance, that Spanning Tree Congestion is in XP parameterized by treewidth alone (as $\mathrm{stc}(G) \leq n^2$ for all G), a fact that cannot be inferred using Courcelle's theorem.

Second, and more importantly, having an explicit algorithm at hand, we are able to obtain an answer to the following natural question: given that solving Spanning Tree Congestion is hard parameterized by treewidth, is there an FPT algorithm that closely approximates the optimal congestion? By applying a technique of Lampis [54] which modifies exact DP algorithms to obtain approximate ones, we get an FPT approximation scheme, which runs in time $(\frac{w}{\varepsilon})^{\mathcal{O}(w)} n^{\mathcal{O}(1)}$ (that is, FPT in $w + \frac{1}{\varepsilon}$) and returns a $(1+\varepsilon)$-approximate solution, for any desired $\varepsilon > 0$. This result naturally complements the problem's hardness for treewidth and is presented in Section 4.2. We complete that section by presenting a simple Win/Win argument which extends our algorithm to an algorithm that is FPT parameterized

by clique-width plus k; this is based on a result of Gurski and Wanke [42] stating that graphs of bounded clique-width with no large complete bipartite subgraphs actually have bounded treewidth.

4.1 FPT Algorithm Parameterized by Treewidth and Congestion

In this section, we prove the following theorem:

▶ **Theorem 14.** *Let G be a graph with treewidth w, and let $k > 0$. There is an algorithm that finds a spanning tree of G with congestion k, if it exists, and runs in time $(w+k)^{\mathcal{O}(w)} n^{\mathcal{O}(1)}$.*

Before we proceed with the formal proof of the theorem, we will give an intuition of the algorithm. Following the usual structure for bounded-treewidth graphs, we will use dynamic programming to compute solutions to the subgraphs corresponding to subtrees of the tree decomposition $\mathcal{T}$. For each such subgraph, we consider different possibilities for the solution to manifest on the root bag of the subtree; these different possibilities, which we call *states*, represent equivalent classes of solutions that are "compatible" with respect to the rest of the graph, and thus it is sufficient to compute a feasible solution for each such class.

To obtain the algorithm, we simply need to specify the states, as well as how to recursively obtain feasible solutions for each. Our states are composed of a tree, which we call the *skeleton*, as well as values of congestion for each of its edges. The skeleton of a solution T for a given bag X_t intuitively represents how T connects the vertices of X_t, and it can be obtained by contracting the vertices in $V \setminus X_t$ that have degree 1 or 2 in T. Thus, the skeleton is a tree containing the vertices in X_t, plus at most $|X_t|$ vertices with degree at least 3. The congestion values correspond to the congestion induced on an edge by edges in $\mathcal{T}_t$, the subtree of $\mathcal{T}$ rooted at t.

There is one further particularity that we must consider: when constructing a skeleton from a solution, some of the resulting edges may represent paths in the subtree rooted at t, while others represent paths outside of this subtree. Thus, we label each edge of the skeleton with one of three types: a *present* edge is simply an edge between two vertices in X_t; a *past* edge represents a path in $\mathcal{T}_t$; and a *future* edge represents a path outside $\mathcal{T}_t$, that is, one that is not (yet) in the solution, but that must be added to make it compatible with the state. We similarly label the vertices with the type *present* if they are in X_t, *past* if they are not in X_t but in $\mathcal{T}_t$, and *future* otherwise.

Throughout the section, we assume familiarity with the definition and usual notation for treewidth (see e.g. [21, Chapter 7]). When referring to subgraphs of G, we often refer to subgraphs induced by subtrees of $\mathcal{T}$: we denote by $\mathcal{T}_t$ the subtree of $\mathcal{T}$ rooted at t, and by $G[S]$ the subgraph of G induced by the vertices contained in bags of S, i.e. the subgraph $G\big[\bigcup_{t\in S} X_t\big]$; for convenience of notation, we write $G[\mathcal{T}_t]$ instead of $G[V(\mathcal{T}_t)]$.

The algorithm starts by computing a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ for G with width at most $2w+1$ and at most $\mathcal{O}(nw)$ bags, which can be computed in time $2^{\mathcal{O}(w)} \cdot n$ [50].

We now formalize the definition of skeleton:

▶ **Definition 15.** *Given a graph X, a* skeleton (S, ℓ) *is a tree S together with a labeling $\ell\colon E(S) \cup V(S) \to \{-1, 0, 1\}$, such that:*

1. *$V(X) \subseteq V(S)$;*
2. *for any $v \in V(S) \setminus V(X)$, $\deg_S(v) \geq 3$;*
3. *for any $v \in V(S)$, $\ell(v) = 0$ if and only if $v \in X_t$;*
4. *for any $uv \in E(S)$, $\ell(u) = \ell(uv)$ or $\ell(u) = 0$ (similarly for v).*
5. *for any $uv \in E(S)$, if $\ell(uv) = 0$ then $uv \in E(X)$;*

If (S, ℓ) satisfies every property except Property 2, we call it a quasi-skeleton*.*

We denote by $\ell^{-1}(i)$, $i \in \{-1, 0, 1\}$ the graph obtained from the edges with label i and their endpoints (which can have label i or 0).

The labeling ℓ corresponds to the edge and vertex types and is encoded as -1 for past, 0 for present and 1 for future.

We define a dynamic programming table D with entries for every node $t \in \mathcal{T}$ and every triple (S, ℓ, c), where (S, ℓ) is a skeleton for $G[X_t]$ and $c\colon E(S) \to [0, k]$ is a function assigning a congestion to each edge. Informally, $D[t, (S, \ell, c)]$ represents a forest $F \subseteq G[\mathcal{T}_t]$ such that $F \cup \ell^{-1}(1)$ is a solution of congestion at most k for $G[\mathcal{T}_t]$, and the congestion $c(uv)$ on an edge of the skeleton corresponds to the maximum congestion of the u-v-path in F. We formalize the desired properties by the definition below:

▶ **Definition 16.** *We say that a forest $F \subseteq G[\mathcal{T}_t]$ is a* consistent solution *for $(t, (S, \ell, c))$ if:*

1. *F is a forest of $G[\mathcal{T}_t]$ on the same vertex set;*
2. *For $uv \in E(S)$, if $\ell(uv) = 0$, then $uv \in F$;*
3. *For $uv \in E(S)$, if $\ell(uv) = -1$, then F contains a u-v-path F_{uv} with edges from $G[\mathcal{T}_t] - E(X_t)$;*
4. *$T' := F \cup \ell^{-1}(1)$ is a tree;*
5. *For any edge $e \in E(S)$ with $\ell(e) \in \{0, 1\}$, the congestion in T' induced by $G[\mathcal{T}_t] - E(X_t)$ on e is $c(e)$;*
6. *For any edge $e \in E(S)$ with $\ell(e) = -1$, the congestion in T' induced by $G[\mathcal{T}_t] - E(X_t)$ on every edge of F_e is at most $c(e)$;*
7. *For any $e \in E(T')$, the congestion in T' induced by $G[\mathcal{T}_t] - E(X_t)$ on e is at most k.*

Before we describe the recursive rules of our algorithm, we show an operation called *simplification* which takes a quasi-skeleton and transforms it into a skeleton, which is necessary to keep the number of skeletons small enough.

▶ **Lemma 17 ($\star$).** *Let X be a graph, (S, ℓ) be a quasi-skeleton for that graph, and $c\colon E(S) \to [k]$. We can obtain a skeleton (S', ℓ') for X by starting with $(S', \ell') = (S, \ell)$, $c' = c$ and iteratively contracting a vertex $v \in V(S') \setminus V(X)$ with degree less than 3 as follows:*

- *if it has degree 1, we simply remove v and its incident edge from S', ℓ', c;*
- *if it has degree 2 and label $\lambda = \ell(v)$, we replace v and its incident edges by an edge uw between its neighbors u and w, and set $\ell'(uw) = \lambda$, and adjust $c'(uw) = \max\{c(uv), c(vw)\}$.*

We name this process simplification*.*

We will now see how to construct the entries $D[t, (S, \ell, c)]$ recursively:

- **Leaf t:** the only allowed skeleton is $S = (\varnothing, \varnothing)$, and $D[t, (S, \varnothing, \varnothing)] = (\varnothing, \varnothing)$.
- **Forget node t:** let t' be the child of t and let $\{v\} = X_{t'} \setminus X_t$. For any solution $D[t', (S', \ell', c')]$, if v has an incident edge with label 1, the solution is invalid; otherwise, we set $D[t, (S, \ell, c)] = D[t', (S', \ell', c')]$ for (S, ℓ, c) obtained as follows:
  1. we start with $(S, \ell, c) = (S', \ell', c')$;
  2. we add the congestion corresponding to the forgotten vertex v: for any edge $uv \in E(G)$, $u \in X_t$, increment $c(e)$ on each edge e on the u-v-path in S; if $c(e) > k$ for any edge, the solution is invalid and the process stops;
  3. then we mark v as a past vertex ($\ell(v) = -1$) and apply simplification.
- **Introduce node t:** let t' be the child of t and let $\{v\} = X_t \setminus X_{t'}$. For any (S, ℓ, c), we obtain (S', ℓ', c') from (S, ℓ, c) by setting $\ell'(v) = 1$ and applying simplification. If $D[t', (S', \ell', c')]$ exists, we construct $D[t, (S, \ell, c)]$ from $D[t', (S', \ell', c')]$ by adding the vertex v and every of its incident edges $e \in E(S)$ with label $\ell(e) = 0$.

- **Join node t:** given solutions $D[t_1, (S, \ell_1, c_1)]$, $D[t_2, (S, \ell_2, c_2)]$, a valid solution can be obtained if $c_1(e) + c_2(e) \leq k$, if $\ell_1(x) = -1$ implies $\ell_2(x) = 1$ and vice-versa ($x \in V(S) \cup E(S)$), and $\ell_1(e) = 0$ if and only if $\ell_2(e) = 0$.
  We define $\ell(x) = \min\{\ell_1(x), \ell_2(x)\}$, $x \in V(S) \cup E(S)$ and $c(e) = c_1(e) + c_2(e)$, $e \in E(S)$, and set $D[t, (S, \ell, c)] = D[t_1, (S, \ell_1, c_1)] \cup D[t_2, (S, \ell_2, c_2)]$.

To obtain a solution to the problem, we simply compute $T = D[r, (\varnothing, \varnothing, \varnothing)]$. If T is a consistent solution, then it is a forest containing the vertices $V(G)$ by Property 1, it is a tree by Property 4, and has congestion at most k by Property 7. We therefore conclude that T is a spanning tree of G with congestion at most k, as desired.

Lemmas 18–20 show that T is indeed a consistent solution, that if there is a feasible solution then our algorithm can always find it, and that the algorithm runs in the stated running time.

▶ **Lemma 18** (⋆). *For any t and (S, ℓ, c), if $D[t, (S, \ell, c)]$ exists, then it is a consistent solution.*

▶ **Lemma 19** (⋆). *Let T be a tree with congestion at most k. Then for any t, there is (S_t, ℓ_t, c_t) such that $D[t, (S, \ell, c)]$ exists.*

▶ **Lemma 20** (⋆). *The running time of the algorithm is $(w + k)^{\mathcal{O}(w)} n^{\mathcal{O}(1)}$.*

4.2 FPT Approximation and Clique-width

In this section we leverage the algorithm of Theorem 14 to obtain two further results. On the one hand, we observe that the FPT algorithm of Theorem 14 relies on two parameters (treewidth and the optimal congestion), but it is likely impossible to improve this to an exact algorithm parameterized by treewidth alone (indeed, we saw that Theorem 1 implies that the problem is hard even for parameters much more restricted than treewidth). We are therefore motivated to consider the question of approximation and present an FPT approximation scheme parameterized by treewidth alone. Our algorithm is based on a combination of the algorithm of Theorem 14 together with a technique introduced in [54] (later also used among others in [3, 4, 19, 55]) which allows us to perform dynamic programming while maintaining approximate values for the congestion. The end result is an FPT approximation scheme, which for any $\varepsilon > 0$ is able to return a $(1 + \varepsilon)$-approximate solution in time $(w/\varepsilon)^{\mathcal{O}(w)} n^{\mathcal{O}(1)}$, that is, FPT in $w + \frac{1}{\varepsilon}$.

Having established this, we obtain a second extension of Theorem 14, to the more general parameter clique-width. Here, we rely on a Win/Win argument: suppose we are given an input graph G of clique-width w and are asked if a tree of congestion k can be found; if G contains a large bi-clique (in terms of k), then we show that this can be found and we can immediately say No; otherwise, by a well-known result of Gurski and Wanke [42], we infer that the graph actually has low treewidth, so we can apply Theorem 14.

▶ **Theorem 21** (⋆). *There is an algorithm which, for all $\varepsilon > 0$, when given as input a graph G of treewidth w, returns a spanning tree of congestion at most $(1 + \varepsilon)\mathrm{stc}(G)$ in time $(\frac{w}{\varepsilon})^{\mathcal{O}(w)} n^{\mathcal{O}(1)}$.*

▶ **Theorem 22** (⋆). *There is an algorithm which, when given as input a graph G, an integer k, and a clique-width expression for G with w labels, correctly decides if $\mathrm{stc}(G) \leq k$ in time $(wk)^{\mathcal{O}(wk)} n^{\mathcal{O}(1)}$.*

5 FPT Algorithms

Given the hardness results of Section 3, we are motivated to search for structural parameters that render Spanning Tree Congestion fixed-parameter tractable. In this section we present various such results, starting with the parameter distance to clique in Section 5.1, then vertex integrity in Section 5.2, and finally feedback edge number in Section 5.3.

5.1 Distance to Clique

Corollary 3 implies that Spanning Tree Congestion remains W[1]-hard even on very structured dense instances. In this subsection we search for parameters that render the problem tractable on dense instances, and present an FPT algorithm when parameterized by the distance to clique of the input graph, arguably one of the most restrictive such parameters. Interestingly, the running time of our algorithm is dictated by the "easy" case, where the clique modulator is large with respect to the size of the graph. We remark that a modulator to clique of a graph G is a vertex cover in the complement $\bar{G}$ of G, and thus the minimum modulator can be computed by employing any FPT algorithm for Vertex Cover (e.g. [18, 44]) on $\bar{G}$.

▶ **Theorem 23** (⋆). *Given $G = (V, E)$ and $S \subseteq V$ with $G - S$ being a clique, there is an algorithm that returns a spanning tree of G of congestion $\mathrm{stc}(G)$ in time $2^{\mathcal{O}(k^3)}n^{\mathcal{O}(1)}$, where $n = |V|$ and $k = |S|$.*

5.2 Vertex integrity

Here we prove that the parameterization by vertex integrity renders Spanning Tree Congestion FPT.

▶ **Theorem 24** (⋆). Spanning Tree Congestion *is fixed-parameter tractable parameterized by vertex integrity.*

5.3 Feedback Edge Number

Notice that the algorithm of Theorem 14 already implies that Spanning Tree Congestion is FPT parameterized by the feedback edge number fes of the input graph: any instance with $k \geq \mathrm{fes} + 1$ is trivially yes, thus one can easily obtain an $\mathrm{fes}^{\mathcal{O}(\mathrm{fes})}n^{\mathcal{O}(1)}$ algorithm. A natural question is whether the slightly super-exponential parametric dependence can be overcome, and we answer this affirmatively by providing such an algorithm. In fact, more strongly, we present a kernelization algorithm that results in a graph with only $\mathcal{O}(\mathrm{fes})$ vertices and edges, thus allowing us to exhaustively guess the spanning tree of minimum congestion. To do so, we notice that we can safely delete vertices of degree 1, resulting in a graph with only $\mathcal{O}(\mathrm{fes})$ vertices of degree larger than 2. Next, we can contract most of the remaining edges, thereby leaving only $\mathcal{O}(\mathrm{fes})$ of them, thus allowing us to guess exhaustively which edges belong to an optimal solution.

▶ **Theorem 25.** Spanning Tree Congestion *admits a kernel with $\mathcal{O}(\mathrm{fes})$ vertices and edges, where* fes *denotes the feedback edge number of the input graph.*

Proof. Let G_0 denote the input connected graph. We start by exhaustively deleting vertices of degree 1 in G_0; it is easy to see that this is safe, as any such vertex is connected to any spanning tree via the single edge it is incident with, which is of congestion 1. Let G denote

the resulting connected graph, where fes denotes its feedback edge number; notice that the deletion of vertices cannot increase the feedback edge number of a graph. Let $V^{=2}(G)$ and $V^{\geq 3}(G)$ denote the sets of vertices of G of degree exactly 2 and at least 3 respectively; notice that they induce a partition on $V(G)$, since all vertices of G are of degree at least 2. Bentert, Dittmann, Kellerhals, Nichterlein, and Niedermeier [5, Lemma 2] have proved that $|V^{\geq 3}(G)| < 2\text{fes}$ and $\sum_{v \in V^{\geq 3}(G)} \deg_G(v) = 2(|V^{\geq 3}(G)| + \text{fes} - 1) < 6\text{fes}$.[5] We next define the following reduction rule.

Rule ($\diamond$). Let G be a graph with (not necessarily distinct) vertices $u, v \in V^{\geq 3}(G)$ such that there is a u–v path P in G whose internal vertices all belong to $V^{=2}(G)$. Let $L_P \geq 1$ be equal to the number of internal vertices of P. Then,

- **(i)** if $u = v$ and $L_P > 2$, contract edges in P such that only 2 internal vertices are left,
- **(ii)** if $u \neq v$, $\{u, v\} \in E(G)$, and $L_P > 1$, contract edges in P such that only 1 internal vertex is left,
- **(iii)** if $u \neq v$ and $\{u, v\} \notin E(G)$, delete all internal vertices of P and add the edge $\{u, v\}$.

Notice that Rule ($\diamond$) can be applied at most n times, since each of its applications reduces the number of vertices of the graph. Let G' denote the connected graph obtained after exhaustively doing so. Notice that G can be obtained from G' by only subdividing edges; edge subdivision does not change the spanning tree congestion of unweighted graphs [9, Lemma 7.10], thus it holds that $\text{stc}(G) = \text{stc}(G')$. Furthermore, it is easy to see that $V^{\geq 3}(G') = V^{\geq 3}(G)$, and in fact any such vertex has the same degree in both G and G'. Consequently, it follows that $\sum_{v \in V^{\geq 3}(G')} \deg_{G'}(v) = \sum_{v \in V^{\geq 3}(G)} \deg_G(v) < 6\text{fes}$.

Let E_1 and E_2 define a partition on the edges of G', where $E_1 = \{e \in E(G') \mid e \cap V^{\geq 3}(G') \neq \varnothing\}$ denotes the set of edges in $E(G')$ with at least one endpoint belonging to $V^{\geq 3}(G')$, and E_2 the rest, whose endpoints both belong to $V^{=2}(G')$. Due to the previous discussion, it holds that $|E_1| < 6\text{fes}$. As for E_2, since Rule ($\diamond$) has been applied exhaustively, any such edge can only be due to Case (i) of Rule ($\diamond$), thus $|E_2| \leq |E_1|/2$ holds. In total, it follows that $|E(G')| < 9\text{fes}$. Furthermore, since G' is connected, it follows that $|V(G')| \leq |E(G')| + 1$. ◀

6 Conclusion

In this paper we have presented a number of new results on the parameterized complexity of Spanning Tree Congestion, painting an almost-complete picture regarding its tractability under the most standard parameterizations. As a direction for future work, it would be interesting to consider the problem's tractability parameterized by the neighborhood diversity, the treewidth plus the maximum degree, as well as whether an FPT-AS parameterized by clique-width exists. Furthermore, although the problem is FPT by vertex cover due to Theorem 24, the algorithm is based on an ILP, and a simpler (and faster) combinatorial algorithm might be possible under this parameterization; along these lines, it would be interesting to determine whether the problem admits a polynomial kernel in this case. We additionally mention that it is unknown whether the problem remains NP-hard on cographs or when $k = 4$.

[5] We mention in passing a slight mistake in their proof, where they claim that $\sum_{v \in V^{\geq 3}(G)} \deg_G(v) = 3|V^{\geq 3}(G)|$ instead.

References

1 Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. *SIAM J. Comput.*, 48(2):227–248, 2019. doi:10.1137/17M1115575.

2 Pankaj K. Agarwal. Ray shooting and other applications of spanning trees with low stabbing number. *SIAM J. Comput.*, 21(3):540–570, 1992. doi:10.1137/0221035.

3 Eric Angel, Evripidis Bampis, Bruno Escoffier, and Michael Lampis. Parameterized power vertex cover. *Discret. Math. Theor. Comput. Sci.*, 20(2), 2018. doi:10.23638/DMTCS-20-2-10.

4 Rémy Belmonte, Michael Lampis, and Valia Mitsou. Parameterized (approximate) defective coloring. *SIAM J. Discret. Math.*, 34(2):1084–1106, 2020. doi:10.1137/18M1223666.

5 Matthias Bentert, Alexander Dittmann, Leon Kellerhals, André Nichterlein, and Rolf Niedermeier. An adaptive version of brandes' algorithm for betweenness centrality. *J. Graph Algorithms Appl.*, 24(3):483–522, 2020. doi:10.7155/JGAA.00543.

6 Kristóf Bérczi, Tamás Király, Yusuke Kobayashi, Yutaro Yamaguchi, and Yu Yokoi. Finding spanning trees with perfect matchings. *Discret. Appl. Math.*, 371:137–147, 2025. doi:10.1016/j.dam.2025.04.001.

7 Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electron. Colloquium Comput. Complex.*, TR03-049, 2003. URL: https://eccc.weizmann.ac.il/eccc-reports/2003/TR03-049/.

8 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

9 Hans L. Bodlaender, Fedor V. Fomin, Petr A. Golovach, Yota Otachi, and Erik Jan van Leeuwen. Parameterized complexity of the spanning tree congestion problem. *Algorithmica*, 64(1):85–111, 2012. doi:10.1007/S00453-011-9565-7.

10 Hans L. Bodlaender, Kyohei Kozawa, Takayoshi Matsushima, and Yota Otachi. Spanning tree congestion of k-outerplanar graphs. *Discret. Math.*, 311(12):1040–1045, 2011. doi:10.1016/J.DISC.2011.03.002.

11 Narek Bojikian, Alexander Firbas, Robert Ganian, Hung P. Hoang, and Krisztina Szilágyi. Fine-grained complexity of computing degree-constrained spanning trees. *CoRR*, abs/2503.15226, 2025. doi:10.48550/arXiv.2503.15226.

12 Paul S. Bonsma and Florian Zickfeld. A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graphs. *SIAM J. Discret. Math.*, 25(4):1652–1666, 2011. doi:10.1137/100801251.

13 Glencora Borradaile, Erin Wolf Chambers, David Eppstein, William Maxwell, and Amir Nayyeri. Low-stretch spanning trees of graphs with bounded width. In *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020*, volume 162 of *LIPIcs*, pages 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.SWAT.2020.15.

14 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*, volume 13453 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2022. doi:10.1007/978-3-031-15914-5_8.

15 Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM J. Discret. Math.*, 8(3):359–387, 1995. doi:10.1137/S0895480192237403.

16 Alberto Castejón and Mikhail I. Ostrovskii. Minimum congestion spanning trees of grids and discrete toruses. *Discuss. Math. Graph Theory*, 29(3):511–519, 2009. doi:10.7151/DMGT.1461.

17 L. Sunil Chandran, Yun Kuen Cheung, and Davis Issac. Spanning tree congestion and computation of generalized Győri-Lovász partition. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 32:1–32:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ICALP.2018.32.

18 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/J.TCS.2010.06.026.

19 Huairui Chu and Bingkai Lin. FPT approximation using treewidth: Capacitated vertex cover, target set selection and vector dominating set. In *34th International Symposium on Algorithms and Computation, ISAAC 2023*, volume 283 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ISAAC.2023.19.

20 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

21 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

22 Andreas Darmann and Janosch Döcker. On simplified NP-complete variants of Monotone 3-Sat. *Discret. Appl. Math.*, 292:45–58, 2021. doi:10.1016/J.DAM.2020.12.010.

23 Louis DeBiasio and Allan Lo. Spanning trees with few branch vertices. *SIAM J. Discret. Math.*, 33(3):1503–1520, 2019. doi:10.1137/17M1152759.

24 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.

25 Martin Doucha and Jan Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012*, volume 7464 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2012. doi:10.1007/978-3-642-32589-2_32.

26 Feodor F. Dragan, Fedor V. Fomin, and Petr A. Golovach. Spanners in sparse graphs. *J. Comput. Syst. Sci.*, 77(6):1108–1119, 2011. doi:10.1016/J.JCSS.2010.10.002.

27 Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM J. Comput.*, 38(2):608–628, 2008. doi:10.1137/050641661.

28 Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. *SIAM J. Comput.*, 38(5):1761–1781, 2008. doi:10.1137/060666202.

29 Sándor P. Fekete and Jana Kremer. Tree spanners in planar graphs. *Discret. Appl. Math.*, 108(1-2):85–103, 2001. doi:10.1016/S0166-218X(00)00226-2.

30 Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, Frances A. Rosamond, and Saket Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory Comput. Syst.*, 45(4):822–848, 2009. doi:10.1007/S00224-009-9167-9.

31 Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width is NP-complete. *SIAM J. Discret. Math.*, 23(2):909–939, 2009. doi:10.1137/070687256.

32 Fedor V. Fomin, Petr A. Golovach, and Erik Jan van Leeuwen. Spanners of bounded degree graphs. *Inf. Process. Lett.*, 111(3):142–144, 2011. doi:10.1016/J.IPL.2010.10.021.

33 Jakub Gajarský, Michael Lampis, Kazuhisa Makino, Valia Mitsou, and Sebastian Ordyniak. Parameterized algorithms for parity games. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015*, volume 9235 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2015. doi:10.1007/978-3-662-48054-0_28.

34 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013. doi:10.1007/978-3-319-03898-8_15.

35 Robert Ganian. Improving vertex cover as a graph parameter. *Discret. Math. Theor. Comput. Sci.*, 17(2):77–100, 2015. doi:10.46298/DMTCS.2136.

36 Robert Ganian, Petr Hlinený, Jaroslav Nesetril, Jan Obdrzálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing height of dense graphs. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:7)2019.

37 Robert Ganian, Petr Hlinený, Jaroslav Nesetril, Jan Obdrzálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When trees grow low: Shrubs and fast MSO_1. In *Mathematical*

Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012. Proceedings, volume 7464 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 2012. doi:10.1007/978-3-642-32589-2_38.

38 Robert Ganian and Viktoriia Korchemna. The complexity of bayesian network learning: Revisiting the superstructure. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 430–442, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/040a99f23e8960763e680041c601acab-Abstract.html.

39 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

40 Luisa Gargano, Pavol Hell, Ladislav Stacho, and Ugo Vaccaro. Spanning trees with bounded number of branch vertices. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 355–365. Springer, 2002. doi:10.1007/3-540-45465-9_31.

41 Luisa Gargano and Adele A. Rescigno. An FPT algorithm for spanning trees with few branch vertices parameterized by modular-width. In *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023*, volume 272 of *LIPIcs*, pages 50:1–50:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.50.

42 Frank Gurski and Egon Wanke. The tree-width of clique-width bounded graphs without $K_{n,n}$. In *Graph-Theoretic Concepts in Computer Science, 26th International Workshop, WG 2000*, volume 1928 of *Lecture Notes in Computer Science*, pages 196–205. Springer, 2000. doi:10.1007/3-540-40064-8_19.

43 Magnús M. Halldórsson, Guy Kortsarz, Pradipta Mitra, and Tigran Tonoyan. Network design under general wireless interference. *Algorithmica*, 83(11):3469–3490, 2021. doi:10.1007/S00453-021-00866-Z.

44 David G. Harris and N. S. Narayanaswamy. A faster algorithm for vertex cover parameterized by solution size. In *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024*, volume 289 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.STACS.2024.40.

45 Stephen W. Hruska. On tree congestion of graphs. *Discret. Math.*, 308(10):1801–1809, 2008. doi:10.1016/J.DISC.2007.04.030.

46 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/j.jcss.2012.04.004.

47 Daniel J. Kleitman and Douglas B. West. Spanning trees with many leaves. *SIAM J. Discret. Math.*, 4(1):99–106, 1991. doi:10.1137/0404010.

48 Petr Kolman. Approximating spanning tree congestion on graphs with polylog degree. In *Combinatorial Algorithms - 35th International Workshop, IWOCA 2024*, volume 14764 of *Lecture Notes in Computer Science*, pages 497–508. Springer, 2024. doi:10.1007/978-3-031-63021-7_38.

49 Petr Kolman. Approximation of spanning tree congestion using hereditary bisection. In *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025*, volume 327 of *LIPIcs*, pages 63:1–63:6. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.STACS.2025.63.

50 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.

51 Kyohei Kozawa and Yota Otachi. Spanning tree congestion of rook's graphs. *Discuss. Math. Graph Theory*, 31(4):753–761, 2011. doi:10.7151/DMGT.1577.

52 Kyohei Kozawa, Yota Otachi, and Koichi Yamazaki. On spanning tree congestion of graphs. *Discret. Math.*, 309(13):4215–4224, 2009. doi:10.1016/J.DISC.2008.12.021.

53 Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012. doi:10.1007/S00453-011-9554-X.

54 Michael Lampis. Parameterized approximation schemes using graph widths. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 775–786. Springer, 2014. doi:10.1007/978-3-662-43948-7_64.

55 Michael Lampis. Minimum stable cut and treewidth. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 92:1–92:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.92.

56 Hiu-Fai Law. Spanning tree congestion of the hypercube. *Discret. Math.*, 309(23-24):6644–6648, 2009. doi:10.1016/J.DISC.2009.07.007.

57 Christian Löwenstein. *In the complement of a dominating set.* PhD thesis, Technische Universität Ilmenau, Germany, August 2010. URL: https://www.db-thueringen.de/receive/dbt_mods_00016280.

58 Christian Löwenstein, Dieter Rautenbach, and Friedrich Regen. On spanning tree congestion. *Discret. Math.*, 309(13):4653–4655, 2009. doi:10.1016/J.DISC.2009.01.012.

59 Huong Luu and Marek Chrobak. Better hardness results for the minimum spanning tree congestion problem. *Algorithmica*, 87(1):148–165, 2025. doi:10.1007/s00453-024-01278-5.

60 Martin Nägele and Rico Zenklusen. A new dynamic programming approach for spanning trees with chain constraints and beyond. *Mathematics of Operations Research*, 49(4):2078–2108, 2024. doi:10.1287/moor.2023.0012.

61 Yoshio Okamoto, Yota Otachi, Ryuhei Uehara, and Takeaki Uno. Hardness results and an exact exponential algorithm for the spanning tree congestion problem. *J. Graph Algorithms Appl.*, 15(6):727–751, 2011. doi:10.7155/JGAA.00246.

62 M. I. Ostrovskii. Minimal congestion trees. *Discret. Math.*, 285(1-3):219–226, 2004. doi:10.1016/J.DISC.2004.02.009.

63 Mikhail I. Ostrovskii. Minimum congestion spanning trees in planar graphs. *Discret. Math.*, 310(6-7):1204–1209, 2010. doi:10.1016/J.DISC.2009.11.016.

64 Yota Otachi. A survey on spanning tree congestion. In *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 165–172. Springer, 2020. doi:10.1007/978-3-030-42071-0_12.

65 Shai Simonson. A variation on the min cut linear arrangement problem. *Math. Syst. Theory*, 20(4):235–252, 1987. doi:10.1007/BF01692067.

66 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *J. ACM*, 62(1):1:1–1:19, 2015. doi:10.1145/2629366.