Polynomial Bounds for the Grid-Minor Theorem

Chandra Chekuri[*]
Dept. of Computer Science
University of Illinois
Urbana, IL 61801
chekuri@illinois.edu

Julia Chuzhoy[†]
Toyota Technological Institute at Chicago
6045 S. Kenwood Ave
Chicago, IL 60637
cjulia@ttic.edu

ABSTRACT

One of the key results in Robertson and Seymour's seminal work on graph minors is the Grid-Minor Theorem (also called the Excluded Grid Theorem). The theorem states that for every fixed-size grid H, every graph whose treewidth is large enough, contains H as a minor. This theorem has found many applications in graph theory and algorithms. Let $f(k)$ denote the largest value, such that every graph of treewidth k contains a grid minor of size $f(k) \times f(k)$. The best current quantitative bound, due to recent work of Kawarabayashi and Kobayashi [15], and Leaf and Seymour [18], shows that $f(k) = \Omega(\sqrt{\log k / \log\log k})$. In contrast, the best known upper bound implies that $f(k) = O(\sqrt{k/\log k})$ [22]. In this paper we obtain the first polynomial relationship between treewidth and grid-minor size by showing that $f(k) = \Omega(k^{\delta})$ for some fixed constant $\delta > 0$, and describe an algorithm, whose running time is polynomial in $|V(G)|$ and k, that finds a model of such a grid-minor in G.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms

General Terms

Algorithms, Theory

Keywords

Treewidth, grid minor theorem

1. INTRODUCTION

The seminal work of Roberston and Seymour on graph minors makes essential use of the notions of tree decompositions and treewidth. A key structural result in their work is the Grid-Minor theorem (also called the Excluded Grid theorem), which states that for every fixed-size grid H, every graph whose treewidth is large enough, contains H as a minor. This theorem has found many applications in graph theory and algorithms. Let $f(k)$ denote the largest value, such that every graph of treewidth k contains a grid minor of size $f(k) \times f(k)$. The quantitative estimate for f given in the original proof of Robertson and Seymour [23] was substantially improved by Robertson, Seymour and Thomas [22] who showed that $f(k) = \Omega(\log^{1/5} k)$; see [13, 12] for a simpler proof with a slightly weaker bound. There have been recent improvements by Kawarabayashi and Kobayashi [15], and by Leaf and Seymour [18], giving the best current bound of $f(k) = \Omega(\sqrt{\log k / \log\log k})$. On the other hand, the known upper bounds on f are polynomial in k. It is easy to see, for example by considering the complete graph on n nodes with treewidth $n-1$, that $f(k) = O(\sqrt{k})$. This can be slightly improved to $f(k) = O(\sqrt{k/\log k})$ by considering sparse random graphs (or $\Omega(\log n)$-girth constant-degree expanders) [22]. Robertson et al. [22] suggest that this value may be sufficient, and Demaine et al. [9] conjecture that the bound of $f(k) = \Theta(k^{1/3})$ is both necessary and sufficient. It has been an important open problem to prove a polynomial relationship between a graph's treewidth and the size of the largest grid-minor in it[1]. In this paper we prove the following theorem, which accomplishes this goal, while also giving a polynomial-time algorithm to find a large grid-minor.

Theorem 1.1 *There is a universal constant $\delta > 0$, such that for every $k \geq 1$, every graph G of treewidth k contains a grid-minor of size $\Omega(k^{\delta}) \times \Omega(k^{\delta})$. Moreover, there is a randomized algorithm that, given G, with high probability outputs a model of the grid-minor of size $\Omega(k^{\delta}) \times \Omega(k^{\delta})$ in G, and whose running time is polynomial in $|V(G)|$ and k.*

Our proof shows that δ is at least $\frac{1}{98} - o(1)$ in the preceding theorem. We obtain the following corollary by the observation that any planar graph H is a minor of a grid of size $k' \times k'$ where $k' = O(|V(H)|)$ [22].

[*]Supported in part by NSF grants CCF-1016684 & CCF-1319376, and by TTI Chicago during a sabbatical visit in Fall 2013.

[†]Supported in part by NSF CAREER grant CCF-0844872, NSF grant CCF-1318242 and Sloan Research Fellowship.

[1]The relationship between grid-minors and treewidth is much tighter in some special classes of graphs. In planar graphs $f(k) = \Omega(k)$ [22]; a similar linear relationship is known in bounded-genus graphs [10] and graphs that exclude a fixed graph H as a minor [11] (see also [15]).

Corollary 1.1 *There is a universal constant c such that, if G excludes a planar graph H as a minor, then the treewidth of G is $O(|V(H)|^c)$.*

The Grid-Minor theorem has several important applications in graph theory and algorithms, and also in proving lower bounds. The quantitative bounds in some of these applications can be directly improved by our main theorem. We anticipate that there will be other applications for our main theorem, and also for the algorithmic and graph-theoretic tools that we develop here.

Our proof and algorithm are based on a combinatorial object, called a path-of-sets system that we informally describe now; see Figure 1. A path-of-sets system of width r and height h consists of a collection of r disjoint sets of nodes $S_1, \ldots, S_r$ together with collections of paths $\mathcal{P}_1, \ldots, \mathcal{P}_{r-1}$ that are disjoint, which connect the sets in a path-like fashion. The number of paths in each set $\mathcal{P}_i$ is h. Moreover, for each i, the induced graph $G[S_i]$ satisfies the following routing properties for the end-points of the paths $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$ (sets A_i and B_i of vertices in the figure): for any pair $A \subseteq A_i$, $B \subseteq B_i$ of vertex subsets with $|A| = |B|$, there are $|A|$ node-disjoint paths connecting A to B in $G[S_i]$.

Given a path-of-sets system of width h and height h, we can efficiently find a grid minor of size $\Omega(h^{1/2}) \times \Omega(h^{1/2})$ in G, slightly strengthening a similar recent result of Leaf and Seymour [18], who use a related combinatorial object that they call an (h, r)-grill. Our main contribution is to show that, given a graph G with treewidth k, one can efficiently build a path-of-sets system of width h and height h, if $h^c \leq k/\text{polylog}(k)$, where c is a fixed constant. The central ideas for the construction build upon and extend recent work on algorithms for the Maximum Edge-Disjoint Paths problem with constant congestion [6, 7], and connections to treewidth [4, 2]. In order to construct the path-of-sets system, we use a closely related object, called a tree-of-sets system. The definition of the tree-of-sets system is very similar to the definition of the path-of-sets system, except that, instead of connecting the clusters S_i into a single long path, we connect them into a tree whose maximum vertex degree is at most 3. We extend and strengthen the results of [6, 7, 4], by showing an efficient algorithm, that, given a graph of treewidth k, constructs a large tree-of-sets system. We then show how to construct a large path-of-sets system, given a large tree-of-sets system. We believe that the tree-of-sets system is an interesting combinatorial object of independent interest and hope that future work will yield simpler and faster algorithms for constructing it, as well as improved parameters. This could lead to improvements in algorithms for related routing problems, and better bounds for Theorem 1.1.

We note that our definition of the tree-of-sets system requires very strong disjointness properties: the paths connecting the different clusters must be completely disjoint from each other, and internally disjoint from the clusters S_i. While the combinatorial objects constructed in [6] are somewhat similar to the tree-of-sets system, they do not provide this guarantee. The work of [7] only guarantees that every edge participates in at most two paths, or in one path and one cluster, while the work of [4] only guarantees that every vertex may participate in a constant number of such paths (for some large constant), in addition to at most one cluster. The main technical contributions of our paper, compared to previous work are (1) ensuring the strong disjointness properties of the tree-of-sets system; and (2) an algorithm that constructs a path-of-sets system from a tree-of-sets system. A useful technical tool that we develop is an algorithm for boosting well-linkedness, generalizing prior work on boosting edge-well-linkedness [5], which is of independent interest and has applications to reducing congestion in node-capacitated routing problems.

Organization: The paper is based on several technical results and the proof is somewhat long. Section 2 summarizes some of the high-level definitions and tools. Section 3 formally defines a path-of-sets system and states the theorem to construct a grid-minor from it. Section 4 describes the construction of a path-of-sets system from a tree-of-sets system. Section 5 describes the construction of a tree-of-sets system. Due to space constraints many details are omitted and the reader is referred to the full version of the paper [3].

2. PRELIMINARIES

All graphs in this paper are finite, they may have parallel edges, but no self-loops. Given a graph $G = (V, E)$ and a set of vertices $A \subseteq V$, we denote by $\text{out}_G(A)$ the set of edges with exactly one endpoint in A and by $E_G(A)$ the set of edges with both endpoints in A. For disjoint sets of vertices A and B, the set of edges with one end point in A and the other in B is denoted by $E_G(A, B)$. We may omit the subscript G if it is clear from the context. For a vertex v in a graph G we use $d_G(v)$ to denote its degree. Given a set $\mathcal{P}$ of paths in G, we denote by $V(\mathcal{P})$ the set of all vertices participating in paths in $\mathcal{P}$, and similarly, $E(\mathcal{P})$ is the set of all edges that participate in paths in $\mathcal{P}$. We sometimes refer to sets of vertices as *clusters*. All logarithms are to the base of 2.

A useful and simple claim that we need is the following.

Claim 2.1 *Let T be a rooted tree such that $|V(T)| \geq \ell p$ for some positive integers ℓ, p. Then T has at least ℓ leaves or a root-to-leaf path containing at least p vertices.*

The treewidth of a graph $G = (V, E)$ is typically defined via tree decompositions. A tree-decomposition for G consists of a tree $T = (V(T), E(T))$ and a collection of sets $\{X_v \subseteq V\}_{v \in V(T)}$ called bags, such that the following two properties are satisfied: (i) for each edge $(a, b) \in E$, there is some node $v \in V(T)$ with both $a, b \in X_v$ and (ii) for each vertex $a \in V$, the set of all nodes of T whose bags contain a form a non-empty (connected) subtree of T. The *width* of a given tree decomposition is $\max_{v \in V(T)} |X_v| - 1$, and the treewidth of a graph G, denoted by $\text{tw}(G)$, is the width of a minimum-width tree decomposition for G.

We say that a graph H is a minor of a graph G, iff H can be obtained from G by a sequence of edge deletion and contraction operations. Equivalently, H is a minor of G iff there is a map $\varphi : V(H) \to 2^{V(G)}$ assigning to each vertex $v \in V(H)$ a subset $\varphi(v)$ of vertices of G, such that: (1) For each $v \in V(H)$, the sub-graph of G induced by $\varphi(v)$ is connected; (2) If $u, v \in V(H)$ and $u \neq v$, then $\varphi(u) \cap \varphi(v) = \emptyset$; and (3) For each edge $e = (u, v) \in E(H)$, there is an edge in $E(G)$ with one endpoint in $\varphi(u)$ and the other endpoint in $\varphi(v)$. A map φ satisfying these conditions is called *a model of H in G*. We say that G contains a $(g \times g)$-grid minor iff some minor H of G is isomorphic to the $(g \times g)$ grid.

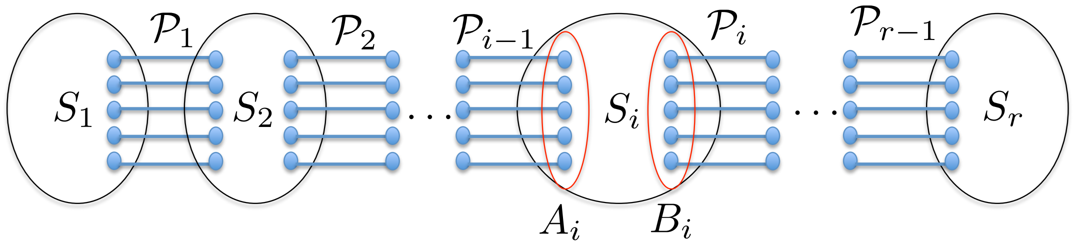


Figure 1: A path-of-sets system. Each set $\mathcal{P}_i$ contains h paths. All paths in $\bigcup_{i=1}^{r-1}\mathcal{P}_i$ are node-disjoint and internally disjoint from $\bigcup_{i=1}^{r} S_i$.

Sparsest Cut and the Flow-Cut Gap Suppose we are given a graph $G = (V, E)$, and a subset $\mathcal{T} \subseteq V$ of k terminals. The sparsity of a cut $(S, \overline{S})$ with respect to $\mathcal{T}$ is $\Phi(S) = \frac{|E(S,\overline{S})|}{\min\{|S\cap\mathcal{T}|,|\overline{S}\cap\mathcal{T}|\}}$, and the value of the sparsest cut in G is defined to be: $\Phi(G) = \min_{S\subset V}\{\Phi(S)\}$. The goal of the sparsest cut problem is, given an input graph G and a set $\mathcal{T}$ of terminals, to find a cut of minimum sparsity. Arora, Rao and Vazirani [1] have shown an $O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We denote this algorithm by $\mathcal{A}_{\text{ARV}}$, and its approximation factor by $\beta_{\text{ARV}}(k) = O(\sqrt{\log k})$.
A problem dual to sparsest cut is the maximum concurrent flow problem. We use the same graph G and set the capacity $c(e)$ of every edge e to 1. Given a flow $F : E \to \mathbb{R}^+$, the *edge-congestion* of the flow is the maximum, over all edges $e \in E$, of $F(e)/c(e)$. If the edge-congestion of F is at most 1, then we sometimes say that F causes no edge-congestion. For the above definition of the sparsest cut problem, the corresponding variation of the concurrent flow problem asks to find the maximum value λ, such that every pair of terminals can send λ/k flow units to each other simultaneously with no edge-congestion. The flow-cut gap is the maximum ratio, in any graph and for any set of k terminals in the graph, between the value of the minimum sparsest cut and the value λ^* of the maximum concurrent flow for the terminals in the graph. The flow-cut gap in undirected graphs, that we denote by $\beta_{\text{FCG}}(k)$ throughout the paper, is known to be $\Theta(\log k)$ [19]. Therefore, if $\Phi(G) = \alpha$, then every pair of terminals can simultaneously send $\frac{\alpha}{k\beta_{\text{FCG}}(k)}$ flow units to each other with no edge-congestion. Equivalently, every pair of terminals can send $1/k$ flow units to each other with edge-congestion at most $\beta_{\text{FCG}}(k)/\alpha$. Moreover, given any matching M on the set $\mathcal{T}$ of terminals, one unit of flow for each pair in M can be simultaneously routed with congestion at most $2\beta_{\text{FCG}}(k)/\alpha$.

Linkedness and Well-Linkedness We define the notion of linkedness and the different notions of well-linkedness that we use.

Definition 2.1 *We say that a set $\mathcal{T}$ of vertices is α-well-linked*[2] *in G, iff for any partition (A, B) of the vertices of G into two subsets, $|E(A,B)| \geq \alpha \cdot \min\{|A\cap\mathcal{T}|, |B\cap\mathcal{T}|\}$.*

[2]This notion of well-linkedness is based on edge-cuts and we distinguish it from node-well-linkedness that is directly related to treewidth. For technical reasons it is easier to work with edge-cuts and hence we use the term well-linked to mean edge-well-linkedness, and explicitly use the term node-well-linkedness when necessary.

Notice that if a set $\mathcal{T}$ of terminals is α-well-linked in G, then the value of the sparsest cut in G with respect to $\mathcal{T}$ is at least α.

Definition 2.2 *We say that a set $\mathcal{T}$ of vertices is* node-well-linked *in G, iff for any pair $(\mathcal{T}_1, \mathcal{T}_2)$ of equal-sized subsets of $\mathcal{T}$, there is a collection $\mathcal{P}$ of $|\mathcal{T}_1|$* **node-disjoint** *paths, connecting the vertices of $\mathcal{T}_1$ to the vertices of $\mathcal{T}_2$. (Note that $\mathcal{T}_1$, $\mathcal{T}_2$ are not necessarily disjoint, and we allow a singleton vertex as a path).*

Our algorithm proceeds by reducing the degree of the input graph to polylog(k), while preserving the treewidth to within a factor of polylog(k). The notions of edge- and node-well-linkedness can be related to each other via the maximum degree and we exploit this connection throughout the algorithm. We will repeatedly use the following simple claim, whose proof appears in the full version of the paper.

Claim 2.2 *Let $G = (V, E)$ be any graph, and let $\mathcal{T}$ be any subset of vertices of G called terminals, such that $\mathcal{T}$ is α-well-linked in G. Let E' be any subset of m' edges of G, such that $m' < \frac{\alpha|\mathcal{T}|}{3}$. Then there is a connected component C of $G \setminus E'$, containing at least $|\mathcal{T}| - \frac{m'}{\alpha}$ terminals.*

Corollary 2.1 *Let G be a graph with maximum vertex degree Δ, and let $\mathcal{T}$ be any subset of vertices of G such that $\mathcal{T}$ is α-well-linked in G. Let X be any subset of n' vertices of G, with $n' < \frac{\alpha|\mathcal{T}|}{3\Delta}$. Then there is a connected component C of $G \setminus X$ containing at least $|\mathcal{T}| - \frac{n'\Delta}{\alpha}$ vertices of $\mathcal{T}$.*

The corollary follows from Claim 2.2, by letting E' be the set of edges incident on the vertices of X.

Definition 2.3 *Two disjoint vertex subsets A, B are* linked *in G iff for any pair of equal-sized subsets $A' \subseteq A$, $B' \subseteq B$ there is a set $\mathcal{P}$ of $|A'| = |B'|$ node-disjoint paths connecting A' to B' in G.*

The proof of the following Theorem appears in the full version of the paper.

Theorem 2.1 *Suppose we are given two disjoint subsets $\mathcal{T}_1, \mathcal{T}_2$ of vertices of G, with $|\mathcal{T}_1|, |\mathcal{T}_2| \geq \kappa$, such that $\mathcal{T}_1 \cup \mathcal{T}_2$ is α-well-linked in G, and each one of the sets $\mathcal{T}_1, \mathcal{T}_2$ is node-well-linked in G. Let $\mathcal{T}'_1 \subset \mathcal{T}_1$, $\mathcal{T}'_2 \subset \mathcal{T}_2$, be any pair of subsets with $|\mathcal{T}'_1| = |\mathcal{T}'_2| \leq \frac{\alpha\kappa}{4\Delta}$. Then $\mathcal{T}'_1$ and $\mathcal{T}'_2$ are linked in G.*

Boosting Well-Linkedness Suppose we are given a graph G and a set $\mathcal{T}$ of vertices of G called terminals, where $\mathcal{T}$ is α-well-linked in G. Boosting theorems allow us to boost the well-linkedness by selecting an appropriate subset of the terminals, whose well-linkedness is greater than α. A standard spanning-tree based clustering scheme of [5] allows one to obtain a set $\mathcal{T}' \subseteq \mathcal{T}$ of $\Omega(\alpha|\mathcal{T}|)$ terminals, such that $\mathcal{T}'$ is 1/2-well-linked. However, we need a stronger result: we would like to find a large subset $\mathcal{T}' \subset \mathcal{T}$, such that $\mathcal{T}'$ is **node-well-linked** in G. The following theorem, whose proof appears in the full version of the paper, allows us to achieve this.

Theorem 2.2 *Suppose we are given a connected graph $G = (V, E)$ with maximum vertex degree Δ, and a subset $\mathcal{T}$ of κ vertices called terminals, such that $\mathcal{T}$ is α-well-linked in G, for some $\alpha < 1$, and $\kappa \geq 8\Delta/\alpha$. Then there is a subset $\mathcal{T}' \subset \mathcal{T}$ of $\Omega\left(\frac{\alpha\kappa}{\Delta}\right)$ terminals, such that $\mathcal{T}'$ is node-well-linked in G. Moreover, if $\kappa \geq 64\Delta^4\beta_{\text{ARV}}(\kappa)/\alpha$, then there is an efficient algorithm that computes a subset $\mathcal{T}' \subset \mathcal{T}$ of $\Omega\left(\frac{\alpha}{\Delta^5\beta_{\text{ARV}}(\kappa)} \cdot \kappa\right)$ terminals, such that $\mathcal{T}'$ is node-well-linked in G. The algorithm also computes, for each terminal $t \in \mathcal{T}'$, a tree $T_t \subseteq G$ containing at least $\lceil 1/\alpha \rceil$ terminals of $\mathcal{T}$, with $t \in V(T_t)$, such that all trees $\{T_t\}_{t \in \mathcal{T}'}$ are pairwise node-disjoint.*

Finally, we would like to obtain a slightly stronger result. Suppose we are given a connected graph $G = (V, E)$ with maximum vertex degree Δ, and r disjoint subsets $\mathcal{T}_1, \ldots, \mathcal{T}_r$ of vertices called terminals, such that $\mathcal{T} = \bigcup_j \mathcal{T}_j$ is α-well-linked in G, and $|\mathcal{T}_j| \geq \kappa$ for all j. We would like to select, for each $1 \leq j \leq r$, a large subset $\mathcal{T}_j^* \subset \mathcal{T}_j$ of terminals, such that $\mathcal{T}_j^*$ is node-well-linked in G, and for every pair $1 \leq j \neq j' \leq r$, $\mathcal{T}_j^*, \mathcal{T}_{j'}^*$ are linked in G. The following corollary, whose proof appears in the full version of the paper, combines Theorem 2.2 with Theorem 2.1 to achieve this goal.

Corollary 2.2 *Let G be a connected graph, with maximum vertex degree at most Δ, and let $\mathcal{T}_1, \ldots, \mathcal{T}_r$ be a collection of disjoint vertex subsets, called terminals, such that $\mathcal{T} = \bigcup_j \mathcal{T}_j$ is α-well-linked in G, and $|\mathcal{T}_j| \geq \kappa$ for all j, where $\kappa \geq 64\Delta^4\beta_{\text{ARV}}(\kappa)/\alpha$. Assume that we apply Theorem 2.2 to each set $\mathcal{T}_j$ of terminals in turn independently, and let $\mathcal{T}_j'$ be the outcome of Theorem 2.2 when applied to $\mathcal{T}_j$. Let $\mathcal{T}_j^* \subset \mathcal{T}_j'$ be any subset of $\frac{|\mathcal{T}_j'|}{12\Delta} = \Omega\left(\frac{\alpha}{\Delta^6\beta_{\text{ARV}}(\kappa)} \cdot \kappa\right)$ terminals. Then for each $1 \leq j \leq r$, $\mathcal{T}_j^*$ is node-well-linked in G, and for all $1 \leq i \neq j \leq r$, $\mathcal{T}_i^*$ and $\mathcal{T}_j^*$ are linked in G.*

Treewidth, Well-Linkedness and Degree Reduction The following lemma summarizes an important connection between the treewidth and the size of the largest node-well-linked set of vertices.

Lemma 2.1 (Reed [21]) *Let k be the size of the largest node-well-linked set in G. Then $k \leq \text{tw}(G) \leq 4k$.*

Lemma 2.1 guarantees that any graph G of treewidth k contains a set X of $\Omega(k)$ vertices, that is node-well-linked in G. The proof of Theorem 1.1 uses the notion of edge-well-linkedness as well as node-well-linkedness. In order to be able to translate between both types of well-linkedness we reduce the maximum vertex degree of the input graph G. Combining a lemma from Kreutzer and Tazari [17] and the cut-matching game of Khandekar, Rao and Vazirani [16], we obtain the following theorem that provides the starting point for our algorithm. The proof appears in the full version of the paper.

Theorem 2.3 *Let G be any graph with $\text{tw}(G) = k$. Then there is an efficient randomized algorithm to compute a subgraph G' of G with maximum vertex degree $\Delta = O(\log^3 k)$, and a subset X of $\Omega(k/\operatorname{poly}\log k)$ vertices of G', such that X is node-well-linked in G', with high probability.*

We note that one can also reduce the degree to a constant with an additional polylog(k) factor loss in the treewidth [4]; the constant can be made 4 with a polynomial factor loss in treewidth [17].

Re-Routing Two Sets of Disjoint Paths We need the following lemma, whose proof closely follows arguments of Conforti, Hassin and Ravi [8].

Lemma 2.2 *Let G be a directed graph, and let $\mathcal{X}, \mathcal{Y}$ be two sets of directed simple paths in G, where all paths in $\mathcal{X} \cup \mathcal{Y}$ share the same destination vertex s. The paths in $\mathcal{X}$ are disjoint from each other, except for sharing the destination s, and the same is true for $\mathcal{Y}$ (but a vertex $v \neq s$ may appear on a path in $\mathcal{X}$ and on a path in $\mathcal{Y}$.) Let H be the graph obtained by the union of the paths in $\mathcal{X} \cup \mathcal{Y}$. Then we can efficiently find a subset $\mathcal{X}' \subseteq \mathcal{X}$ of at least $|\mathcal{X}| - |\mathcal{Y}|$ paths, and for each path $Q \in \mathcal{Y}$, a path $\hat{Q}$ in graph H, with the same endpoints as Q, such that, if we denote $\mathcal{Y}' = \left\{\hat{Q} \mid Q \in \mathcal{Y}\right\}$, then all paths in $\mathcal{X}' \cup \mathcal{Y}'$ are pairwise disjoint (except for sharing the last vertex s).*

3. A PATH-OF-SETS SYSTEM

In this section we define our main combinatorial object, called the path-of-sets system. We start with a few definitions.

Suppose we are given a collection $\mathcal{S} = \{S_1, \ldots, S_r\}$ of disjoint vertex subsets of $V(G)$. Let $S_i, S_j \in \mathcal{S}$ be any two such subsets. We say that a path P *connects* S_i to S_j iff the first vertex of P belongs to S_i and the last vertex of P belongs to S_j. We say that P connects S_i to S_j *directly*, if additionally P does not contain any vertices of $\bigcup_{S \in \mathcal{S}} S$ as inner vertices.

Definition 3.1 *A path-of-sets system of width r and height h consists of:*

- *A sequence $\mathcal{S} = \{S_1, \ldots, S_r\}$ of r disjoint vertex subsets of G, where for each i, $G[S_i]$ is connected;*
- *For each $1 \leq i < r$, a set $\mathcal{P}_i$ of h disjoint paths, connecting S_i to S_{i+1}* **directly** *(that is, paths in $\mathcal{P}_i$ do not contain the vertices of $\bigcup_{S \in \mathcal{S}} S$ as inner vertices), such that all paths in $\bigcup_i \mathcal{P}_i$ are mutually disjoint;*

and has the following additional property. For each $1 \leq i < r$, let B_i be the set of vertices of $\mathcal{P}_i$ that belong to S_i, and let A_{i+1} be the set of vertices of $\mathcal{P}_i$ that belong to S_{i+1}. Then for each $1 < i < r$, sets A_i and B_i are linked in $G[S_i]$. (See Figure 1).

We note that Leaf and Seymour [18] have defined a very similar object, called an (h,r)-grill, and they implicitly showed that the two objects are roughly equivalent. Namely, a path-of-sets system with parameters h and r contains an (h,r)-grill as a minor, while an (h,r)-grill contains a path-of-sets system of height h and width $\Omega(r/h)$. They also show an efficient algorithm, that, given an (h,r)-grill with $h=\Omega(g^3)$ and $r=\Omega(g^4)$, finds a $(g\times g)$-grid minor in the grill[3]. The following theorem gives better bounds when the starting point is a path-of-sets system. The proof appears in the full version of the paper.

Theorem 3.1 *Given a path-of-sets system of width h and height h in a graph G, there is an efficient algorithm that finds a model of a grid minor of size $\Omega(\sqrt{h})\times\Omega(\sqrt{h})$ in G.*

4. FINDING A PATH-OF-SETS SYSTEM

We can view a path-of-sets system as a meta-path, whose vertices $v_1,\dots,v_r$ correspond to the sets $S_1,\dots,S_r$, and each edge $e=(v_i,v_{i+1})$ corresponds to the collection $\mathcal{P}_i$ of h disjoint paths. Unfortunately, we do not know how to find such a meta-path directly (except for $r=O(\log k)$, which is not enough for us). As we show below, a generalization of the work of [7], combined with some ideas from [4] gives a construction of a meta-tree of degree at most 3, instead of the meta-path. We define the corresponding object that we call a *strong tree-of-sets system*.

Definition 4.1 *A strong tree-of-sets system with parameters $r,\tilde{h}$ consists of:*

- *A collection $\mathcal{S}=\{S_1,\dots,S_r\}$ of r disjoint vertex subsets of G, where for each $1\le i\le r$, $G[S_i]$ is connected;*
- *A tree T over a set $\{v_1,\dots,v_r\}$ of vertices, whose maximum vertex degree is at most 3;*
- *For each edge $e=(v_i,v_j)$ of T, we are given a set $\mathcal{P}^*_e$ of $\tilde{h}$ disjoint paths, connecting S_i to S_j* **directly** *(that is, paths in $\mathcal{P}^*_e$ do not contain the vertices of $\bigcup_{S\in\mathcal{S}}S$ as inner vertices). Moreover, all paths in $\mathcal{P}^*=\bigcup_{e\in E(T)}\mathcal{P}^*_e$ are pairwise disjoint.*

  For a set $S_i\in\mathcal{S}$, and an edge $e\in E(T)$ incident on v_i, let $\Gamma_{S_i}(e)$ be the set of the endpoints of the paths in $\mathcal{P}^*_e$, that belong to S_i. We then have:
- *For each $S_i\in\mathcal{S}$, for every pair $e,e'\in E(T)$ of edges incident to v_i, where $e\neq e'$, the sets $\Gamma_{S_i}(e)$ and $\Gamma_{S_i}(e')$ are linked in $G[S_i]$.*

The following technical theorem expands and generalizes the results of [7].

Theorem 4.1 *Suppose we are given a graph G of maximum vertex degree $\Delta=O(\log^3 k)$, and a subset $\mathcal{T}$ of k vertices called terminals, such that $\mathcal{T}$ is node-well-linked in G and the degree of every vertex in $\mathcal{T}$ is 1. Additionally, assume that we are given parameters $r>1,\tilde{h}>4\log k$, such that $k/\log^8 k>c'\tilde{h}r^{23}\Delta^{19}$, where c' is a sufficiently large but fixed constant. Then there is an efficient randomized algorithm that with high probability computes a strong tree-of-sets system $(\mathcal{S},T,\bigcup_{e\in E(T)}\mathcal{P}^*_e)$ in G, with parameters $\lfloor r\rfloor,\lfloor\tilde{h}\rfloor$. Moreover, for all $S_i\in\mathcal{S}$, $S_i\cap\mathcal{T}=\emptyset$.*

[3]In fact [18] shows a slightly stronger result that a (h,r)-grill with $h\ge(2g+1)(2\ell-5)+2$ and $r\ge\ell(2g+\ell-2)$ contains a $g\times g$ grid-minor or a bipartite-clique $K_{\ell,\ell}$ as a minor. This can give slightly improved bounds on the grid-minor size if the given graph excludes bipartite-clique minors for small ℓ.

We defer the proof of Theorem 4.1 to Section 5. The following theorem is a central result of this section. It allows us to obtain a path-of-sets system from a strong tree-of-sets system.

Theorem 4.2 *There is an efficient algorithm, that, given a strong tree-of-set system $(\mathcal{S},T,\bigcup_{e\in E(T)}\mathcal{P}^*_e)$ with parameters $r,\tilde{h}$, and integers h^*,r^*, such that $(r^*)^2\le r$ and $\tilde{h}>16h^*(r^*)^2+1$, computes a path-of-sets system of width r^* and height h^*.*

Before we prove the preceding theorem, we use the results stated so far to complete the proof of the main result of the paper.

Proof of Theorem 1.1. We assume that k is large enough, so, e.g. $k^{1/30}>c^*\log k$ for some large enough constant c^*. Given a graph $G=(V,E)$ with treewidth k, we use Theorem 2.3 to compute a subgraph G' of G with maximum vertex degree $\Delta=O(\log^3 k)$, and a set X of $\Omega(k/\operatorname{poly}\log k)$ vertices, such that X is node-well-linked in G'. We add a new set $\mathcal{T}$ of $|X|$ vertices, each of which connects to a distinct vertex of X with an edge. For convenience, we denote this new graph by G, and $|\mathcal{T}|$ by k, and we refer to the vertices of $\mathcal{T}$ as *terminals*. Clearly, the maximum vertex degree of G is at most $\Delta=O(\log^3 k)$, the degree of every terminal is 1, and $\mathcal{T}$ is node-well-linked in G. We then apply Theorem 4.1 to G and $\mathcal{T}$ to obtain a strong tree-of-sets system $(\mathcal{S},T,\bigcup_{e\in E(T)}\mathcal{P}^*_e)$, with parameters $r=\frac{k^{2/49}}{\log^2 k}$, $\tilde{h}=\frac{k^{3/49}}{\log^{24}k}$. It is easy to verify that the conditions of the theorem hold for these values.
We then apply Theorem 4.2 to obtain a path-of-set system with height $h^*=\lfloor\frac{\tilde{h}^{1/3}}{4}\rfloor$ and width $r^*=h^*$. Again it is easy to verify that the conditions of the theorem are satisfied for these choices. Finally we use Theorem 3.1 to construct a grid minor of size $\Omega(\sqrt{h^*})\times\Omega(\sqrt{h^*})$. Notice that $h^*=\Theta(\tilde{h}^{1/3})=\Theta(k^{1/49}/\operatorname{poly}\log k)$, and the the size of the grid minor computed by the algorithm is $\left(\Omega\left(\frac{k^{1/98}}{\operatorname{poly}\log k}\right)\times\Omega\left(\frac{k^{1/98}}{\operatorname{poly}\log k}\right)\right)$. □

The rest of this section is devoted to proving Theorem 4.2. Let $(\mathcal{S},T,\cup_{e\in E(T)}\mathcal{P}^*_e)$ be a strong tree-of-set system with parameters r and $\tilde{h}$. Let h^*,r^* be integers such that $(r^*)^2\le r$ and $\tilde{h}>16h^*(r^*)^2+1$.
For convenience, for each set $S\in\mathcal{S}$, we denote the corresponding vertex of T by v_S. If tree T contains a root-to-leaf path of length at least r^*, then we are done, as this path gives a path-of-sets system of height $\tilde{h}\ge h^*$ and width r^*. Otherwise, since $|V(T)|=r\ge(r^*)^2$, T must contain at least r^* leaves (see Claim 2.1). Let L be any subset of r^* leaves of T (if there are more leaves in T, we only choose r^* of them). Let $\mathcal{L}=\{S\in\mathcal{S}\mid v_S\in L\}$ be the collection of r^* clusters whose corresponding vertices belong to L. We next show how to build a path-of-sets system, whose collection of sets is $\mathcal{L}$.

Intuitively, we would like to perform a depth-first-search tour on the meta-tree T. This should be done with many paths in parallel. In other words, we want to build h^* disjoint paths, that visit the sets in $\mathcal{S}$ in the same order — the order of the tour. The clusters in $\mathcal{L}$ will then serve as the sets $\mathcal{S}$ in our final path-of-sets system, and the collection of h^* paths that we build will be used for the paths $\mathcal{P}_i$. In order for this to work, we need to route up to three sets of paths across clusters $S \in \mathcal{S}$. For example, if the vertex v_S corresponding to the cluster S is a degree-3 vertex in T, then for the DFS tour, we need to route three sets of paths across S: one set connecting the paths coming from v_S's parent to its first child, one set connecting the paths coming back from the first child to the second child, and one set connecting the paths coming back from the second child to its parent. Even though every pair of relevant vertex subsets on the interface of S is linked, this property only guarantees that we can route one such set of paths, which presents a major technical difficulty in using this approach directly. Moreover, it is not clear how to coordinate the routing between different clusters, without losing too many paths.
Our algorithm consists of two phases. In the first phase, we build a collection of disjoint paths, connecting the cluster corresponding to the root of the tree T to the clusters in $\mathcal{L}$, along the root-to-leaf paths in T. In the second phase, we build the path-of-sets system by exploiting the paths constructed in Step 1, to simulate the tree tour.

4.1 Step 1

Let G' be the graph obtained from the union of $G[S]$ for all $S \in \mathcal{S}$, and the sets $\mathcal{P}^*_e$ of paths, for all $e \in E(T)$. We root T at any degree-1 vertex, and we let S^* be the cluster corresponding to the root of T. The goal of the first step is summarized in the following theorem.

Theorem 4.3 *We can efficiently compute in graph G', for each $S \in \mathcal{L}$, a collection $\mathcal{Q}_S$ of $\lfloor \tilde{h}/r^* \rfloor$ paths, that have the following properties:*

- *Each path $Q \in \mathcal{Q}_S$ starts at a vertex of S^* and terminates at a vertex of S; its inner vertices are disjoint from S and S^*.*
- *For each path $Q \in \mathcal{Q}_S$, for each cluster S', where $v_{S'}$ lies on the path connecting v_{S^*} to v_S in T, the intersection of Q with $G'[S']$ is a contiguous segment of Q. For any cluster S' where $v_{S'}$ is not on the path connecting v_{S^*} to v_S in T, $Q \cap S' = \emptyset$.*
- *The paths in $\mathcal{Q} = \bigcup_{S \in \mathcal{L}} \mathcal{Q}_S$ are vertex-disjoint.*

Notice that from the structure of graph G', if P is the path connecting v_{S^*} to v_S in the tree T, then every path in $\mathcal{Q}_S$ visits every cluster S' with $v_{S'} \in P$ exactly once, in the order in which they appear on P, and it does not visit any other clusters of $\mathcal{S}$.

Proof. For each cluster $S' \in \mathcal{S}$, let $n(S')$ be the number of the descendants of $v_{S'}$ in the tree T that belong to L. If $S' \neq S^*$, then let e be the edge of the tree T connecting $v_{S'}$ to its parent. Let $\Gamma_{S'} = \Gamma_{S'}(e)$ be the set of vertices of S' that serve as endpoints of the paths in $\mathcal{P}^*_e$. We process the tree in top to bottom order, while maintaining a set $\mathcal{Q}$ of disjoint paths. We ensure that the following invariant holds throughout the algorithm. Let S, S' be any pair of clusters, such that v_S is the parent of $v_{S'}$ in T. Assume that so far the algorithm has processed v_S but it has not processed $v_{S'}$ yet. Then there is a collection $\mathcal{Q}_{S'} \subseteq \mathcal{Q}$ of $n(S') \cdot \lfloor \tilde{h}/r^* \rfloor$ paths connecting S^* to S' in $\mathcal{Q}$. Each such path does not share vertices with S', except for its last vertex, which must belong to $\Gamma_{S'}$. Moreover, for every path $Q \in \mathcal{Q}_{S'}$, for every cluster S'' where $v_{S''}$ lies on the path connecting v_{S^*} to $v_{S'}$ in T, the intersection of $G'[S'']$ and Q is a contiguous segment of Q, and for any other cluster S'', $Q \cap S'' = \emptyset$.
In the first iteration, we start with the root vertex v_{S^*}. Let v_S be its unique child, and let $e = (v_{S^*}, v_S)$ be the corresponding edge of T. We let $\mathcal{Q}_S$ be any subset of $n(S) \cdot \lfloor \tilde{h}/r^* \rfloor$ paths of $\mathcal{P}^*_e$, and we set $\mathcal{Q} = \mathcal{Q}_S$. (Notice that $|L| \cdot \lfloor \tilde{h}/r^* \rfloor \leq \tilde{h} = |\mathcal{P}^*_e|$, since $|L| = r^*$, so we can always find such a subset of paths).
Consider now some non-leaf vertex v_S, and assume that its parent has already been processed. We assume that v_S has two children. The case where v_S has only one child is treated similarly. Let $\mathcal{Q}_S \subset \mathcal{Q}$ be the subset of paths currently connecting S^* to S, and let $\Gamma' \subseteq \Gamma_S$ be the endpoints of these paths that belong to S. Let $v_{S'}, v_{S''}$ be the children of v_S in T, and let $e_1 = (v_S, v_{S'}), e_2 = (v_S, v_{S''})$ be the corresponding edges of T. We need the following claim.

Claim 4.1 *We can efficiently find a subset $\Gamma_1 \subset \Gamma_S(e_1)$ of $n(S') \cdot \lfloor \tilde{h}/r^* \rfloor$ vertices and a subset $\Gamma_2 \subset \Gamma_S(e_2)$ of $n(S'') \cdot \lfloor \tilde{h}/r^* \rfloor$ vertices, together with a set $\mathcal{R}$ of $|\Gamma'|$ disjoint paths contained inside $G'[S]$, where each path connects a vertex of Γ' to a distinct vertex of $\Gamma_1 \cup \Gamma_2$.*

Proof. We build the following flow network, starting with $G[S]$. Set the capacity of every vertex in S to 1. Add a sink t, and connect every vertex in Γ' to t with a directed edge. Add a new vertex s_1 of capacity $n(S') \cdot \lfloor \tilde{h}/r^* \rfloor$ and connect it with a directed edge to every vertex of $\Gamma_S(e_1)$. Similarly, add a new vertex s_2 of capacity $n(S'') \cdot \lfloor \tilde{h}/r^* \rfloor$ and connect it with a directed edge to every vertex of $\Gamma_S(e_2)$. Finally, add a source s and connect it to s_1 and s_2 with directed edges.
From the integrality of flow, it is enough to show that there is an s-t flow of value $|\Gamma'| = n(S) \cdot \lfloor \tilde{h}/r^* \rfloor = (n(S') + n(S'')) \cdot \lfloor \tilde{h}/r^* \rfloor$ in this flow network. Since Γ' and $\Gamma_S(e_1)$ are linked, there is a set $\mathcal{P}_1$ of $|\Gamma'|$ disjoint paths connecting the vertices of Γ' to the vertices of $\Gamma_S(e_1)$. We send $n(S')/n(S)$ flow units along each such path. Similarly, there is a set $\mathcal{P}_2$ of $|\Gamma'|$ disjoint paths connecting vertices of Γ' to vertices of $\Gamma_S(e_2)$. We send $n(S'')/n(S)$ flow units along each such path. It is immediate to verify that this gives a feasible s-t flow of value $|\Gamma'|$ in this network. $\square$

Let $\mathcal{P}_1 \subseteq \mathcal{P}^*(e_1)$ be a subset of paths whose endpoints belong to Γ_1, and define $\mathcal{P}_2 \subseteq \mathcal{P}^*(e_2)$ similarly for Γ_2. Concatenating the paths in $\mathcal{Q}_S$, $\mathcal{R}$, and $\mathcal{P}_1 \cup \mathcal{P}_2$, we obtain two collections of paths: set $\mathcal{Q}_{S'}$ of $n(S') \cdot \lfloor \tilde{h}/r^* \rfloor$ paths, connecting S^* to S', and set $\mathcal{Q}_{S''}$ of $n(S'') \cdot \lfloor \tilde{h}/r^* \rfloor$ paths, connecting S^* to S'', that have the desired properties. We delete the paths of $\mathcal{Q}_S$ from $\mathcal{Q}$, and add the paths in $\mathcal{Q}_{S'}$ and $\mathcal{Q}_{S''}$ instead.
Once all non-leaf vertices of the tree T are processed, we obtain the desired collection of paths.

4.2 Step 2

In this step, we process the tree T in the bottom-up order, gradually building the path-of-sets system. We will imitate the depth-first-search tour of the tree, and exploit the sets $\{\mathcal{Q}_S \mid S \in \mathcal{L}\}$ of paths constructed in Step 1 to perform this step.

For every vertex v_S of the tree T, let T_{v_S} be the subtree of T rooted at v_S. Define a sub-graph G_S of G' to be the union of all clusters $G'[S']$ with $v_{S'} \in V(T_{v_S})$, and all sets $\mathcal{P}^*_e$ of paths with $e \in E(T_{v_S})$. We also define $L_S \subseteq L$ to be the set of all descendants of v_S that belong to L, and $\mathcal{L}_S = \{S' \mid v_{S'} \in L_S\}$ the collection of the corresponding clusters.

We process the tree T in a bottom to top order, maintaining the following invariant. Let v_S be any vertex of T, and let ℓ_S be the length of the longest simple path connecting v_S to any of its descendants in T. Once vertex v_S is processed, we have computed a path-of-sets system $(\mathcal{L}_S, \mathcal{P}^S)$ of height h^* and width $|\mathcal{L}_S|$, that is completely contained in G_S. (That is, the path-of-sets system is defined over the collection $\mathcal{L}_S$ of vertex subsets - all subsets $S' \in \mathcal{L}$ where $v_{S'}$ is a descendant of v_S in T). Let $A, B \in \mathcal{L}_S$ be the first and the last set on the path-of-sets system. Then we also compute subsets $\mathcal{Q}'_A \subseteq \mathcal{Q}_A$, $\mathcal{Q}'_B \subseteq \mathcal{Q}_B$ of paths of size at least $\lfloor \frac{\tilde{h}}{2r^*} \rfloor - 8\ell_S \cdot h^*$, such that the paths in $\mathcal{Q}'_A \cup \mathcal{Q}'_B$ are completely disjoint from the paths in $\mathcal{P}^S$ (see Figure 2). Note that $\mathcal{Q}_A, \mathcal{Q}_B$ are the sets of paths computed in Step 1, so the paths in $\mathcal{Q}_A \cup \mathcal{Q}_B$ are also disjoint from $\bigcup_{S' \in \mathcal{L}} S'$, except that one endpoint of each such path must belong to A or B. We note that since the tree height is bounded by r^*, $\lfloor \frac{\tilde{h}}{2r^*} \rfloor - 8\ell_S \cdot h^* \geq \lfloor \frac{\tilde{h}}{2r^*} \rfloor - 8r^* \cdot h^* > 0$ where the latter inequality follows from the assumption that $\tilde{h} > 16h^*(r^*)^2 + 1$.

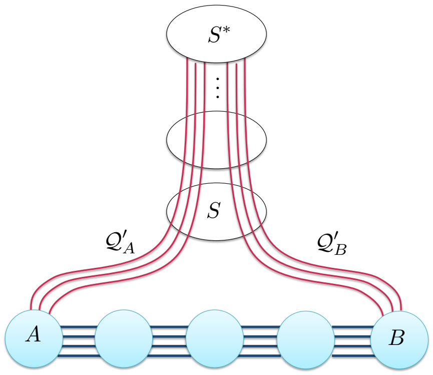
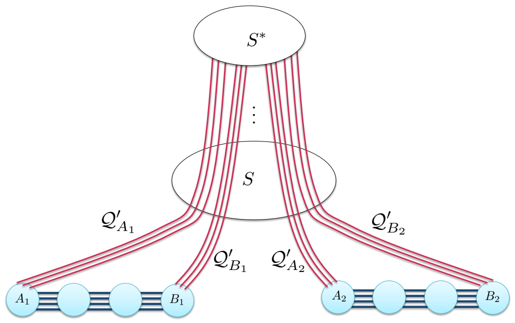


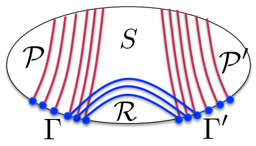
Figure 2: Invariant for Step 2.

Clearly, once all vertices of the tree T are processed, we obtain the desired path-of-sets system $(\mathcal{L}, \mathcal{P})$ of width r^* and height h^*. We now describe the algorithm for processing each vertex.

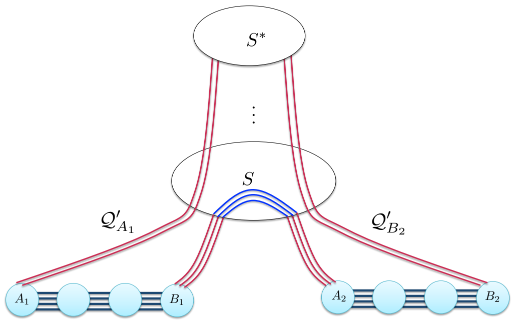
If v_S is a leaf of T, then we do nothing. If $v_S \in L$, then the path-of-sets system consists of only $\mathcal{S} = \{S\}$, with $A = B = S$. We let $\mathcal{Q}'_A, \mathcal{Q}'_B$ be any pair of disjoint subsets of $\mathcal{Q}_S$ containing $\lfloor \frac{\tilde{h}}{2r^*} \rfloor$ paths each. If v_S is a degree-2 vertex of T, then we also do nothing. The path-of-sets system is inherited from its child, and the corresponding sets $\mathcal{Q}'_A, \mathcal{Q}'_B$ remain unchanged. Assume now that v_S is a degree-3 vertex, and let $v_{S'}, v_{S''}$ be its two children. Consider the path-of-sets systems that we computed for its children: $(\mathcal{L}_{S'}, \mathcal{P}^{S'})$



(a) The beginning



(b) Finding the set $\mathcal{R}$



(c) The end

Figure 3: Processing a degree-3 vertex v_S.

for S' and $(\mathcal{L}_{S''}, \mathcal{P}^{S''})$ for S''. Let A_1, B_1 be the first and the last cluster of the first system, and A_2, B_2 the first and the last cluster of the second system (see Figure 3(a)). The idea is to connect the two path-of-sets systems into a single system, by joining one of $\{A_1, B_1\}$ to one of $\{A_2, B_2\}$ by h^* disjoint paths. These paths will be a concatenation of sub-paths of some paths from $\mathcal{Q}'_{A_1} \cup \mathcal{Q}'_{B_1} \cup \mathcal{Q}'_{A_2} \cup \mathcal{Q}'_{B_2}$, and additional paths contained inside S.

Consider the paths in $\mathcal{Q}'_{A_1}$ and direct these paths from A_1 towards S^*. For each such path Q, let v_Q be the first vertex of Q that belongs to S. Let $\Gamma_1 = \{v_Q \mid q \in \mathcal{Q}'_{A_1}\}$. We similarly define $\Gamma_2, \Gamma'_1, \Gamma'_2$ for $\mathcal{Q}'_{B_1}$, $\mathcal{Q}'_{A_2}$ and $\mathcal{Q}'_{B_2}$, respectively. Denote $\Gamma = \Gamma_1 \cup \Gamma_2$, and $\Gamma' = \Gamma'_1 \cup \Gamma'_2$. For simplicity, we denote the portions of the paths in $\mathcal{Q}'_{A_1} \cup \mathcal{Q}'_{B_1}$ that are contained in S by $\mathcal{P}$, and the portions of paths in $\mathcal{Q}'_{A_2} \cup \mathcal{Q}'_{B_2}$ that are contained in S by $\mathcal{P}'$ (see Figure 3(b)).

Our goal is to find a set $\mathcal{R}$ of $4h^*$ disjoint paths inside S connecting Γ to Γ', such that the paths in $\mathcal{R}$ intersect at

most $8h^*$ paths in $\mathcal{P}$, and at most $8h^*$ paths in $\mathcal{P}'$. Since sets Γ, Γ' are linked in $G'[S]$, we can find a set $\mathcal{R}$ of $4h^*$ disjoint paths inside S connecting Γ to Γ', however these paths may intersect many paths in $\mathcal{P} \cup \mathcal{P}'$. We start from an arbitrary set $\mathcal{R}$ of $4h^*$ disjoint paths connecting Γ to Γ' inside S. We next re-route these paths, using Lemma 2.2.
We apply Lemma 2.2 twice. First, we unify all vertices of Γ into a single vertex s, and direct the paths in $\mathcal{P}$ and the paths in $\mathcal{R}$ towards it. We then apply Lemma 2.2 to the two sets of paths, with $\mathcal{P}$ as $\mathcal{X}$ and $\mathcal{R}$ as $\mathcal{Y}$. Let $\tilde{\mathcal{P}} \subset \mathcal{P}$, $\mathcal{R}'$ be the two resulting sets of paths. We discard from $\tilde{\mathcal{P}}$ paths that share endpoints with paths in $\mathcal{R}'$ (at most $|\mathcal{R}'|$ paths). Then $|\tilde{\mathcal{P}}| \geq |\mathcal{P}| - 2|\mathcal{R}| = |\mathcal{P}| - 8h^*$, and $\mathcal{R}'$ contains $4h^*$ disjoint paths connecting vertices in Γ to vertices in Γ'. Moreover, the paths in $\tilde{\mathcal{P}} \cup \mathcal{R}'$ are completely disjoint.
Next, we unify all vertices in Γ' into a single vertex s, and direct all paths in $\mathcal{P}'$ and $\mathcal{R}'$ towards s. We then apply Lemma 2.2 to the two resulting sets of paths, with $\mathcal{P}'$ serving as $\mathcal{X}$ and $\mathcal{R}'$ serving as $\mathcal{Y}$. Let $\tilde{\mathcal{P}}' \subset \mathcal{P}$ and $\mathcal{R}''$ be the two resulting sets of paths. We again discard from $\tilde{\mathcal{P}}'$ all paths that share an endpoint with a path in $\mathcal{R}''$ – at most $|\mathcal{R}''|$ paths. Then $|\tilde{\mathcal{P}}'| \geq |\tilde{\mathcal{P}}| - 2|\mathcal{R}''| \geq |\tilde{\mathcal{P}}| - 8h^*$, and the paths in $\tilde{\mathcal{P}}' \cup \mathcal{R}''$ are completely disjoint from each other. Notice also that the paths in $\mathcal{R}''$ remain disjoint from the paths in $\tilde{\mathcal{P}}$, since the paths in $\mathcal{R}''$ only use vertices that appear on the paths in $\mathcal{R}' \cup \mathcal{P}'$, which are disjoint from $\tilde{\mathcal{P}}$.
Consider now the final set $\mathcal{R}''$ of paths. The paths in $\mathcal{R}''$ connect the vertices of $\Gamma_1 \cup \Gamma_2$ to the vertices of $\Gamma'_1 \cup \Gamma'_2$. There must be two indices $i, j \in \{1, 2\}$, such that at least a quarter of the paths in $\mathcal{R}''$ connect the vertices of Γ_i to the vertices of Γ'_j. We assume without loss of generality that $i = 2, j = 1$, so at least h^* of the paths in $\mathcal{R}''$ connect the vertices of Γ_2 to the vertices of Γ'_1. Let $\mathcal{R}^* \subset \mathcal{R}''$ be the set of these paths. We obtain a collection $\mathcal{P}^*$ of h^* paths connecting B_1 to A_2, by concatenating the prefixes of the paths in $\mathcal{Q}'_{B_1}$, the paths in $\mathcal{R}''$, and the prefixes of the paths in $\mathcal{Q}'_{A_2}$ (see Figure 3(c)). Notice that the paths in $\mathcal{P}^*$ are completely disjoint from the two path-of-sets systems, except for their endpoints that belong to B_1 and A_2. This gives us a new path-of-sets system, whose collection of sets is $\mathcal{S} = \mathcal{L}_S$. The first and the last sets in this system are A_1 and B_2, respectively. In order to define the new set $\mathcal{Q}'_{A_1}$, we discard from $\mathcal{Q}'_{A_1}$ all paths that share vertices with paths in $\mathcal{R}''$ (as observed before, there are at most $8h^*$ such paths). Since, at the beginning of the current iteration, $|\mathcal{Q}'_{A_1}| \geq \lfloor \frac{\tilde{h}}{2r^*} \rfloor - 8h^* \ell_{S'} \geq \lfloor \frac{\tilde{h}}{2r^*} \rfloor - 8h^*(\ell_S - 1)$, at the end of the current iteration, $|\mathcal{Q}'_{A_1}| \geq \lfloor \frac{\tilde{h}}{2r^*} \rfloor - 8h^* \ell_S$ as required. The new set $\mathcal{Q}'_{B_2}$ is defined similarly. From the construction, the paths in $\mathcal{Q}'_{A_1} \cup \mathcal{Q}'_{B_2}$ are completely disjoint from the paths in $\mathcal{R}^*$, and hence they are completely disjoint form all paths participating in the new path-of-sets system. This finishes the proof of Theorem 4.2.
In order to complete the proof of Theorem 1.1, it is now enough to prove Theorem 4.1. We do so in the following section.

5. FINDING A TREE-OF-SETS SYSTEM

In this section we prove Theorem 4.1. For this purpose we first define a weaker version of the tree-of-sets system, that is easier to work with, and then show how one can obtain a strong tree-of-set systems from it.

Definition 5.1 *Given a set S of vertices in graph G, the* interface *of S is $\Gamma = \{v \in S \mid \exists e = (u, v) \in \text{out}_G(S)\}$. We say that S has the α-bandwidth property in G iff its interface Γ is α-well-linked in $G[S]$.*

Definition 5.2 *A weak tree-of-sets system with parameters r, h, α_{BW} consists of:*

- *A collection $\mathcal{S} = \{S_1, \ldots, S_r\}$ of r disjoint vertex subsets of G, where for each $1 \leq i \leq r$, $G[S_i]$ is connected;*
- *A tree T over a set $\{v_1, \ldots, v_r\}$ of vertices, whose maximum vertex degree is at most 3;*
- *For each edge $e = (v_i, v_j)$ of T, we are given a set $\mathcal{P}_e$ of h disjoint paths, connecting S_i to S_j* **directly** *(that is, paths in $\mathcal{P}_e$ do not contain the vertices of $\bigcup_{S \in \mathcal{S}} S$ as inner vertices). Moreover, all paths in $\mathcal{P} = \bigcup_{e \in E(T)} \mathcal{P}_e$ are pairwise disjoint,*

and has the following additional property. Let G' be the subgraph of G obtained by the union of $G[S_i]$ for all $S_i \in \mathcal{S}$ and $\bigcup_{e \in E(T)} \mathcal{P}(e)$. Then each $S_i \in \mathcal{S}$ has the α_{BW}-bandwidth property in G'.

In the following lemma, we show how to obtain a strong tree-of-sets system from a weak tree-of-sets system.

Lemma 5.1 *There is an efficient randomized algorithm, that, given a graph G with maximum vertex degree Δ, and a weak tree-of-sets system $(\mathcal{S}, T, \bigcup_{e \in E(T)} \mathcal{P}_e)$ with parameters r, h, α_{BW} in G, computes a strong tree-of-sets system $(\mathcal{S}, T, \bigcup_{e \in E(T)} \mathcal{P}^*_e)$ with parameters $r, \tilde{h}$, such that*

- *$\mathcal{P}^*_e \subset \mathcal{P}_e$ for each $e \in E(T)$*
- *$\tilde{h} = \Omega(\frac{\alpha^2_{\text{BW}}}{\Delta^{11}(\beta_{\text{ARV}}(h))^2} \cdot h)$*

Proof. We prove the lemma by boosting well-linkedness using the claims in Section 2. Consider the given weak tree-of-sets system $(\mathcal{S}, T, \cup_{e \in E(T)} \mathcal{P}_e)$ in G. Let $S_i \in \mathcal{S}$. Consider some edge $e = (v_i, v_j)$ of T. Let Γ_1 be the set of endpoints of the paths in $\mathcal{P}_e$ that belong to S_i, and recall that Γ_1 is α_{BW}-well-linked in $G[S_i]$. We apply Theorem 2.2 to S_i and Γ_1, to obtain a subset $\Gamma'_1 \subseteq \Gamma_1$ of $\Theta\left(\frac{\alpha_{\text{BW}}}{\Delta^5 \beta_{\text{ARV}}(h)} \cdot h\right)$ vertices, such that Γ'_1 is node-well-linked in $G[S_i]$. Let $\mathcal{P}'_e \subset \mathcal{P}_e$ be a subset of paths whose endpoint belongs to Γ'_1, so $|\mathcal{P}'_e| = \Theta\left(\frac{\alpha_{\text{BW}}}{\Delta^5 \beta_{\text{ARV}}(h)} \cdot h\right)$. Let Γ_2 be the set of endpoints of the paths in $\mathcal{P}'_e$ that belong to S_j. We apply Theorem 2.2 to S_j and Γ_2 to obtain a subset $\Gamma'_2 \subset \Gamma_2$ of $\Theta\left(\left(\frac{\alpha_{\text{BW}}}{\Delta^5 \beta_{\text{ARV}}(h)}\right)^2 \cdot h\right)$ vertices, such that Γ'_2 is node-well-linked in S_j. Let $\mathcal{P}''_e \subset \mathcal{P}'_e$ be a subset of all paths whose endpoint belongs to Γ'_2. Finally, we select an arbitrary subset $\mathcal{P}^*_e$ of $\tilde{h} = \lfloor \frac{|\mathcal{P}''_e|}{12\Delta} \rfloor = \Theta\left(\frac{\alpha^2_{\text{BW}}}{\Delta^{11}(\beta_{\text{ARV}}(h))^2} \cdot h\right)$ paths. We denote by $\Gamma_{S_i}(e)$ the endpoints of the paths in $\mathcal{P}^*_e$ that belong to S_i, and we denote by $\Gamma_{S_j}(e)$ the endpoints of the paths in $\mathcal{P}^*_e$ that belong to S_j. We process every edge $e \in T$ in this manner. Consider now any non-leaf vertex $v_i \in T$ and its corresponding set $S_i \in \mathcal{S}$. Let $e \neq e'$ be any pair of edges incident on v_i in T. Then from Corollary 2.2, sets $\Gamma_{S_i}(e)$ and $\Gamma_{S_i}(e')$ of vertices are linked in $G[S_i]$. $\square$

The following theorem is the main result of this section.

Theorem 5.1 *Suppose we are given a graph G of maximum vertex degree $\Delta = O(\log^3 k)$, and a subset $\mathcal{T}$ of k vertices called terminals, such that $\mathcal{T}$ is node-well-linked in G and the degree of every vertex in $\mathcal{T}$ is 1. Additionally, assume that we are given any parameters $r > 1, h > 4\log k$, such that $k/\log^4 k > chr^{19}\Delta^8$, where c is a large enough constant. Then there is an efficient randomized algorithm that with high probability computes a weak tree-of-sets system $(\mathcal{S}, T, \bigcup_{e\in E(T)} \mathcal{P}_e)$ in G, with parameters $\lfloor h\rfloor, \lfloor r\rfloor$ and $\alpha_{\text{BW}} = \Omega\left(\frac{1}{r^2\log^{1.5}k}\right)$. Moreover, for all $S_i \in \mathcal{S}$, $S_i \cap \mathcal{T} = \emptyset$.*

Before we plunge into the proof of Theorem 5.1, we show how to complete the proof of Theorem 4.1.

Proof of Theorem 4.1. Let G be graph with maximum degree $\Delta = O(\log^3 k)$ and a node-well-linked set $\mathcal{T}$ of k terminals each of which has degree 1 in G. Let $\tilde{h}, r$ be parameters such that $k/\log^8 k > c'\tilde{h}r^{23}\Delta^{19}$. Let $\alpha_{\text{BW}} = \Omega\left(\frac{1}{r^2\log^{1.5}k}\right)$, as in the statement of Theorem 5.1. Let $h \geq \tilde{h}$ be the smallest number such that $\tilde{h} = c'' \cdot \frac{\alpha_{\text{BW}}^2}{\Delta^{11}(\beta_{\text{ARV}}(h))^2} \cdot h$ for some sufficiently large constant c''. If G has a tree-of-sets system $(\mathcal{S}, T, \bigcup_{e\in E(T)} \mathcal{P}_e)$ with parameters r, h, α_{BW}, then, via Lemma 5.1, we can obtain a strong tree-of-sets system $(\mathcal{S}, T, \bigcup_{e\in E(T)} \mathcal{P}^*_e)$ with parameters $r, \tilde{h}$. Thus it suffices to verify that the parameters r, h, α_{BW} satisfy the conditions of Theorem 5.1, that is, we need to check that $k/\log^4 k \geq chr^{19}\Delta^8$. It is easy to verify this given the choice of α_{BW} and h and the assumption that $k/\log^8 k > c'\tilde{h}r^{23}\Delta^{19}$. $\square$

The proof of Theorem 5.1 mostly follows the algorithm of [7]. The main difference is a change in the parameters, so that the number of clusters in the tree-of-sets system is polynomial in k and not polylogarithmic, and extending the arguments of [7] to handle vertex connectivity instead of edge connectivity. We also improve and simplify some of the arguments of [7]. For simplicity, if $(\mathcal{S}, T, \bigcup_{e\in E(T)} \mathcal{P}_e)$ is a weak tree-of-sets system in G, with parameters h, r, α_{BW} as in the theorem statement, and for each $S_i \in \mathcal{S}$, $S_i \cap \mathcal{T} = \emptyset$, then we say that it is a *good tree-of-sets system.* In this version of the paper we limit ourselves to giving a very high-level overview of the proof and refer the reader to [3].

The proof uses two main parameters: $r_0 = r^2$, and $h_0 = h \cdot \text{poly}(r \cdot \Delta \cdot \log k)$. We say that a subset S of vertices of G is a *good router* iff the following three conditions hold: (1) $S \cap \mathcal{T} = \emptyset$; (2) S has the α_{BW}-bandwidth property; and (3) S can send a large amount of flow (say at least $h_0/2$ flow units) to $\mathcal{T}$ with no edge-congestion in G. A collection of r_0 disjoint good routers is called a *good family of routers.* Roughly, the proof consists of two parts. The first part shows how to find a good family of routers, and the second part shows that, given a good family routers, we can build a good tree-of-sets system. The proof of the first part is very similar to the argument in [6]; due to space constraints we omit discussion of this here. We sketch the second part.

Suppose we are given a good family $\mathcal{R} = \{S_1, \ldots, S_{r_0}\}$ of routers. We now give a high-level description of an algorithm to construct a good tree-of-sets system from $\mathcal{R}$. The algorithm consists of two phases. We start with the first phase.

Since every set $S_i \in \mathcal{R}$ can send $h_0/2$ flow units to the terminals with no edge-congestion, and the terminals are 1-well-linked in G, it is easy to see that every pair $S_i, S_j \in \mathcal{R}$ of sets can send $h_0/2$ flow units to each other with edge-congestion at most 3, and so there are at least $\frac{h_0}{6\Delta}$ node-disjoint paths connecting S_i to S_j. We build an auxiliary graph H, by contracting each cluster $S_i \in \mathcal{R}$ into a super-node v_i. We view the super-nodes $v_1, \ldots, v_{r_0}$ as the terminals of H, and denote $\tilde{\mathcal{T}} = \{v_1, \ldots, v_{r_0}\}$. We then use known connectivity-preserving reduction procedures in graph H repeatedly, to obtain a new graph H', whose vertex set is $\tilde{\mathcal{T}}$, every pair of vertices remains $\frac{h_0}{\text{poly}(\Delta)}$-edge-connected, and every edge $e = (v_i, v_j) \in E(H')$ corresponds to a path P_e in G, connecting a vertex of S_i to a vertex of S_j. Moreover, the paths $\{P_e \mid e \in E(H')\}$ are node-disjoint, and they do not contain the vertices of $\bigcup_{S\in\mathcal{R}} S$ as inner vertices. More specifically, graph H is obtained from H' by first performing a sequence of edge contraction and edge deletion steps that preserve element-connectivity of the terminals [14], and then performing edge-splitting steps that preserves edge-connectivity [20]. Let Z be a graph whose vertex set is $\tilde{\mathcal{T}}$, and there is an edge (v_i, v_j) in Z iff there are many (say $\frac{h_0}{r_0^2\,\text{poly}(\Delta)}$) parallel edges (v_i, v_j) in H'. We show that Z is a connected graph, and so we can find a spanning tree T of Z. Since $r_0 = r^2$, either T contains a path of length r, or it contains at least r leaves. Consider the first case, where T contains a path P of length r. We can use the path P to define a tree-of-sets system (in fact, it will give a path-of-sets system directly, after we apply Theorem 2.2 to boost well-linkedness inside the clusters that participate in P, and Corollary 2.2 to ensure the linkedness of the corresponding vertex subsets inside each cluster). From now on, we focus on the second case, where T contains r leaves. Assume without loss of generality that the good routers that are associated with the leaves of T are $\mathcal{R}' = \{S_1, \ldots, S_r\}$. We show that we can find, for each $1 \leq i \leq r$, a subset $E_i \subset \text{out}_G(S_i)$ of $h_3 = h\,\text{poly}(r \cdot \Delta)$ edges, such that for each pair $1 \leq i < j \leq r$, there are h_3 node-disjoint paths connecting S_i to S_j in G, where each path starts with an edge of E_i and ends with an edge of E_j. In order to compute the sets E_i of edges, we show that we can simultaneously connect each set S_i to the set $S^* \in \mathcal{R}$ corresponding to the root of tree T with many paths. For each i, let $\mathcal{P}_i$ be the collection of paths connecting S_i to S^*. We will ensure that all paths in $\bigcup_i \mathcal{P}_i$ are node-disjoint. The existence of the sets $\mathcal{P}_i$ of paths follows from the fact that all sets S_i can simultaneously send large amounts of flow to S^* (along the leaf-to-root paths in the tree T) with relatively small congestion. After boosting the well-linkedness of the endpoints of these paths in S^* using Theorem 2.2 for each $\mathcal{P}_i$ separately, and ensuring that, for every pair $\mathcal{P}_i, \mathcal{P}_j$ of such path sets, their endpoints are linked inside S^* using Corollary 2.2, we obtain somewhat smaller subsets $\mathcal{P}'_i \subset \mathcal{P}_i$ of paths for each i. The desired set E_i of edges is obtained by taking the first edge on every path in $\mathcal{P}'_i$. We now proceed to the second phase.

The execution of the second phase is very similar to the execution of the first phase, except that the initial graph H is built slightly differently. We will ignore the clusters in $\mathcal{R}\setminus\mathcal{R}'$. For each cluster $S_i \in \mathcal{R}'$, we delete all edges in $\text{out}_G(S_i) \setminus E_i$ from G, and then contract the vertices of S_i into a super-node v_i. As before, we consider the set $\tilde{\mathcal{T}} = \{v_1, \ldots, v_r\}$ of supernodes to be the terminals of the resulting graph $\tilde{H}$. Observe that now the degree of every terminal v_i is exactly

h_3, and the edge-connectivity between every pair of terminals is also exactly h_3. It is this additional property that allows us to build the tree-of-sets system in this phase. As before, we perform standard splitting operations to reduce graph $\tilde{H}$ to a new graph $\tilde{H}'$, whose vertex set is $\tilde{\mathcal{T}}$. As before, every edge $e = (v_i, v_j)$ in $\tilde{H}'$ corresponds to a path P_e connecting a vertex of S_i to a vertex of S_j in G; all paths in $\left\{P_e \mid e \in E(\tilde{H}')\right\}$ are node-disjoint, and they do not contain the vertices of $\bigcup_{S\in\mathcal{R}'} S$ as inner vertices. However, we now have the additional property that the degree of every vertex v_i in $\tilde{H}'$ is h_3, and the edge-connectivity of every pair of vertices is also h_3. We build a graph $\tilde{Z}$ on the set $\tilde{\mathcal{T}}$ of vertices as follows: for every pair (v_i, v_j) of vertices, if there number of edges (v_i, v_j) in $\tilde{H}'$ is $n_{i,j} > h_3/r^3$, then we add $n_{i,j}$ parallel edges (v_i, v_j) to $\tilde{Z}$. Otherwise, if $n_{i,j} < h_3/r^3$, then we do not add any edge connecting v_i to v_j. We then show that the degree of every vertex in $\tilde{Z}$ remains very close to h_3, and the same holds for edge-connectivity of every pair of vertices in $\tilde{Z}$. Note that every pair v_i, v_j of vertices of $\tilde{Z}$ is either connected by many parallel edges, or there is no edge (v_i, v_j) in $\tilde{Z}$. In the final step, we show that we can construct a spanning tree of $\tilde{Z}$ with maximum vertex degree bounded by 3. This spanning tree immediately defines a good tree-of-sets system. A way of seeing that graph $\tilde{Z}$ has a spanning tree of degree at most 3 is to observe that graph $\tilde{Z}$ is 1-tough (that is, if we remove q vertices from $\tilde{Z}$, there are at most q connected components in the resulting graph, for any q). It is known that any 1-tough graph has a spanning tree of degree at most 3 [25]. For algorithmic purposes, we use a different proof and construct the desired spanning tree using the algorithm of Singh and Lau [24].

Acknowledgment. We thank Tasos Sidiropoulos for suggesting the problem of improving the bounds in the grid-minor theorem, and in particular for asking whether the ideas in routing algorithms are useful for this purpose.

6. REFERENCES

[1] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[2] Chandra Chekuri and Julia Chuzhoy. Large-treewidth graph decompositions and applications. In *Proc. of ACM STOC*, pages 291–300, 2013.

[3] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *CoRR*, abs/1305.6577, 2013.

[4] Chandra Chekuri and Alina Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proc. of ACM-SIAM SODA*, 2013.

[5] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. The all-or-nothing multicommodity flow problem. *SIAM Journal on Computing*, 42(4):1467–1493, 2013.

[6] Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *Proc. of ACM STOC*, pages 855–874, 2012.

[7] Julia Chuzhoy and Shi Li. A polylogarithimic approximation algorithm for edge-disjoint paths with congestion 2. In *Proc. of IEEE FOCS*, 2012.

[8] M. Conforti, R. Hassin, and R. Ravi. Reconstructing edge-disjoint paths. *Operations Research Letters*, 31(4):273–276, 2003.

[9] Erik Demaine, MohammadTaghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Improved grid minor bounds and wagnerâĂŹs contraction. *Algorithmica*, 54:142–180, 2009.

[10] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and *h*-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

[11] Erik D. Demaine and MohammadTaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.

[12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[13] Reinhard Diestel, Tommy R. Jensen, Konstantin Yu. Gorbunov, and Carsten Thomassen. Highly connected sets and the excluded grid theorem. *J. Comb. Theory, Ser. B*, 75(1):61–73, 1999.

[14] H. R. Hind and O. Oellermann. Menger-type results for three or more vertices. *Congressus Numerantium*, 113:179–204, 1996.

[15] K. Kawarabayashi and Y. Kobayashi. Linear min-max relation between the treewidth of H-minor-free graphs and its largest grid minor. In *Proc. of STACS*, 2012.

[16] Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. *J. ACM*, 56(4):19:1–19:15, July 2009.

[17] Stephan Kreutzer and Siamak Tazari. On brambles, grid-like minors, and parameterized intractability of monadic second-order logic. In *Proc. of ACM-SIAM SODA*, pages 354–364, 2010.

[18] Alexander Leaf and Paul Seymour. Treewidth and planar minors. Manuscript, available at https://web.math.princeton.edu/~pds/papers/treewidth/paper.pdf, 2012.

[19] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 46:787–832, 1999.

[20] W. Mader. A reduction method for edge connectivity in graphs. *Ann. Discrete Math.*, 3:145–164, 1978.

[21] Bruce Reed. *Surveys in Combinatorics*, chapter Treewidth and Tangles: A New Connectivity Measure and Some Applications. Cambridge University Press, 1997.

[22] N Robertson, P Seymour, and R Thomas. Quickly Excluding a Planar Graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, November 1994.

[23] Neil Robertson and P D Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, August 1986.

[24] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 661–670. ACM, 2007.

[25] Sein Win. On a connection between the existence of k-trees and the toughness of a graph. *Graphs and Combinatorics*, 5:201–205, 1989.