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 [22] who showed that $f(k) = \Omega(\log^2 k)$; 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 $f(k) = \Omega(\log k / \log \log k)$. In this paper we obtain the first polynomial relationship between treewidth and grid-minor size by showing that $f(k) = \Omega(k^3)$ 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 Robertson 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 $f(k)$ in the original proof of Robertson and Seymour [23] was substantially improved by Robertson, Seymour and Thomas [22] who showed that $f(k) = \Omega(\log^{1/3} k)$; 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 $f(k) = \Omega(\log k / \log \log k)$. 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 $f(k) = \Omega(\sqrt{k})$. This can be slightly improved to $f(k) = \Omega(\sqrt{k} / \log k)$ 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 [17]. 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 $\delta$ is at least $\frac{1}{27} - o(1)$ 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 $k' = O(|V(H)|^{1/2})$ [23].

1 The relationship between grid-minors and treewidth is much tighter in some special classes of graphs. In planar graphs $f(k) = \Omega(k)$ [23]; a similar linear relationship is known in bounded-genus graphs [10] and graphs that exclude a fixed graph $H$ as a minor [14] (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 $P_1, \ldots, P_{r-1}$ that are disjoint, which connect the sets in a path-like fashion. The number of paths in each set $P_i$ is $h$. Moreover, for each $i$, the induced graph $G[S_i]$ satisfies the following routing properties for the endpoints of the paths $P_{i-1}$ and $P_i$ (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 $|A|$ edge-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 $\Omega(h^{1/2} \times h^{1/2})$ in $G$, slightly strengthening a similar recent result of Leaf and Seymour [18], who use a related combinatorial object that they call an $(h, h)$-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 $h^2 \leq k/\text{polylog}(k)$, 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 [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. 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 page breaking 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 $out_G(A)$ 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 endpoint 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 $P$ of paths in $G$, we denote by $V(P)$ the set of all vertices participating in paths in $P$, and similarly, $E(P)$ is the set of all edges that participate in paths in $P$. 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 $\ell, p$. Then $T$ has at least $\ell$ 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 $\{X_v \subseteq V \mid v \in V(T)\}$ 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 $G$, if $H$ can be obtained from $G$ by a sequence of edge deletion and contraction operations.

Equivalently, $H$ is a minor of $G$ if there is a map $\varphi : V(H) \to V(G)$ 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 $\varphi$ satisfying these conditions is called a model of $H$ in $G$. We say that $G$ contains a $(g \times g)$-grid minor if some minor $H$ of $G$ is isomorphic to the $(g \times g)$ grid.
**Sparsest Cut and the Flow-Cut Gap** Suppose we are given a graph $G = (V, E)$, and a subset $T \subseteq V$ of $k$ terminals. The sparsity of a cut $(S, \overline{S})$ with respect to $T$ is $\Phi(S) = \frac{|E(S, \overline{S})|}{\min\{|S\cap T|, |\overline{S}\cap T|^\alpha\}}$, and the value of the sparsest cut in $G$ is defined to be: $\Phi(G) = \min_{SCV} \{\Phi(S)\}$. The goal of the sparsest cut problem is, given an input graph $G$ and a set $T$ of terminals, to find a cut of minimum sparsity.

**Theorem 2.1** Let $G = (V, E)$ be any graph, and let $T$ be any subset of vertices of $G$ called terminals, such that $T$ is $\alpha$-well-linked in $G$. Let $E'$ be any subset of $n'$ edges of $G$, such that $n' < \frac{n^2}{4\Delta}$. Then there is a connected component $C$ of $G \setminus E'$, containing at least $|T| - \frac{n'^2}{n^2}$ terminals.

**Corollary 2.1** Let $G$ be a graph with maximum vertex degree $\Delta$, and let $T$ be any subset of vertices of $G$ such that $T$ is $\alpha$-well-linked in $G$. Let $X$ be any subset of $n'$ vertices of $G$, with $n' < \frac{n^2}{4\Delta^2}$. Then there is a connected component $C$ of $G \setminus X$ containing at least $|T| - \frac{n'^2}{\alpha n}$ vertices of $T$.

**Definition 2.2** We say that a set $T$ of vertices is $\alpha$-well-linked in $G$, iff for any pair $(T_1, T_2)$ of equal-sized subsets of $T$, there is a collection $P$ of $|T_1|$ node-disjoint paths, connecting the vertices of $T_1$ to the vertices of $T_2$. (Note that $T_1$, $T_2$ 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 $T$ be any subset of vertices of $G$ called terminals, such that $T$ is $\alpha$-well-linked in $G$. Let $E'$ be any subset of $n'$ edges of $G$, such that $n' < \frac{n^2}{4\Delta}$. Then there is a connected component $C$ of $G \setminus E'$, containing at least $|T| - \frac{n'^2}{n^2}$ terminals.
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 $\alpha$-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 $\alpha$. A standard spanning-tree based clustering scheme of $[6]$ 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 $\mathcal{T}' \subseteq \mathcal{T}$, such that $\mathcal{T}'$ is node-well-linked in $G$. In order to achieve this, Lemma 2.1 guarantees that any graph $G$ is node-well-linked in $G$, for some $\alpha < 1$, and $\kappa \geq 8\Delta/\alpha$. Then there is a subset $\mathcal{T}' \subseteq \mathcal{T}$ of $\Omega(\frac{\Delta}{\alpha^2})$ 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 $\mathcal{T}' \subseteq \mathcal{T}$ of $\Omega\left(\frac{\alpha}{\Delta \beta_{\text{ARV}}(\kappa)} \cdot \kappa\right)$ 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 $\left\lceil \frac{1}{\alpha} \right\rceil$ 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 $\Delta$, and $r$ disjoint subsets $T_1, \ldots, T_r$ of vertices called terminals, such that $\mathcal{T} = \bigcup_j T_j$ is $\alpha$-well-linked in $G$, and $|T_j| \geq \kappa$ for all $j$. We would like to select, for each $1 \leq j \leq r$, a large subset $T''_j \subseteq T_j$ of terminals, such that $T''_j$ is node-well-linked in $G$, and for every pair $1 \leq j \neq j' \leq r$, $T''_j$, $T''_{j'}$ 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.

**Theorem 2.2** Suppose we are given a connected graph $G = (V, E)$ with maximum vertex degree $\Delta$, and a subset $\mathcal{T}$ of $\kappa$ vertices called terminals, such that $\mathcal{T}$ is $\alpha$-well-linked in $G$, for some $\alpha < 1$, and $\kappa \geq 8\Delta/\alpha$. Then there is a subset $\mathcal{T}' \subseteq \mathcal{T}$ of $\Omega(\frac{\Delta}{\alpha^2})$ 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 $\mathcal{T}' \subseteq \mathcal{T}$ of $\Omega\left(\frac{\alpha}{\Delta \beta_{\text{ARV}}(\kappa)} \cdot \kappa\right)$ 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 $\left\lceil \frac{1}{\alpha} \right\rceil$ terminals of $\mathcal{T}$, with $t \in V(T_t)$, such that all trees $\{T_t\}_{t \in \mathcal{T}'}$ are pairwise node-disjoint.

**Corollary 2.2** Let $G$ be a connected graph, with maximum vertex degree at most $\Delta$, and let $T_1, \ldots, T_r$ be a collection of disjoint vertex subsets, called terminals, such that $\mathcal{T} = \bigcup_j T_j$ is $\alpha$-well-linked in $G$, and $|T_j| \geq \kappa$ for all $j$, where $\kappa \geq 64\Delta^4 \beta_{\text{ARV}}(\kappa)/\alpha$. Assume that we apply Theorem 2.2 to each set $T_j$ of terminals in turn independently, and let $T''_j$ be the outcome of Theorem 2.2 when applied to $T_j$. Let $T''_j \subseteq T''_j$ be any subset of $\bigcup_j T''_j = \Omega\left(\frac{\alpha}{\Delta \beta_{\text{ARV}}(\kappa)} \cdot \kappa\right)$ terminals. Then for each $1 \leq j \leq r$, $T''_j$ is node-well-linked in $G$, and for all $1 \leq i \neq j \leq r$, $T''_i$ and $T''_j$ are linked in $G$.

**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 $\Omega(k/\text{poly log} k)$ 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 $[17]$, 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 $X, Y$ be two sets of directed simple paths in $G$, where all paths in $X \cup Y$ share the same destination vertex $s$. The paths in $X$ are disjoint from each other, except for sharing the destination $s$, and the same is true for $Y$ (but a vertex $v \neq s$ may appear on a path in $X$ and on a path in $Y$.) Let $H$ be the graph obtained by the union of the paths in $X \cup Y$. Then we can efficiently find a subset $X' \subseteq X$ of at least $|X| - |Y|$ paths, and for each path $Q \in Y$, a path $\hat{Q}$ in graph $H$, with the same endpoints as $Q$, such that, if we denote $Y' = \{\hat{Q} | Q \in Y\}$, then all paths in $X' \cup Y'$ 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 $S = \{S_1, \ldots, S_r\}$ of disjoint vertex subsets of $V(G)$. Let $S_i, S_j \in S$ be any two such subsets. We say that a path $P$ connects $S_i$ to $S_j$ if 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 $\bigcup_{S \in S} S$ as inner vertices.

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

- A sequence $S = \{S_1, \ldots, S_r\}$ of $r$ disjoint vertex subsets of $G$, where for each $i$, $G[S_i]$ is connected;
- For each $1 \leq i < r$, a set $P_i$ of $h$ disjoint paths, connecting $S_i$ to $S_{i+1}$ directly (that is, paths in $P_i$ do not contain the vertices of $\bigcup_{S \in S} S$ as inner vertices), such that all paths in $\bigcup_i P_i$ are mutually disjoint; and has the following additional property. For each $1 \leq i < r$, let $B_i$ be the set of vertices of $P_i$ that belong to $S_i$, and let $A_{i+1}$ be the set of vertices of $P_i$ 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 $[7]$.)
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 \(r = \Omega(g^3)\), finds a \((g \times g)\)-grid minor in the grill. 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 \(v_1, \ldots, v_r\) correspond to the sets \(S_1, \ldots, S_r\), and each edge \(e = (v_i, v_{i+1})\) corresponds to the collection \(P_i\) 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 [2], 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 \(r, h\) consists of:

- A collection \(S = \{S_1, \ldots, S_r\}\) 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 \(P^*_e\) of \(h\) disjoint paths, connecting \(S_i\) to \(S_j\) directly (that is, paths in \(P^*_e\) do not contain the vertices of \(\bigcup_{S \in S} S\) as inner vertices). Moreover, all paths in \(P^*_e = \bigcup_{P \in E(T)} P^*_e\) are pairwise disjoint.
- For a set \(S_i \in S\), 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 \(P^*_e\), that belong to \(S_i\). We then have:
- For each \(S_i \in S\), 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 theorem expands and generalizes the results of [7].

**Theorem 4.1** Suppose we are given a graph \(G\) of maximum vertex degree \(\Delta = O(\log^2 k)\), and a subset \(T\) of \(k\) vertices called terminals, such that \(T\) is node-well-linked in \(G\) and the degree of every vertex in \(T\) is 1. Additionally, assume that we are given parameters \(r > 1, h > 4\log k\), such that \(k/\log^k k > c'h^{r^{2r}}\Delta^\alpha\), 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 \((S, T, \bigcup_{P \in E(T)} P^*_e)\) in \(G\), with parameters \([r], [h]\). Moreover, for all \(v_i \in S, S_i \cap T = \emptyset\).

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 \((S, T, \bigcup_{P \in E(T)} P^*_e)\) with parameters \(r, h\), and integers \(h^*, r^*\), such that \((r^*)^2 \leq r\) and \(h > 16h^* (r^*)^2 + 1\), 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 \(G = (V, E)\) of 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 \(\Omega(k/\log k)\) vertices, such that \(X\) is node-well-linked in \(G'\). We add a new set \(T\) 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 \(|T|\) by \(k\), and we refer to the vertices of \(T\) 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 \(T\) is node-well-linked in \(G\). We then apply Theorem 4.1 to \(G\) and \(T\) to obtain a strong tree-of-sets system \((S,T,\bigcup_{P \in E(T)} P^*_e)\), with parameters \(r = k^{1/49}/\log^2 k, h = k^{1/49}\).

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 \(h^* = \lceil k^{1/49} \rceil\) 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 6.1 to construct a grid minor of size \(\Omega(\sqrt{h^*}) \times \Omega(\sqrt{h^*})\). Notice that \(h^* = \Theta(k^{1/49}) = \Theta(k^{1/49} / \log k)\), and the the size of the grid minor computed by the algorithm is \(\Omega(\frac{k^{1/98}}{\log k} \times \frac{k^{1/98}}{\log k})\).

The rest of this section is devoted to proving Theorem 4.2. Let \((S, T, \bigcup_{P \in E(T)} P^*_e)\) be a strong tree-of-set system with parameters \(r\) and \(h\). Let \(h^*, r^*\) be integers such that \((r^*)^2 \leq r\) and \(h > 16h^* (r^*)^2 + 1\).

For convenience, for each set \(S_i \in S\), we denote the corresponding vertex of \(T\) by \(v_{S_i}\). 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 \(h \geq h^*\) and width \(r^*\).

Otherwise, since \(|V(T)| = \frac{r}{(r^*)^2}\), \(T\) 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 \(L = \{S \in S \mid v_{S} \in L\}\) 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 \(L\).
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 $S$ in the same order — the order of the tour. The clusters in $L$ will then serve as the sets $S$ in our final path-of-systems set, and the collection of $h^*$ paths that we build will be used for the paths $P_i$. In order for this to work, we need to route up to three sets of paths across clusters $S \in S$: 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 $L$, along the root-to-leaf paths in $T$. In the second phase, we build the path-of-systems 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 $S \in S$, and the sets $P^*_e$ 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 $S \in L$, a collection $Q_S$ of $[h/r^*]$ paths, that have the following properties:

- Each path $Q \in Q_S$ 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 $Q \in Q_S$, 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 $Q = \bigcup_{S \in L} Q_S$ 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 $Q_S$ 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 $S$.

**Proof.** For each cluster $S' \in S$, 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 connecting $v_{S'}$ to its parent. Let $\Gamma' = \Gamma_{S'}^e$ be the set of vertices of $S'$ that serve as endpoints of the paths in $P_{S'}^*$, we process the tree in top to bottom order, while maintaining a set $Q$ 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 $Q_{S'} \subseteq Q$ of $n(S') \cdot \lfloor h/r^* \rfloor$ paths connecting $S^*$ to $S'$ in $Q$. Each such path does not share vertices with $S'$, except for its last vertex, which must belong to $\Gamma_{S'}$. Moreover, for every path $Q \in Q_{S'}$, 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 $Q_S$ be any subset of $n(S') \cdot \lfloor h/r^* \rfloor$ paths of $P^*_e$, and we set $Q = Q_S$. (Notice that $|L| \cdot \lfloor h/r^* \rfloor \leq h = |P^*_e|$, 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 $Q_S \subseteq Q$ 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 $\Gamma_1 \subseteq \Gamma_{S'(e_1)}$ of $n(S') \cdot \lfloor h/r^* \rfloor$ vertices and a subset $\Gamma_2 \subseteq \Gamma_{S'(e_2)}$ of $n(S'') \cdot \lfloor h/r^* \rfloor$ vertices, together with a set $R$ of $|\Gamma'|$ disjoint paths contained inside $G[S]$, where each path connects a vertex of $\Gamma'$ 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 $\Gamma'$ to $t$ with a directed edge. Add a new vertex $s_1$ of capacity $n(S') \cdot \lfloor h/r^* \rfloor$ and connect it with a directed edge to every vertex of $\Gamma_{S'(e_1)}$. Similarly, add a new vertex $s_2$ of capacity $n(S'') \cdot \lfloor h/r^* \rfloor$ 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 easy to show that there is an $s$-$t$ flow of value $|\Gamma'| = n(S') \cdot \lfloor h/r^* \rfloor + n(S'') \cdot \lfloor h/r^* \rfloor$ in this flow network. Since $\Gamma'$ and $\Gamma_{S'(e_1)}$ are linked, there is a set $P_1$ of $|\Gamma'|$ disjoint paths connecting the vertices of $\Gamma'$ 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 $P_2$ of $|\Gamma'|$ disjoint paths connecting vertices of $\Gamma'$ 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.

Let $P_1 \subseteq P^*(e_1)$ be a subset of paths whose endpoints belong to $\Gamma_1$, and define $P_2 \subseteq P^*(e_2)$ similarly for $\Gamma_2$. Concatenating the paths in $Q_S$, $R$, and $P_1 \cup P_2$, we obtain two collections of paths: set $Q_{S'}$ of $n(S') \cdot \lfloor h/r^* \rfloor$ paths, connecting $S^*$ to $S'$, and set $Q_{S''}$ of $n(S'') \cdot \lfloor h/r^* \rfloor$ paths, connecting $S^*$ to $S''$, that have the desired properties. We delete the paths of $Q_S$ from $Q$, and add the paths in $Q_{S'}$ and $Q_{S''}$ 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 \( \{ Q_S \mid S \in \mathcal{L} \} \) 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 \( L_S \subseteq \mathcal{L} \) to be the set of all descendants of \( v_S \) that belong to \( L \), and \( E_S = \{ S' \mid v_{S'} \in L_S \} \) 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 \( \ell_S \) 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 \( \lfloor \frac{h}{10r} \rfloor - 8\ell_S \cdot h^* \), 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 \( r^* \), \( \lfloor \frac{h}{10r} \rfloor - 8\ell_S \cdot h^* \geq \frac{h}{10r} - 8r^* \cdot h^* > 0 \) where the latter inequality follows from the assumption that \( h > 16h^*(r^*)^2 + 1 \).

**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 \( S = \{ S \} \), with \( A = B = S \). We let \( \mathcal{Q}_A, \mathcal{Q}_B \) be any pair of disjoint subsets of \( Q_S \) containing \( \lfloor \frac{h}{10r} \rfloor \) 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: \((\mathcal{L}_{S'}, \mathcal{P}_{S'})\) 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 \( \Gamma = \{ v_Q \mid q \in \mathcal{Q}_{A_1} \} \). We similarly define \( \Gamma_2 \), \( \Gamma_3 \), \( \Gamma_4 \) for \( \mathcal{Q}_{B_1}, \mathcal{Q}_{A_2}, \) and \( \mathcal{Q}_{B_2} \), respectively. Denote \( \Gamma = (1 \cup \Gamma_2, \Gamma_3 \cup \Gamma_4) \), and \( \Gamma' = (1 \cup \Gamma_2, \Gamma_3 \cup \Gamma_4) \). 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)).

**Figure 3: Processing a degree-3 vertex \( v_S \).**

Our goal is to find a set \( \mathcal{R} \) of \( 4h^* \) disjoint paths inside \( S \) connecting \( \Gamma \) to \( \Gamma' \), such that the paths in \( \mathcal{R} \) intersect at
most 8h' paths in \( P \), and at most 8h' paths in \( P' \). Since sets \( \Gamma, \Gamma' \) are linked in \( G'[S] \), we can find a set \( R \) of 4h' disjoint paths inside \( S \) connecting \( \Gamma \) to \( \Gamma' \), however these paths may intersect many paths in \( P \cup P' \). We start from an arbitrary set \( R \) of 4h' disjoint paths connecting \( \Gamma \) to \( \Gamma' \) inside \( S \). We next re-route these paths, using Lemma 2.2.

We apply Lemma 2.2 twice. First, we unify all vertices of \( \Gamma \) into a single vertex \( s \), and direct all paths in \( P \subset P' \cap R' \) be the two resulting sets of paths. We discard from \( P \) paths that share endpoints with paths in \( R' \) (at most \(|R'| \) paths). Then \(|P| = |P| - 2|R| = |P| - 8h' \), and \( R' \) contains 4h' disjoint paths connecting vertices in \( \Gamma \) to vertices in \( \Gamma' \).

Moreover, the paths in \( P \cup R' \) are completely disjoint. Next, we unify all vertices in \( \Gamma' \) into a single vertex \( s \), and direct all paths in \( P' \) and \( R'' \) towards \( s \). We then apply Lemma 2.2 to the two resulting sets of paths, with \( P' \) serving as \( \mathcal{X} \) and \( R'' \) serving as \( \mathcal{Y} \). This finishes the proof.

**5. FINDING A TREE-OF-SETS SYSTEM**

In this section we prove Theorem 1.1. 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-sets 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 \( \alpha \)-bandwidth property in \( G \) iff its interface \( \Gamma \) is \( \alpha \)-well-linked in \( G[S] \).

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

- A collection \( S = \{ S_1, \ldots, S_r \} \) 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 \( P_e \) of \( h \) disjoint paths, connecting \( S_i, S_j \) directly (that is, paths in \( P_e \) do not contain the vertices of \( \bigcup_{S \in \mathcal{S}} S \) as inner vertices).

Moreover, all paths in \( P = \bigcup_{e \in E(T)} P_e \) 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 \( \bigcup_{e \in E(T)} P_e \). Then each \( S_i \in \mathcal{S} \) has the \( \alpha_{BW} \)-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 \( \Delta \), and a weak tree-of-sets system \((S,T, \bigcup_{e \in E(T)} P_e)\) with parameters \( r,h,\alpha_{BW} \) in \( G \), computes a strong tree-of-sets system

\[(S',T, \bigcup_{e \in E(T)} P'_e)\]

with parameters \( r,h \), such that

\[P'_e \subset P_e \quad \text{for each} \quad e \in E(T)\]

\[h = \Omega(\frac{\alpha_{BW}^2}{2^\Delta S_{\Delta ARV}(k)}), \quad h\]

**Proof.** We prove the lemma by boosting well-linkedness using the claims in Section 2. Consider the given weak tree-of-sets system \((S,T, \bigcup_{e \in E(T)} P_e)\) in \( G \). Let \( S_i \in \mathcal{S} \). Consider some edge \( e = (v_i,v_j) \) of \( T \). Let \( \Gamma_{1} \) be the set of endpoints of the paths in \( P_e \) that belong to \( S_i \), and recall that \( \Gamma_{1} \) is \( \alpha_{BW} \)-well-linked in \( G[S_i] \). We apply Theorem 2.2 to \( S_i, \Gamma_{1}, \) and obtain to obtain a subset \( \Gamma'_{1} \subseteq \Gamma_{1} \) of vertices, such that \( \Gamma'_{1} \) is node-well-linked in \( G[S_i] \). Let \( P'_e \subset P_e \) be a subset of paths whose endpoint belongs to \( \Gamma'_{1} \), so \( |P'_e| = \Theta \left( \frac{\alpha_{BW}}{S_{\Delta ARV}(k)} \right) \). Let \( \Gamma_{2} \) be the set of endpoints of the paths in \( P'_e \) that belong to \( S_j \). We apply Theorem 2.2 to \( S_j \) and \( \Gamma_{2} \) to obtain a subset \( \Gamma'_{2} \subseteq \Gamma_{2} \) of vertices, such that \( \Gamma'_{2} \) is node-well-linked in \( S_j \). Let \( P''_e \subset P'_e \) be a subset of all paths whose endpoint belongs to \( \Gamma'_{2} \). Finally, we select an arbitrary subset \( P^*_e \) of \( h \) paths: \( |P^*_e| = \Theta \left( \frac{\alpha_{BW}}{S_{\Delta ARV}(k)^2} \right) \) paths. We denote by \( \Gamma_{S_i}(e) \) the endpoints of the paths in \( P^*_e \) that belong to \( S_i \), and we denote by \( \Gamma_{S_j}(e) \) the endpoints of the paths in \( P^*_e \) 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 \( G[S_i] \).

The following theorem is the main result of this section.

**Theorem 4.4**
Theorem 5.1 Suppose we are given a graph $G$ of maximum vertex degree $\Delta = O(\log^3 k)$, and a subset $T$ of $k$ vertices called terminals, such that $T$ is node-wise-linked in $G$ and the degree of every vertex in $T$ is 1. Additionally, assume that we are given any parameters $r > 1, h > 4 \log k$, such that $k/\log^2 k > ch^{19} \Delta^8$, 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 $(S, T, \bigcup_{e \in \mathcal{E}(T)} \mathcal{P}_e)$ in $G$, with parameters $[h], [r]$ and $\alpha_{BW} = \Omega \left( \frac{1}{\log^2 k} \right)$. Moreover, for all $S \subseteq S, S \cap T = \emptyset$.

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

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 $(S, T, \bigcup_{e \in \mathcal{E}(T)} \mathcal{P}_e)$ is a weak tree-of-sets system in $G$, with parameters $h, r, \alpha_{BW}$ as in the theorem statement, and for each $S_i \subseteq S, S_i \cap T = \emptyset$, 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 if the following three conditions hold: (1) $S \cap T = \emptyset$; (2) $S$ has the $\alpha_{BW}$-bandwidth property; and (3) $S$ can send a large amount of flow (say at least $h_0/2$ flow units) to $T$ 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 $\mathcal{R} = \{S_1, \ldots, S_n\}$ 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 $\frac{h_0}{3\Delta}$ 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 supernode $v_i$. We view the super-nodes $v_1, \ldots, v_n$ as the terminals of $H$, and denote $\mathcal{T} = \{v_1, \ldots, v_n\}$. We then use known connectivity-preserving reduction procedures in graph $H$ repeatedly, to obtain a new graph $H'$, whose vertex set is $T$, every pair of vertices remains 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 $\bigcup_{e \in \mathcal{E}} S$ 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 [11], and then performing edge-splitting steps that preserves edge-connectivity [20]. Let $Z$ be a graph whose vertex set is $T$, and there is an edge $(v_i, v_j)$ in $Z$ iff there are many (say $\frac{h_0}{3\Delta}$) 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 $T' = \{S_1, \ldots, S_r\}$. We show that we can find, for each $1 \leq i \leq r$, a subset $E_i \subseteq \mathcal{O}(S_i)$ of $h_0 = h \cdot \text{poly}(r \cdot \Delta)$ edges, such that for each pair $1 \leq i < j \leq r$, there are $h_0$ 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 $P_i$ be the collection of paths connecting $S_i$ to $S'$. We will ensure that all paths in $\bigcup_{i} P_i$ are node-disjoint. The existence of the sets $P_i$ 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 $P_i$ separately, and ensuring that, for every pair $P_i, P_j$ of such path sets, their endpoints are linked inside $S'$ using Corollary 2.2, we obtain somewhat smaller subsets $P_i' \subseteq P_i$ of paths for each $i$. The desired set $E_i$ of edges is obtained by taking the first edge on every path in $P_i'$. 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 $\mathcal{R} \setminus T'$. For each cluster $S_i \in \mathcal{R}$, we delete all edges in $\mathcal{O}(S_i) \setminus E_i$ from $G$, and then contract the vertices of $S_i$ into a supernode $v_i$. As before, we consider the set $\mathcal{P} = \{v_1, \ldots, v_n\}$ of supernodes to be the terminals of the resulting graph $\tilde{H}$. Observe that now the degree of every terminal $v_i$ is exactly
h₃, and the edge-connectivity between every pair of terminals is also exactly h₃. 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 H to a new graph H′, whose vertex set is ℋ. As before, every edge e = (vᵢ, vⱼ) in H′ corresponds to a path Pᵢ, connecting a vertex of Sᵢ to a vertex of Sⱼ in G; all paths in \( \{ Pᵢ \mid e ∈ E(H') \} \) are node-disjoint, and they do not contain the vertices of \( \bigcup_{S ∈ R} S \) as inner vertices. However, we now have the additional property that the degree of every vertex vᵢ in H′ is h₃, and the edge-connectivity of every pair of vertices is also h₃. We build a graph ˜Z on the set ℋ of vertices as follows: for every pair (vᵢ, vⱼ) of vertices, if there is no edge (vᵢ, vⱼ) in H′, then we add \( nᵢ,j > h₃/r₃ \), and the same holds for edge-connectivity of every pair of vertices in ˜Z. Note that every pair vᵢ, vⱼ of vertices of ˜Z is either connected by many parallel edges, or there is no edge (vᵢ, vⱼ) in ˜Z. In the final step, we show that we can construct a spanning tree of ˜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 ˜Z has a spanning tree of degree at most 3 is to observe that every edge \( (vᵢ, vⱼ) \) of degree at most 3 [25].

### 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.

### REFERENCES


