Polynomial Bounds for the Grid-Minor Theorem

Chandra Chekuri∗ Julia Chuzhoy†

September 21, 2014

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 [KK12], and Leaf and Seymour [LS12], shows that $f(k) = \Omega(\sqrt{\log k / \log \log k})$. In contrast, the best known upper bound implies that $f(k) = O(\sqrt{k / \log k})$ [RST94]. 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 such a grid-minor.

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 $f$ given in the original proof of Robertson and Seymour [RS86] was substantially improved by Robertson, Seymour and Thomas [RST94] who showed that $f(k) = \Omega(\log^{1/6} k)$; see [DJGT99] and [Die12] for a simpler proof with a slightly weaker bound. There have been recent improvements by Kawarabayashi and Kobayashi [KK12], and by Leaf and Seymour [LS12], 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) [RST94]. Robertson et al. [RST94] suggest that this value may be sufficient, and Demaine et al. [DHK09] 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

∗Dept. of Computer Science, University of Illinois, Urbana, IL 61801. chekuri@illinois.edu. Supported in part by NSF grants CCF-1016684 & CCF-1319376, and by TTI Chicago during a sabbatical visit in Fall 2013.
†Toyota Technological Institute, Chicago, IL 60637. Email: cjulia@ttic.edu. Supported in part by NSF CAREER grant CCF-0844872, NSF grant CCF-1318242 and Sloan Research Fellowship.
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 \( k \) has a grid-minor of size \( \Omega(k^\delta) \times \Omega(k^\delta) \). Moreover, there is a randomized algorithm that, given \( G \), outputs a model of a grid-minor of size \( \Omega(k^\delta) \times \Omega(k^\delta) \), and whose running time is polynomial in \( |V(G)| \) and \( k \).

Our proof shows that \( \delta \) is at least \( 1/98 - o(1) \) 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 \( k' = O(|V(H)|) \) [RST94].

**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 \( h \) 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 \( P_i \) is \( h \). Moreover, for each \( i \), the induced graph \( G[S_i] \) satisfies the following routing properties for the end-points of the paths \( P_{i-1} \) 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 \( |A| \) 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 model of 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 [LS12], 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

---

1The relationship between grid-minors and treewidth is much tighter in some special classes of graphs. In planar graphs \( f(k) = \Omega(k) \) [RST94], a similar linear relationship is known in bounded-genus graphs [DFHT05] and graphs that exclude a fixed graph \( H \) as a minor [DH08] (see also [KK12]).
constant congestion \cite{Chu12,CL12}, and connections to treewidth \cite{CE13,CC13}. 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 \cite{Chu12,CL12,CE13}, 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.

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. We say that an event $\mathcal{E}$ holds with high probability, if the probability of $\mathcal{E}$ is at least $1 - 1/n^c$ for some constant $c > 1$, where $n$ is the number of the graph vertices. We use the following simple claim several times.

Claim 2.1 Let $\{x_1, \ldots, x_n\}$ be any set of non-negative integers, with $\sum_i x_i = N$, and $x_i \leq 2N/3$ for all $i$. Then we can efficiently compute a partition $(A,B)$ of $\{1,\ldots, n\}$, such that $\sum_{i \in A} x_i \geq N/3$ and $\sum_{i \in B} x_i \geq N/3$.

Proof: We assume without loss of generality that $x_1 \geq x_2 \geq \cdots \geq x_n$, and process the integers in this order. When $x_1$ is processed, we add $i$ to $A$ if $\sum_{j \in A} x_j \leq \sum_{j \in B} x_j$, and we add it to $B$ otherwise. We claim that at the end of this process, $\sum_{i \in A} x_i, \sum_{i \in B} x_i \geq N/3$. Indeed, if $x_1 \geq N/3$, then 1 is added to $A$, and, since $x_1 \leq 2N/3$, it is easy to see that both subsets of integers sum up to at least $N/3$. Otherwise, $|\sum_{i \in A} x_i - \sum_{i \in B} x_i| \leq \max_i \{x_i\} \leq x_1 \leq N/3$.

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

Claim 2.2 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 of length at least $p$.

Proof: Suppose $T$ has fewer than $\ell$ leaves and each root-to-leaf path has length less than $p$. Then, since each node is in some root to leaf path, $|V(T)| < \ell p$ which contradicts our assumption.

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 $G$. 

3
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:

- For each $v \in V(H)$, the sub-graph of $G$ induced by $\varphi(v)$ is connected;
- If $u, v \in V(H)$ and $u \neq v$, then $\varphi(u) \cap \varphi(v) = \emptyset$; and
- For each edge $e = (u, v) \in E(H)$, there is an edge in $E(G)$ with one endpoint in $\varphi(v)$ and the other endpoint in $\varphi(u)$.

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 iff some minor $H$ of $G$ is isomorphic to the $(g \times g)$ grid.

### 2.1 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 $\Phi(S) = \frac{|E(S, \overline{S})|}{\min \{|S \cap T|, |\overline{S} \cap T|\}}$, and the value of the sparsest cut in $G$ is defined to be: $\Phi(G) = \min_{S \subseteq V} \{\Phi(S)\}$. 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. Arora, Rao and Vazirani [ARV09] have shown an $O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We denote this algorithm by $A_{ARV}$, and its approximation factor by $\beta_{ARV}(k) = O(\sqrt{\log k})$.

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 $\lambda$, such that every pair of terminals can send $\lambda/k$ 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 $\lambda^*$ of the maximum concurrent flow for the terminals in the graph. The flow-cut gap in undirected graphs, that we denote by $\beta_{FCG}(k)$ throughout the paper, is known to be $\Theta(\log k)$ [LR99]. Therefore, if $\Phi(G) = \alpha$, then every pair of terminals can simultaneously send $\frac{\alpha}{k\beta_{FCG}(k)}$ 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 $\beta_{FCG}(k)/\alpha$. Moreover, given any matching $M$ on the set $T$ of terminals, one unit of flow for each pair in $M$ can be simultaneously routed with congestion at most $2\beta_{FCG}(k)/\alpha$.

### 2.2 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 $T$ of vertices is $\alpha$-well-linked\(^2\) in $G$, iff for any partition $(A, B)$ of the vertices of $G$ into two subsets, $|E(A, B)| \geq \alpha \cdot \min \{|A \cap T|, |B \cap T|\}$.

\(^2\)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-linked, and explicitly use the term node-well-linked when necessary.
Notice that if a set $T$ of terminals is $\alpha$-well-linked in $G$, then the value of the sparsest cut in $G$ with respect to $T$ is at least $\alpha$. Notice also that if a set $T$ of terminals is $\alpha$-well-linked in $G$, then so is any subset $T' \subseteq T$.

**Definition 2.2** We say that a set $T$ of vertices is node-well-linked in $G$, iff for any pair $(T_1, T_2)$ of equal-sized subsets of $T$, there is a collection $\mathcal{P}$ 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 empty paths).

Our algorithm proceeds by reducing the degree of the input graph to $\text{poly log}(k)$, while preserving the treewidth to within a factor of $\text{poly log}(k)$. As we show below, in bounded-degree graphs, the notions of edge- and node-well-linkedness are closely related to each other, and we exploit this connection throughout the algorithm.

We will repeatedly use the following simple claim, whose proof appears in the Appendix.

**Claim 2.3** 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 $n' < \frac{\alpha |T|}{\Delta}$. Then there is a connected component $C$ of $G \setminus E'$, containing at least $|T| - \frac{n'}{\alpha}$ terminals.

**Corollary 2.1** Let $G$ be any graph with maximum vertex degree at most $\Delta$, 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 $n' < \frac{\alpha |T|}{\Delta \Delta}$. Then there is a connected component $C$ of $G \setminus X$ containing at least $|T| - \frac{n' \Delta}{\alpha}$ vertices of $T$.

The corollary follows from Claim 2.3 by letting $E'$ be the set of all edges incident on the vertices of $X$.

**Definition 2.3** We say that 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 $G$.

**Theorem 2.1** Suppose we are given two disjoint subsets $T_1, T_2$ of vertices of $G$, with $|T_1|, |T_2| \geq \kappa$, such that $T_1 \cup T_2$ is $\alpha$-well-linked in $G$, and each one of the sets $T_1, T_2$ is node-well-linked in $G$. Let $T_1' \subset T_1, T_2' \subset T_2$, be any pair of subsets with $|T_1'| = |T_2'| \leq \frac{\alpha \kappa}{\Delta}$. Then $T_1'$ and $T_2'$ are linked in $G$.

**Proof:** If $|T_1| > \kappa$, then we discard from $T_1$ vertices that belong to $T_1 \setminus T_1'$, until $|T_1| = \kappa$ holds, and we do the same for $T_2$. Notice that $T_1 \cup T_2$ continues to be $\alpha$-well-linked, and each of the resulting sets $T_1, T_2$ is still node-well-linked.

Let $T = T_1 \cup T_2$. We refer to vertices in $T$ as terminals. Notice that $|T| = 2\kappa$. Assume for contradiction that $T_1', T_2'$ are not linked in $G$. Then there are two sets $A \subseteq T_1', B \subseteq T_2'$, with $|A| = |B| = \kappa'$ for some $\kappa' \leq \frac{\alpha \kappa}{\Delta}$, and a set $S$ of $\kappa' - 1$ vertices, separating $A$ from $B$ in $G$. From Corollary 2.1, there is a connected component $C$ of $G \setminus S$ containing at least $|T| - \frac{\kappa' \Delta}{\alpha} \geq 1.5\kappa$ terminals.

Therefore, $C$ contains at least $\kappa / 2$ terminals of $T_1$, and at least $\kappa / 2$ terminals of $T_2$. We claim that at least one terminal of $A$ must belong to $C$. Assume otherwise. Let $A'$ be any subset of $\kappa'$ terminals of $T_1 \cap C$. Since $T_1$ is node-well-linked in $G$, set $S$ cannot separate $A$ from $A'$, a contradiction. Therefore, at least one terminal $t \in A$ belongs to $C$. Similarly, at least one terminal $t' \in B$ belongs to $C$, contradicting the fact that $S$ separates $A$ from $B$. 

\[\Box\]
2.3 Boosting Well-Linkedness

Suppose we are given a graph $G$ and a set $T$ of vertices of $G$ called terminals, where $T$ 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 $\alpha$. We start with the following simple claim, that has been extensively used in past work to boost well-linkedness of terminals.

Claim 2.4 Suppose we are given a graph $G$ and a set $T$ of vertices of $G$, called terminals, such that $T$ is $\alpha$-well-linked. Assume further that we are given a collection $S$ of trees in $G$, and every tree $T \in S$ is associated with a subset $\Gamma_T \subseteq V(T) \cap T$ of $\lceil 1/\alpha \rceil$ terminals, such that, for every pair $T \neq T'$ of the trees, $\Gamma_T \cap \Gamma_{T'} = \emptyset$. Assume further that each edge of $G$ belongs to at most $c$ trees, and let $T' \subset T$ be any subset of terminals, containing exactly one terminal from each tree, we obtain a subset of $\Omega(\kappa)$ terminals.

Proof: Let $(A,B)$ be any partition of the vertices of $G$, and let $T_A = T' \cap A$, $T_B = T' \cap B$. Assume w.l.o.g. that $|T_A| \leq |T_B|$ and denote $|T_A| = \kappa$. Let $T'_B \subseteq T_B$ be any subset of $\kappa$ terminals in $T_B$. Our goal is to show that $|E(A,B)| \geq \kappa/(c+1)$. In order to do so, it is enough to show the existence of a flow $F$ of value $\kappa$ from $T_A$ to $T_B$ with edge-congestion at most $c + 1$. We construct the flow $F$ below.

For each terminal $t \in T_A$, consider the unique tree $T(t) \in S$, with $t \in V(T(t))$, and let $X = \bigcup_{t \in T_A} \Gamma_{T(t)}$. Define $Y$ similarly for $T_B$. Since the sets $\Gamma_{T}$ are pairwise disjoint, $|X| = |Y| = \kappa \cdot \lceil 1/\alpha \rceil$. Since the set $T$ of terminals is $\alpha$-well-linked, from the min-cut/max-flow theorem, there is a flow $F'$ in $G$ from $X$ to $Y$, where every terminal in $X$ sends one flow unit, every terminal in $Y$ receives one flow unit, and the total edge-congestion is at most $1/\alpha$. Scaling this flow down by factor $\lceil 1/\alpha \rceil$, we obtain a new flow $F''$ from $X$ to $Y$, where every terminal in $X$ sends $1/\lceil 1/\alpha \rceil$ flow units, every terminal in $Y$ receives $1/\lceil 1/\alpha \rceil$ flow units, and the edge-congestion due to $F''$ is at most $1$.

The final flow $F$ will consist of a concatenation of three flows: $F_1,F'',F_2$. Flow $F_1$ is defined as follows. Each terminal $t \in T_A$ sends one flow unit to the $\lceil 1/\alpha \rceil$ terminals of $\Gamma_t$, splitting the flow evenly among them, so every terminal in $\Gamma_t$ receives $1/\lceil 1/\alpha \rceil$ flow units. The flow is sent along the edges of the tree $T(t)$. The flow $F_1$ is the union of all such flows from all terminals $t \in T_A$. The flow $F_2$ is defined similarly with respect to $T_B$, except that we reverse the direction of the flow. The final flow is a concatenation of $F_1,F'',F_2$. It is easy to see that this is a valid flow from $T_A$ to $T_B$, where every terminal in $T_A$ sends one flow unit and every terminal in $T_B$ receives one flow unit, so the flow value is $\kappa$. Since each edge participates in at most $c$ trees in $S$, the congestion due to $F$ is bounded by $c + 1$.

The above claim gives a straightforward way to boost the well-linkedness of a given set $T$ of terminals, as follows. Let $G$ be a connected graph with maximum vertex degree $\Delta$, and $T \subseteq V(G)$ a set of $\kappa$ terminals, that is $\alpha$-well-linked in $G$. Then we can build a spanning tree $T$ of $G$, and partition it into disjoint sub-trees, each containing at least $\lceil 1/\alpha \rceil$ and at most $\Delta \cdot \lceil 1/\alpha \rceil$ terminals. Selecting one terminal from each resulting tree, we obtain a subset of $\Omega(\kappa \alpha / \Delta)$ terminals, that are $1/2$-well-linked. This type of argument has been used before extensively, usually under the name of a “grouping technique” [CKS13, CKS05, RZ10, And10, Chu12].

However, we need a stronger result: given a set $T$ of terminals, that are $\alpha$-well-linked in $G$, we would like to find a large subset $T' \subset T$, such that $T'$ is node-well-linked in $G$. The following theorem allows us to achieve this, generalizing a similar theorem for edge-disjoint routing in [CKS13]. The proof appears in the Appendix.

Theorem 2.2 Suppose we are given a connected graph $G = (V,E)$ with maximum vertex degree $\Delta$, and a subset $T$ of $\kappa$ vertices called terminals, such that $T$ is $\alpha$-well-linked in $G$, for some $\alpha < 1$, and $\kappa \geq 8\Delta / \alpha$. Then there is a subset $T' \subset T$ of $\Omega(\frac{\kappa}{\alpha^3})$ terminals, such that $T'$ is node-well-linked in $G$. 


Moreover, if $\kappa \geq 64\Delta^4\beta_{ARV}(\kappa)/\alpha$, then there is an efficient algorithm that computes a subset $T' \subset T$ of $\Omega\left(\frac{\alpha}{\Delta^4\beta_{ARV}(\kappa)} \cdot \kappa\right)$ terminals, such that $T'$ is node-well-linked in $G$. The algorithm also computes, for each terminal $t \in T'$, a tree $T_t \subseteq G$ containing at least $\lceil 1/\alpha \rceil$ terminals of $T$, with $t \in V(T_t)$, such that all trees $\{T_t\}_{t \in T'}$ 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 $T = \bigcup_j T_j$ 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 $T_j^* \subset T_j$ 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 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 $\Delta$, and let $T_1, \ldots, T_r$ be a collection of disjoint vertex subsets, called terminals, such that $T = \bigcup_j T_j$ is $\alpha$-well-linked in $G$ for some $\alpha < 1$, and $|T_j| \geq \kappa$ for all $j$, where $\kappa \geq 64\Delta^4\beta_{ARV}(\kappa)/\alpha$. 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 $T_j^* \subset T_j'$ be any subset of $\lceil |T_j|/12\Delta \rceil = \Omega\left(\frac{\alpha}{\Delta^4\beta_{ARV}(\kappa)} \cdot \kappa\right)$ 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 $G$.

**Proof:** It is immediate that for each $1 \leq j \leq r$, $T_j^*$ is node-well-linked in $G$, from Theorem 2.2. It now only remains to prove that for all $1 \leq i \neq j \leq r$, $T_i^*$ and $T_j^*$ are linked in $G$.

Fix some $1 \leq i \neq j \leq r$. From Theorem 2.1, in order to prove that $T_i^*$ and $T_j^*$ are linked in $G$, it is enough to show that $T_i^* \cup T_j^*$ is 1/3-well-linked in $G$.

Recall that when Theorem 2.2 was applied to set $T_j$, it returned a set $S = \{T_t\}_{t \in T_j'}$ of disjoint trees, where each tree $T_t$ contains $t$ and at least $\lceil 1/\alpha \rceil - 1$ additional terminals of $T_j$. We let $\Gamma_{T_t}$ be any subset of exactly $\lceil 1/\alpha \rceil$ terminals of $T_j \cap V(T_t)$, that includes the terminal $t$. Similarly, we are given a set $S' = \{T_t\}_{t \in T_j'}$ of disjoint trees for $T_j'$, and we define the sets $\Gamma_{T_t}$ of terminals for each tree $T_t \in S'$ similarly.

Since the original set $T$ of terminals was $\alpha$-well-linked, $T_i \cup T_j$ is also $\alpha$-well-linked. We can then apply Claim 2.4 to the set $T_i \cup T_j$, and the resulting set $T_i' \cup T_j'$ of terminals, with the collection $S \cup S'$ of trees. Since each edge belongs to at most two such trees, we conclude that $T_i' \cup T_j'$ is 1/3-well-linked. □

### 2.4 Treewidth and Well-Linkedness

The following lemma summarizes an important connection between the graph treewidth, and the size of the largest node-well-linked set of vertices.

**Lemma 2.1 (Reed [Ree97])** 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 $G$. Kreuter and Tazari [KT10] give a constructive version of this lemma, obtaining a set $X$ with slightly weaker properties. Lemma 2.2 below rephrases, in terms convenient to us, Lemma 3.7 in [KT10].

---

3Lemma 2.2 is slightly weaker than what was shown in [KT10]. We use it since it suffices for our purposes and avoids the introduction of additional notation.
Lemma 2.2 There is an efficient algorithm, that, given a graph \( G \) of treewidth \( k \), finds a set \( X \) of \( \Omega(k) \) vertices, such that \( X \) is \( \alpha^* = \Omega(1/\log k) \)-well-linked in \( G \). Moreover, for any partition \((X_1, X_2)\) of \( X \) into two equal-sized subsets, there is a collection \( \mathcal{P} \) of paths connecting every vertex of \( X_1 \) to a distinct vertex of \( X_2 \), such that every vertex of \( G \) participates in at most \( 1/\alpha^* \) paths in \( \mathcal{P} \).

2.5 A Tree with Many Leaves or a Long 2-Path

Suppose we are given any connected \( n \)-vertex graph \( Z \). A simple path \( P \) in \( Z \) is called a 2-path iff every vertex \( v \in P \) has degree 2 in \( Z \). In particular, \( P \) must be an induced path in \( Z \). The following theorem, due to Leaf and Seymour \[LS12\] states that we can find either a spanning tree with many leaves or a long 2-path in \( Z \). For completeness, the proof appears in the Appendix.

Theorem 2.3 Let \( Z \) be any connected \( n \)-vertex graph, and \( \ell \geq 1, p \geq 1 \) any integers with \( n^2 \ell \geq p + 5 \). Then there is an efficient algorithm that either finds a spanning tree \( T \) with at least \( \ell \) leaves in \( Z \), or a 2-path of length at least \( p \) in \( Z \).

2.6 Re-Routing Two Sets of Disjoint Paths

We need the following lemma, whose proof closely follows arguments of Conforti, Hassin and Ravi \[CHR03\] and appears in the Appendix.

Lemma 2.3 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 \( Y' = \{ \hat{Q} \mid Q \in Y \} \), then all paths in \( X' \cup Y' \) are pairwise disjoint (except for sharing the last vertex \( s \)).

2.7 Cut-Matching Game and Degree Reduction

We say that a (multi)-graph \( G = (V, E) \) is an \( \alpha \)-expander, iff \( \min_{S \subseteq V, |S| \leq |V|/2} \left\{ \frac{|E(S, S)|}{|S|} \right\} \geq \alpha \).

We use the cut-matching game of Khandekar, Rao and Vazirani \[KRV09\]. In this game, we are given a set \( V \) of \( N \) vertices, where \( N \) is even, and two players: a cut player, whose goal is to construct an expander \( X \) on the set \( V \) of vertices, and a matching player, whose goal is to delay its construction. The game is played in iterations. We start with the graph \( X \) containing the set \( V \) of vertices, and no edges. In each iteration \( j \), the cut player computes a bi-partition \((A_j, B_j)\) of \( V \) into two equal-sized sets, and the matching player returns some perfect matching \( M_j \) between the two sets. The edges of \( M_j \) are then added to \( X \). Khandekar, Rao and Vazirani have shown that there is a strategy for the cut player, guaranteeing that after \( O(\log^2 N) \) iterations we obtain a \( 1/2 \)-expander w.h.p. Subsequently, Orecchia et al. \[OSV08\] have shown the following improved bound:

Theorem 2.4 (\[OSV08\]) There is a probabilistic algorithm for the cut player, such that, no matter how the matching player plays, after \( \gamma_{CMG}(N) = O(\log^2 N) \) iterations, graph \( X \) is an \( \alpha_{CMG}(N) = \Omega(\log N) \)-expander, with constant probability.
Let $G$ be any graph with $\text{tw}(G) = k$. The proof of Theorem 1.4 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 and the treewidth, we need to reduce the maximum vertex degree of the input graph $G$. Using the cut-matching game, one can reduce the maximum vertex degree to $O(\log^3 k)$, while only losing a polylog $k$ factor in the treewidth, as was noted in [CE13] (see Remark 2.2). The following theorem, whose proof appears in the Appendix, provides the starting point for our algorithm.

**Theorem 2.5** 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 $\Omega(k/\text{polylog } k)$ 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 [KT10]; the constant can be made 4 with a polynomial factor loss in treewidth [KT10].

### 3 A Path-of-Sets System

In this section we define our main combinatorial object, called a 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 $S_j$ 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 $\bigcup_{S \in S} S$ 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 2).

We say that it is a strong path-of-sets system, if additionally for each $1 < i \leq r$, $A_i$ is node-well-linked in $G[S_i]$, and for each $1 \leq i < r$, $B_i$ is node-well-linked in $G[S_i]$. We note that Leaf and Seymour [LS12] have defined a very similar object, called an $(h, r)$-grill, and they 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 $h = \Omega(g^2)$ and $r = \Omega(g^3)$, finds a $(g \times g)$-grid minor in the grill. The following theorem slightly strengthens their result, when the starting point is a path-of-sets system. Its proof appears in the Appendix.

---

4In fact [LS12] shows a slightly stronger result that a $(h, r)$-grill with $h \geq (2g + 1)(2\ell - 5) + 2$ and $r \geq \ell(2g + \ell - 2)$ 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 $\ell$. 

9
Theorem 3.1 Given a graph $G$ and a path-of-sets system of width $h$ and height $h$ in $G$, there is an efficient algorithm that finds a model of a grid minor of size $\Omega(\sqrt{h}) \times \Omega(\sqrt{h})$ in $G$.

The main technical contribution of our paper is summarized in the following theorem.

Theorem 3.2 Let $G$ be any graph of treewidth $k$, and let $h^*, r^* > 2$ be integral parameters, such that for some large enough constants $c'$ and $c''$, $k/\log^c k > c''h^*(r^*)^{48}$. Then there is an efficient randomized algorithm, that, given $G, h^*$ and $r^*$, w.h.p. computes a strong path-of-sets system of height $h^*$ and width $r^*$ in $G$.

Choosing $h^*, r^* = \Omega(k^{1/49}/\polylog k)$, from Theorem 3.2, we can w.h.p. construct a path-or-sets system of height $h^*$ and width $r^*$ in $G$. From Theorem 3.1 we can construct a grid minor of size $\Omega((k^{1/98}/\polylog k) \times (k^{1/98}/\polylog k))$. The rest of this paper is mostly dedicated to proving Theorem 3.2. In Section 6, we provide some extensions to this theorem, that we believe may be useful in various applications, such as, for example, algorithms for routing problems.

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 $P_e$ 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 [CL12], combined with some ideas from [CE13] 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 tree-of-sets system. We start with the following definitions.

Definition 4.1 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 4.2 A tree-of-sets system with parameters $r, h, \alpha_{\text{bw}}$ 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 $S_i$ 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 $P = \bigcup_{e \in E(T)} P_e$ are pairwise disjoint,

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

The following theorem strengthens the results of [CL12], and its proof appears in Section 5.

Theorem 4.1 Suppose we are given a graph $G$ of maximum vertex degree $\Delta$, 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 any parameters $r > 1, h > 4\log k$, such that $k/\log^4 k > c h^{19} \Delta^8$, where $c$ is a large enough constant. Then there is an efficient randomized algorithm that
with high probability computes a tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P_e)\) in \(G\), with parameters \([h]_r\) and \(\alpha_{BW} = \Omega \left( \frac{1}{r^2 \log^{12\Delta} r} \right)\). Moreover, for all \(S_i \in S\), \(S_i \cap T = \emptyset\).

We prove Theorem \ref{thm:arv} in the following section, and show how to construct a path-of-sets system using this theorem here.

Suppose we are given a tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P_e)\), and an edge \(e \in E(T)\), incident on a vertex \(v_i \in V(T)\). We denote by \(\Gamma_{S_i}(e)\) the set of all endpoints of the paths in \(P(e)\) that belong to \(S_i\).

**Definition 4.3** A tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P(e))\) with parameters \(h, r, \alpha_{BW}\) is a strong tree-of-sets system, iff for each \(S_i \in S\):

- for each edge \(e \in E(T)\) incident on \(v_i\), the set \(\Gamma_{S_i}(e)\) is node well-linked in \(G[S_i]\); and
- for every pair \(e, e' \in E(T)\) of edges incident on \(v_i\), the sets \(\Gamma_{S_i}(e)\) and \(\Gamma_{S_i}(e')\) are linked in \(G[S_i]\).

The following lemma allows us to transform an arbitrary tree-of-sets system into a strong one.

**Lemma 4.1** Let \(G\) be a graph with maximum degree \(\Delta\) and suppose \(G\) has a tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P_e)\) with parameters \(r, h, \alpha_{BW}\). There is an efficient randomized algorithm that given \(G\) and \((S, T, \bigcup_{e \in E(T)} P_e)\) outputs a strong tree-of-set system \((S, T, \bigcup_{e \in E(T)} P_e^*)\) with parameters \(r, \tilde{h}, \alpha_{BW}\) such that \(P_e^* \subset P_e\) for each \(e \in E(T)\), and \(\tilde{h} = \Omega \left( \frac{\alpha_{BW}}{\Delta \log(\beta_{ARV}(h))} \cdot h \right)\).

**Proof:** We prove the lemma by boosting well-linkedness using the claims in Section 2.3. Consider the given tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P_e)\) in \(G\). Let \(S_i \in S\). 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 \(S_i\) and \(\Gamma_1\), to obtain a subset \(\Gamma_1' \subseteq \Gamma_1\) of \(\Theta \left( \frac{\alpha_{BW}}{\Delta \log(\beta_{ARV}(h))} \cdot h \right)\) 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 \(|P_e'| = \Theta \left( \frac{\alpha_{BW}}{\Delta \log(\beta_{ARV}(h))} \cdot h \right)\). 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 \(\Gamma_2' \subset \Gamma_2\) of \(\Theta \left( \frac{\alpha_{BW}}{\Delta \log(\beta_{ARV}(h))} \cdot h \right)\) 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 \(\tilde{h} = \left\lfloor \frac{|P_e'|}{12\Delta} \right\rfloor = \Theta \left( \frac{\alpha_{BW}}{\Delta \log(\beta_{ARV}(h))} \cdot h \right)\) 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 \(e \in E(T)\) in this manner. Consider now any non-leaf vertex \(v_i \in T\) and its corresponding set \(S_i \in S\). 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 \(\Gamma_{S_j}(e')\) of vertices are linked in \(G[S_i]\). It is immediate to see that for each edge \(e \in E(T)\) and each endpoint \(v_i\) of \(e\), \(\Gamma_{S_i}(e)\) is node well-linked in \(G[S_i]\). \(\square\)

The following theorem allows us to obtain a strong path-of-sets system from a strong tree-of-sets system.

**Theorem 4.2** Let \((S, T, \bigcup_{e \in E(T)} P_e^*)\) be a strong tree-of-set system in \(G\) with parameters \(r, \tilde{h}\). There is a polynomial-time algorithm that, given \(G\), \((S, T, \bigcup_{e \in E(T)} P_e^*)\) and integer parameters \(h^*, r^*\) outputs a strong path-of-set system of width \(r^*\) and height \(h^*\) if \((r^*)^2 \leq r\) and \(\tilde{h} > 16h^*(r^*)^2 + 1\).

Before we prove the preceding theorem we use the results stated so far to complete the proof of Theorem 3.2.
Proof of Theorem 3.2. 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 2.5 to compute a subgraph \( G' \) of \( G \) with maximum vertex degree \( \Delta = O(\log^3 k) \), and a set \( X \) of \( \Omega(k/\log^3 k) \) 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 can now assume that \( \frac{k}{\Delta \log^3 k} > \hat{c}h^*(r^*)^{48} \) for some large enough constant \( \hat{c} \).

We set \( r = (r^*)^2 \) and \( h = \frac{c}{\hat{c}} \cdot h^*(r^*)^{10} \Delta^{11} \log^4 k \), so \( h > 4 \log k \) holds. Clearly:

\[
chr^{19} \Delta^8 = (\hat{c}h^*(r^*)^{10} \Delta^{11} \log^4 k) \cdot (r^*)^{38} \Delta^8 = \hat{c}h^*(r^*)^{48} \Delta^{19} \log^4 k.
\]

Therefore, \( \frac{k}{\log^3 k} > chr^{19} \Delta^8 \). We then apply Theorem 4.1 to \( G \) and \( T \) to obtain a tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P_e)\), with parameters \( r \), \( h \) and \( \alpha_{\text{bw}} = \Omega(\frac{1}{r^2 \log^3 k}) \).

We use Lemma 4.1 to convert \((S, T, \bigcup_{e \in E(T)} P_e)\) into a strong tree-of-sets system \((S, T, \bigcup_{e \in E(T)} P^*_e)\) with parameters \( r \) and \( \hat{h} = \Omega(\frac{\alpha_{\text{bw}}^2}{\Delta^{43} \log^8 (\hat{h}^*)}) \cdot \hat{h} \). If \( \hat{c} \) is chosen to be large enough, \( \hat{h} > 16 h^*(r^*)^2 + 1 \) must hold. We then apply Theorem 4.2 to obtain a path-of-set system with height \( h^* \) and width \( r^* \).

\(\square\)

We now prove Theorem 4.2.

Proof of Theorem 4.2. Let \((S, T, \bigcup_{e \in E(T)} P^*_e)\) be a strong tree-of-set system with parameters \( r \) and \( \hat{h} \). Let \( h^*, r^* \) be integers such that \( (r^*)^2 \leq r \) and \( \hat{h} > 16 h^*(r^*)^2 + 1 \).

For convenience, for each set \( S \in \mathcal{S} \), we denote the corresponding vertex of \( T \) by \( v_S \). If \( T \) 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 \( \hat{h} \geq h^* \) and width \( r^* \). The path-of-sets system is strong, since for every edge \( e = (v_i, v_j) \in E(T) \), \( \Gamma_{S_i}(e) \) is node-well-linked in \( G[S_i] \). (We note that since for each \( S \in \mathcal{S} \), \( T \cap S = \emptyset \), and since the degree of each terminal is 1, the terminals do not participate in the clusters \( S_i \) and in the paths \( P_i \) of the path-of-sets system, and so the path-of-sets system is contained in the original graph \( G \).)

Otherwise, since \(|V(T)| = r \geq (r^*)^2 \), \( T \) must contain at least \( r^* \) leaves (see Claim 2.2). Let \( L \) be any subset of \( r^* \) leaves of \( T \) (if there are more leaves in \( T \), we only choose \( r^* \) of them). Let \( \mathcal{L} \subseteq \mathcal{S} \) be the collection of \( r^* \) clusters, whose corresponding vertices belong to \( L \), \( \mathcal{L} = \{ S \in \mathcal{S} \mid v_S \in L \} \). 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 our 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 \( S \) in our final path-of-sets system, 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 \( S \in \mathcal{S} \). 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 (see Figure 2). 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.
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-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 $S \in S$, 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 $\left\lfloor \frac{\tilde{h}}{r^*} \right\rfloor$ 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 $S'$ is a contiguous segment of $Q$. For all other clusters $S'$, $Q \cap S' = \emptyset$.

- 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 $T$ 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 $P^*_e$. 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 $n(S') \cdot \left\lfloor \frac{\tilde{h}}{r^*} \right\rfloor$ 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 $S''$ and $Q$ is a contiguous segment of $Q$, and for any other cluster $S''$, $Q \cap S'' = \emptyset$. 

Figure 2: Routing paths of the DFS tour inside $S$
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 \( n(S) \cdot \alpha \) paths of \( P^e_\alpha \), and we set \( Q = Q_S \). (Notice that \( |L| \cdot \alpha \leq \tilde{h} = |P^e_\alpha| \), 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 \( Q_S \subset Q \) 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 \( n(S') \cdot \alpha \) vertices and a subset \( \Gamma_2 \subset \Gamma_S(e_2) \) of \( n(S'') \cdot \alpha \) vertices, together with a set \( R \) of \( |\Gamma'| \) disjoint paths contained inside \( G'\big|S \), 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 \( n(S') \cdot \alpha \) and connect it with a directed edge to every vertex of \( \Gamma_S(e_1) \). Similarly, add a new vertex \( s_2 \) of capacity \( n(S'') \cdot \alpha \) 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 \( |\Gamma'| = n(S) \cdot \alpha = (n(S') + n(S'')) \cdot \alpha \) in this flow network. Since \( \Gamma' \) and \( \Gamma_S(e_1) \) are linked, there is a set \( \mathcal{P}_1 \) 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 \( \mathcal{P}_2 \) 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. \( \square \)

Let \( \mathcal{P}_1 \subseteq \mathcal{P}(e_1) \) be a subset of paths whose endpoints belong to \( \Gamma_1 \), and define \( \mathcal{P}_2 \subseteq \mathcal{P}(e_2) \) similarly for \( \Gamma_2 \). Concatenating the paths in \( Q_S, R \), and \( \mathcal{P}_1 \cup \mathcal{P}_2 \), we obtain two collections of paths: set \( Q_{S'} \) of \( n(S') \cdot \alpha \) paths, connecting \( S^* \) to \( S' \), and set \( Q_{S''} \) of \( n(S'') \cdot \alpha \) 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. \( \square \)

### 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 \( \{Q_S \mid S \in L \} \) 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 \( P^e_\alpha \) 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 \( L_S = \{S' \mid v_{S'} \in L_S \} \) 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 \( \ell_S \) 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 \( (L_S, P^S) \) of height \( h^* \) and width \( |L_S| \), that is completely contained in \( G_S \). (That is, the path-of-sets system is defined over the collection \( L_S \) of vertex subsets - all subsets \( S' \in L \) 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 $Q'_A \subseteq Q_A, Q'_B \subseteq Q_B$ of paths of size at least $\frac{h}{2r_{\mathcal{L}}} - 8\ell_S \cdot h^*$, such that the paths in $Q'_A \cup Q'_B$ are completely disjoint from the paths in $\mathcal{P}^S$ (see Figure 3). Note that $Q_A, Q_B$ are the sets of paths computed in Step 1, so the paths in $Q_A \cup Q_B$ 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 $r^*$, $\frac{h}{2r_{\mathcal{L}}} - 8\ell_S \cdot h^* \geq \frac{h}{2r_{\mathcal{L}}} - 8r^* \cdot h^* > 0$ where the latter inequality is based on the assumption that $h > 16h^*(r^*)^2 + 1$.

![Figure 3: Invariant for Step 2.](image)

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 $Q'_A, Q'_B$ be any pair of disjoint subsets of $Q_S$ containing $\frac{h}{2r_{\mathcal{L}}}$ 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 $Q'_A, Q'_B$ 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 4[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 $Q'_{A_1} \cup Q'_{B_1} \cup Q'_{A_2} \cup Q'_{B_2}$, and additional paths contained inside $S$.

Consider the paths in $Q'_{A_1}$ 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 $\Gamma_1 = \{v_Q \mid q \in Q'_{A_1}\}$. We similarly define $\Gamma_2, \Gamma'_1, \Gamma'_2$ for $Q'_{B_1}, Q'_{A_2}$ and $Q'_{B_2}$, 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 $Q'_{A_1} \cup Q'_{B_1}$ that are contained in $S$ by $\mathcal{P}$, and the portions of paths in $Q'_{A_2} \cup Q'_{B_2}$ that are contained in $S$ by $\mathcal{P}'$ (see Figure 4(b)). That is,

$$\mathcal{P} = \{P \cap G[S] \mid P \in Q'_{A_1} \cup Q'_{B_1}\}; \quad \mathcal{P}' = \{P \cap G[S] \mid P \in Q'_{A_2} \cup Q'_{B_2}\}$$

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 $\mathcal{P}$, and at most $8h^*$ paths in $\mathcal{P}'$. Notice that in general, since sets $\Gamma, \Gamma'$
are linked in \( G'[S] \), we can find a set \( R \) of \( 4h^* \) disjoint paths inside \( S \) connecting \( \Gamma \) to \( \Gamma' \), but 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 2.3.

![Figure 4: Processing a degree-3 vertex \( v_S \).](image)

We apply Lemma 2.3 twice. First, we unify all vertices of \( \Gamma \) into a single vertex \( s \), and direct the paths in \( P \) and the paths in \( R \) towards it. We then apply Lemma 2.3 to the two sets of paths, with \( P \) as \( X \) and \( R \) as \( Y \). Let \( \tilde{P} \subset P \), \( R' \) be the two resulting sets of paths. We discard from \( \tilde{P} \) paths that share endpoints with paths in \( R' \) (at most \( |R'| \) paths). Then \( |\tilde{P}| \geq |P| - 2|R'| = |P| - 8h^* \), and \( R' \) contains \( 4h^* \) disjoint paths connecting vertices in \( \Gamma \) to vertices in \( \Gamma' \). Moreover, the paths in \( \tilde{P} \cup R' \) 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 2.3 to the two resulting sets of paths, with \( P' \) serving as \( X \) and \( R' \) serving as \( Y \). Let \( \tilde{P}' \subset P' \) and \( R'' \) be the two resulting sets of paths. We again discard from \( \tilde{P}' \) all paths that share an endpoint with a path in \( R'' \) – at most \( |R''| \) paths. Then \( |\tilde{P}'| \geq |P'| - 2|R''| \geq |P'| - 8h^* \), and the paths in \( \tilde{P}' \cup R'' \) are completely disjoint from each other. Notice also that the paths in \( R'' \) remain disjoint from the paths in \( \tilde{P} \), since the paths in \( R'' \) 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 $\Gamma_i$ to the vertices of $\Gamma'_j$. 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 $\Gamma_2$ to the vertices of $\Gamma'_1$. 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 4(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, $|\mathcal{Q}'_{A_1}| \geq \left\lceil \frac{h}{2r^2} \right\rceil - 8h^* \ell_S \geq \left\lceil \frac{h}{2r^2} \right\rceil - 8h^*(\ell_S - 1)$, at the end of the current iteration, $|\mathcal{Q}'_{A_1}| \geq \left\lceil \frac{h}{2r^2} \right\rceil - 8h^* \ell_S$ 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 from all paths participating in the new path-of-sets system.

Notice that each vertex $v_i \in L$ is only incident on one edge $e \in E(T)$, and from the definition of strong tree-of-sets system, $\Gamma_{S_i}(e)$ is node-well-linked in $G[S_i]$. These are the only vertices of $S_i$ that may participate in the paths $\mathcal{P}_j$ of the path-of-sets system, so we obtain a strong path-of-sets system. We note that since for each $S \in \mathcal{S}$, $\mathcal{T} \cap S = \emptyset$, and since the degree of each terminal is 1, the terminals do not participate in the clusters $S_i$ and in the paths $\mathcal{P}_i$ of the path-of-sets system, and so the path-of-sets system is contained in the original graph $G$. □

In order to complete the proof of Theorem 1.1, it is now enough to prove Theorem 4.1.

5 Proof of Theorem 4.1

This part mainly follows the algorithm of CL12. 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 CL12 to handle vertex connectivity instead of edge connectivity. We also improve and simplify some of the arguments of CL12. Some of the proofs and definitions are identical or closely follow those in CL12 and are provided here for the sake of completeness. For simplicity, if $(\mathcal{S}, T, \bigcup_{e \in E(T)} \mathcal{P}_e)$ is a tree-of-sets system in $G$, with parameters $h, r, \alpha_{bw}$ as in the theorem statement, and for each $S_i \in \mathcal{S}$, $S_i \cap T = \emptyset$, then we say that it is a good tree-of-sets system.

5.1 High-Level Overview

In this subsection we provide a high-level overview and intuition for the proof of Theorem 4.1. We also describe a non-constructive proof of the theorem, which is somewhat simpler than the constructive proof that appears below. This high-level description oversimplifies some parts of the algorithm for the sake of clarity. This subsection is not necessary for understanding the algorithm and is only provided for the sake of intuition. A formal self-contained proof appears in the following subsections.

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) $S \cap T = \emptyset$; (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
that allows us to build the tree-of-sets system in this phase. As before, we perform standard splitting

From a Good Family of Routers to a Good Tree-of-Sets System

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 \( \mathcal{R} \) (a formal proof appears in Section 5.4). 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 \( h_0/6 \delta_3 \) node-disjoint paths connecting \( S_i \) to \( S_j \). We build an auxiliary graph \( H \) from \( G \), 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 \( \mathcal{T} = \{ v_1, \ldots, v_{r_0} \} \). We then use standard splitting procedures in graph \( H \) repeatedly, to obtain a new graph \( H' \), whose vertex set is \( \mathcal{T} \), every pair of vertices remains \( h_0 \poly(\Delta) \)-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, and then performing standard edge-splitting steps that preserves edge-connectivity. Let \( Z \) be a graph whose vertex set is \( \mathcal{T} \), and there is an edge \((v_i, v_j)\) in \( Z \) iff there are many (say \( h_0 \poly(\Delta) \)) 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 Theorem 2.1 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 \( h_3 = h \poly(r \cdot \Delta) \) edges, 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 \( \bigcup \mathcal{P}_i \) 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 Theorem 2.1 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 \( \mathcal{T} = \{ v_1, \ldots, v_r \} \) 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 \(\tilde{T}\). 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 \(\{P_e \mid e \in E(\tilde{H}')\}\) 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 \(\tilde{T}\) 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. The construction of the spanning tree is performed using a result of Singh and Lau \[SL07\], who showed an approximation algorithm for constructing a minimum-degree spanning tree of a graph. Their algorithm is based on an LP-relaxation of the problem. They show that, given a feasible solution to the LP-relaxation, one can construct a spanning tree with maximum degree bounded by the maximum fractional degree plus 1. Therefore, it is enough to show that there is a solution to the LP-relaxation on graph \(\tilde{Z}\), where the fractional degree of every vertex is bounded by 2. The fact that the degree of every vertex, and the edge-connectivity of every pair of vertices are very close to the same value allows us to construct such a solution.

An alternative 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 \[Win80\].

Finding a Good Family of Routers

One of the main tools that we use in this part is a good clustering of the graph \(G\) and a legal contracted graph associated with it. We say that a subset \(C \subseteq V(G)\) of vertices is a small cluster iff \(|\text{out}(C)| \leq h_0\), and we say that it is a large cluster otherwise. A partition \(C\) of \(V(G)\) is called a good clustering iff each terminal \(t \in \mathcal{T}\) belongs to a separate cluster \(C_t \in C\), where \(C_t = \{t\}\), all clusters in \(C\) are small, and each cluster has the \(\alpha_{nw}\)-bandwidth property. Given a good clustering \(C\), the corresponding legal contracted graph is obtained from \(G\) by contracting every cluster \(C \in \mathcal{C}\) into a super-node \(v_C\) (notice that terminals are not contracted, since each terminal is in a separate cluster). The legal contracted graph can be seen as a model of \(G\), where we “hide” some irrelevant parts of the graph inside the contracted clusters. The main idea of the algorithm is to exploit the legal contracted graph in order to find a good family of routers, and, if we fail to do so, to construct a smaller legal contracted graph. We start with a non-constructive proof of the existence of a good family of routers in \(G\).

Non-Constructive Proof We assume that \(G\) is minimal inclusion-wise, for which the set \(\mathcal{T}\) of terminals is 1-well-linked. That is, for any edge \(e \in E(G)\), if we delete \(e\) from \(G\), then \(\mathcal{T}\) is not 1-well-linked in the resulting graph. Let \(C^*\) be a good clustering of \(V(G)\) minimizing the total number of edges in the corresponding legal contracted graph (notice that a partition where every vertex belongs to a separate cluster is a good clustering, so such a clustering exists). Consider the resulting legal contracted graph \(G'\). The degree of every vertex in \(G'\) is at most \(h_0\) and, from the well-linkedness of the terminals in \(G\), it is not hard to show that \(G' \setminus \mathcal{T}\) must contain at least \(\Omega(k)\) edges. Then there is a partition \(\{X_1, \ldots, X_{r_0}\}\) of \(V(G') \setminus \mathcal{T}\), where for each \(1 \leq i \leq r_0\), \(|\text{out}_{G'}(X_i)| < O(r_0|E_{G'}(X_i)|)\)
(a random partition of $V(G') \setminus T$ into $r_0$ subsets will have this property with constant probability. This is since, if we denote $m = |E(G') \setminus T|$, then we expect roughly $\frac{m+k}{r_0}$ edges in each out-$G'(X_i)$, and roughly $m/r_0^2$ edges with both endpoints inside $X_i$.)

For each set $X_i$, let $X'_i \subseteq V(G) \setminus T$ be the corresponding subset of vertices of $G$, obtained by uncontracting each supernode $v_C$ (that is, $X'_i = \bigcup_{v_C \subseteq X_i} C$). If $\Gamma_i$ is the interface of $X'_i$ in $G$, then we still have that $|\Gamma_i| < O(r_0 |E_G(X_i)|)$.

As our next step, we would like to find a partition $\mathcal{W}_i$ of the vertices of $X'_i$ into clusters, such that each cluster $W \in \mathcal{W}_i$ has the $\alpha_{bw}$-bandwidth property, and the total number of edges connecting different clusters is at most $O(|\Gamma_i|/r_0) < |E_G(X'_i)|$. We call this procedure bandwidth-decomposition. Assume first that we are able to find such a decomposition. We claim that $\mathcal{W}_i$ must contain at least one good router $S_i$. If this is the case, then we have found the desired family $\{S_1, \ldots, S_{r_0}\}$ of good routers. In order to show that $\mathcal{W}_i$ contains a good router, assume first that at least one cluster $S_i \in \mathcal{W}_i$ is large. The decomposition $\mathcal{W}_i$ already guarantees that $S_i$ has the $\alpha_{bw}$-bandwidth property. If $S_i$ is not a good router, then it must be impossible to send large amounts of flow from $S_i$ to $T$ in $G$. In this case, using known techniques (see appendix of [CNS13]), we can show that we can delete an edge from $G[S_i]$, while preserving the 1-well-linkedness of the terminals, contradicting the minimality of $G$. Therefore, if $\mathcal{W}_i$ contains at least one large cluster, then it contains a good router. Assume now that all clusters in $\mathcal{W}_i$ are small. Then we show a new good clustering $\mathcal{C}'$ of $V(G)$, whose corresponding contracted graph contains fewer edges than $G'$, leading to a contradiction. The new clustering contains all clusters $C \in \mathcal{C}'$ with $C \cap X'_i = \emptyset$, and all clusters in $\mathcal{W}_i$. In other words, we replace the clusters contained in $X'_i$ with the clusters of $\mathcal{W}_i$. The reason the number of edges goes down in the legal contracted graph is that the total number of edges connecting different clusters of $\mathcal{W}_i$ is less than $|E_G(X'_i)|$.

The final part of the proof that we need to describe is the bandwidth-decomposition procedure. Given a cluster $X'_i$, we would like to find a partition $\mathcal{W}_i$ of $X'_i$ into clusters that have the $\alpha_{bw}$-bandwidth property, such that the number of edges connecting different clusters is bounded by $O(|\Gamma_i|/r_0)$. There are by now standard algorithms for finding such a decomposition, where we repeatedly select a cluster in $\mathcal{W}_i$ that does not have the desired bandwidth property, and partition it along a sparse cut [Rac02, CKS05]. Unfortunately, since our bandwidth parameter $\alpha_{bw}$ is independent of $n$, such an approach can only work when $|\Gamma_i|$ is bounded by $\text{poly}(k)$, which is not necessarily true in our case. In order to overcome this difficulty, as was done in [Chu12], we slightly weaken the bandwidth condition, and define a $(k, \alpha_{bw})$-bandwidth property as follows: We say that cluster $C$ with interface $\Gamma$ has the $(k, \alpha_{bw})$-bandwidth property, iff for any pair $A, B \subseteq \Gamma$ of equal-sized disjoint subsets, with $|A|, |B| \leq k$, the minimum edge-cut separating $A$ from $B$ in $G[C]$ has at least $\alpha_{bw} \cdot |A|$ edges. Alternatively, we can send $|A|$ flow units from $A$ to $B$ inside $G[C]$ with edge-congestion at most $1/\alpha_{bw}$. Notice that if $C$ does not have the $(k, \alpha_{bw})$-bandwidth property, then there is a partition $(C_1, C_2)$ of $C$, and two disjoint equal-sized subsets $A \subseteq \Gamma \cap C_1, B \subseteq \Gamma \cap C_2$, with $|A|, |B| \leq k$, such that $|E_G(C_1, C_2)| < \alpha_{bw} \cdot |A|$. We call such a partition a $(k, \alpha_{bw})$-violating cut of $C$. Even if we weaken the definition of the good routers, and replace the $\alpha_{bw}$-bandwidth property with the weaker $(k, \alpha_{bw})$-bandwidth property, we can still construct a good tree-of-systems from a family of good routers. This is since the construction algorithm only uses the $\alpha_{bw}$-bandwidth property of the routers in the weak sense, by sending small amounts of flow (up to $k$ units) across the routers. Given the set $X'_i$, we can now show that there is a partition $\mathcal{W}_i$ of $X'_i$ into clusters that have the $(k, \alpha_{bw})$-bandwidth property, such that the number of edges connecting different clusters is bounded by $O(|\Gamma_i|/r_0)$.

**Constructive Proof** A constructive proof is more difficult, for the following two reasons. First, given a large cluster $S_i$, that has the $\alpha_{bw}$-bandwidth property, but cannot send large amounts of flow to the terminals in $G$, we need an efficient algorithm for finding an edge that can be removed from $G[S_i]$ without violating the 1-well-linkedness of the terminals. While we know that such an edge must
exist, we do not have a constructive proof that allows us to find it. The second problem is related to the bandwidth-decomposition procedure. While we know that, given \( X_i \), there is a desired partition \( W_i \) of \( X_i \) into clusters that have the \((k, \alpha_{BW})\)-bandwidth property, we do not have an algorithmic version of this result. In particular, we need an efficient algorithm that finds a \((k, \alpha_{BW})\)-violating cut in a cluster that does not have the \((k, \alpha_{BW})\)-bandwidth property. (An efficient algorithm that gives a poly\((\log k)\) approximation, by returning an \((\Omega(k), \alpha_{BW} \cdot \text{poly} \log k)\)-violating cut would be sufficient, but as of now we do not have such an algorithm).

In addition to a good clustering defined above, our algorithm uses a notion of acceptable clustering. An acceptable clustering is defined exactly like a good clustering, except that large clusters are now allowed. Each small cluster in an acceptable clustering must have the \(\alpha_{BW}\)-bandwidth property, and each large cluster must induce a connected graph in \(G\).

In order to overcome the difficulties described above, we define a potential function \(\varphi\) over partitions \(\mathcal{C}\) of \(V(G)\). Given such a partition \(\mathcal{C}\), \(\varphi(\mathcal{C})\) is designed to be a good approximation of the number of edges connecting different clusters of \(\mathcal{C}\). Additionally, \(\varphi\) has the following two useful properties. If we are given any acceptable clustering \(\mathcal{C}\), a large cluster \(C \in \mathcal{C}\), and a \((k, \alpha_{BW})\)-violating cut \((C_1, C_2)\) of \(C\), then we can efficiently find a new acceptable clustering \(\mathcal{C}'\) with \(\varphi(\mathcal{C}') < \varphi(\mathcal{C}) - 1/n\). Similarly, if we are given an acceptable clustering \(\mathcal{C}\), and a large cluster \(C \in \mathcal{C}\), such that \(C\) cannot send \(h_0/2\) flow units to the terminals, then we can efficiently find a new acceptable clustering \(\mathcal{C}'\) with \(\varphi(\mathcal{C}') < \varphi(\mathcal{C}) - 1/n\).

The algorithm consists of a number of phases. In every phase, we start with some good clustering \(\mathcal{C}\), where in the first phase, \(\mathcal{C} = \{\{v\} : v \in V(G)\}\). In each phase, we either find a good tree-of-sets system, or find a new good clustering \(\mathcal{C}'\), with \(\varphi(\mathcal{C}') \leq \varphi(\mathcal{C}) - 1\). Therefore, after \(O(|E(G)|)\) phases, we are guaranteed to find a good tree-of-sets system.

We now describe an execution of each phase. Let \(\mathcal{C}\) be the current good clustering, and let \(G'\) be the corresponding legal contracted graph. As before, we find a partition \(\{X_1, \ldots, X_{r_0}\}\) of \(V(G')\) \setminus \(T\), where for each \(1 \leq i \leq r_0\), \(|\text{out}_{G'}(X_i)| < O(r_0|E_{G'}(X_i)|)\), using a simple randomized algorithm. For each set \(X_i\), let \(X'_i \subseteq V(G) \setminus T\) be the corresponding set of vertices in \(G\), obtained by un-contracting each supernode \(v_C \in X_i\). For each \(1 \leq i \leq r_0\), we also construct an acceptable clustering \(C_i\), containing all clusters \(C \in \mathcal{C}\) with \(C \cap X'_i = \emptyset\), and all connected components of \(G[X_i]\) (if any such connected component is a small cluster, we further partition it into clusters with \(\alpha_{BW}\)-bandwidth property). We show that \(\varphi(C_i) \leq \varphi(C) - 1\) for each \(i\). We then perform a number of iterations.

In each iteration, we are given as input, for each \(1 \leq i \leq r_0\), an acceptable clustering \(C_i\), with \(\varphi(C_i) \leq \varphi(C) - 1\), where each large cluster of \(C_i\) is contained in \(X'_i\). An iteration is executed as follows. If, for some \(1 \leq i \leq r_0\), the clustering \(C_i\) contains no large clusters, then \(C_i\) is a good clustering, with \(\varphi(C_i) \leq \varphi(C) - 1\). We then finish the current phase and return the good clustering \(C_i\). Otherwise, for each \(1 \leq i \leq r_0\), there is at least one large cluster \(S_i \in C_i\). We treat the clusters \(\{S_1, \ldots, S_{r_0}\}\) as a potential good family of routers, and try to construct a tree-of-sets system using them. If we succeed in building a good tree-of-sets system, then we are done, and we terminate the algorithm. Otherwise, we will obtain a certificate that one of the clusters \(S_i\) is not a good router. The certificate is either a \((k, \alpha_{BW})\)-violating partition of \(S_i\), or a small cut (containing fewer than \(h_0/2\) edges), separating \(S_i\) from the terminals. In either case, using the properties of the potential function, we can obtain a new acceptable clustering \(C'_i\) with \(\varphi(C'_i) \leq \varphi(C_i) - 1/n\), to replace the current acceptable clustering \(C_i\). We then continue to the next iteration.

Overall, as long as we do not find a good tree-of-sets system, and do not find a good clustering \(\mathcal{C}'\) with \(\varphi(C') \leq \varphi(C) - 1\), we make progress by lowering the potential of one of the acceptable clusterings \(C_i\) by at least \(1/n\). Therefore, after polynomially-many iterations, we are guaranteed to complete the phase.
We note that Theorem 6 in [Chu12] provides an algorithm, that, given a cluster $X'_i$, and an access to an oracle for computing $(k, \alpha_{BW})$-violating cuts, produces a partition $W_i$ of $X'_i$ into clusters that have the $(k, \alpha_{BW})$-bandwidth property, with the number of edges connecting different clusters suitably bounded. The bound on the number of edges is computed by using a charging scheme. The potential function that we use here, whose definition may appear non-intuitive, is modeled after this charging scheme.

In the following subsections, we provide a formal proof of Theorem 4.1. We start by defining the different types of clusterings that we use and the potential function, and analyze its properties. We then turn to describe the algorithm itself.

### 5.2 Vertex Clusterings and Legal Contracted Graphs

Let $n = V(G)$. Our algorithm uses a parameter $r_0 = r^2$. We use the following two parameters for the bandwidth property: $\alpha = \frac{1}{2^{\Theta(r_0 \log k)}}$ used to perform bandwidth-decomposition of clusters, and $\alpha_{BW} = \frac{\alpha}{\beta_{ARV}} \Omega \left( \frac{1}{r_0 \log k} \right)$ - the value of the bandwidth parameter we achieve. Finally, we use a parameter $h_0 = \frac{k}{102r^3 \log k}$. We say that a cluster $C \subseteq V(G)$ is large iff $|\text{out}(C)| \geq h_0$, and we say that it is small otherwise. From the statement of the theorem, we can assume that $h_0 > \Delta$, and:

$$h = O \left( \frac{h_0 \alpha^2}{r^9 \Delta^8 \log k} \right).$$

Next, we define acceptable and good vertex clusterings and legal contracted graphs, exactly as in [CL12].

**Definition 5.1** Given a partition $C$ of the vertices of $V(G)$ into clusters, we say that $C$ is an acceptable clustering of $G$ iff:

- Every terminal $t \in T$ is in a separate cluster, that is, $\{t\} \in C$;
- Each small cluster $C \in C$ has the the $\alpha_{BW}$-bandwidth property; and
- For each large cluster $C \in C$, $G[C]$ is connected.

An acceptable clustering that contains no large clusters is called a good clustering.

**Definition 5.2** Given a good clustering $C$ of $G$, a graph $H_C$ is a legal contracted graph of $G$ associated with $C$, iff we can obtain $H_C$ from $G$ by contracting every $C \in C$ into a super-node $v_C$. We remove all self-loops, but we do not remove parallel edges. (Note that the terminals are not contracted since each terminal has its own cluster).

**Claim 5.1** If $G'$ is a legal contracted graph for $G$, then $G' \setminus T$ contains at least $k/3$ edges.

**Proof:** For each terminal $t \in T$, let $e_t$ be the unique edge adjacent to $t$ in $G'$, and let $u_t$ be the other endpoint of $e_t$. We partition the terminals in $T$ into groups, where two terminals $t, t'$ belong to the same group iff $u_t = u_{t'}$. Let $G$ be the resulting partition of the terminals. Since the degree of every vertex in $G'$ is at most $h_0$, each group $U \in G$ contains at most $h_0$ terminals. Next, we partition the terminals in $T$ into two subsets $X, Y$, where $|X|, |Y| \geq k/3$, and for each group $U \in G$, either $U \subseteq X$, or $U \subseteq Y$ holds. We can find such a partition by greedily processing each group $U \in G$, and adding
all terminals of $U$ to one of the subsets $X$ or $Y$, that currently contains fewer terminals. Finally, we remove terminals from set $X$ until $|X| = k/3$, and we do the same for $Y$. Since the set $T$ of terminals is node-well-linked in $G$, it is 1-edge-well-linked in $G'$, so we can route $k/3$ flow units from $X$ to $Y$ in $G'$, with no edge-congestion. Since no group $U$ is split between the two sets $X$ and $Y$, each flow-path must contain at least one edge of $G' \setminus T$. Therefore, $|E(G' \setminus T)| \geq k/3$.

Given any partition $C$ of the vertices of $G$, we define a potential $\varphi(C)$ for this clustering, exactly as in [CL12]. The idea is that $\varphi(C)$ will serve as a tight bound on the number of edges connecting different clusters in $C$. At the same time, the potential function is designed in such a way, that we can perform a number of useful operations on the current clustering, without increasing the potential.

Suppose we are given any partition $C$ of the vertices of $G$. We define $\varphi(C)$ as $\sum_{e \in E} \varphi(C, e)$ where $\varphi(C, e)$ assigns a potential to each edge $e$; to avoid notational overload we use $\varphi(e)$ for $\varphi(C, e)$. If both endpoints of $e$ belong to the same cluster of $C$, then we set its potential $\varphi(e) = 0$. Otherwise, if $e = (u, v)$, and $u \in C$ with $|\text{out}(C)| = z$, while $v \in C'$ with $|\text{out}(C')| = z'$, then we set $\varphi(e) = 1 + \rho(z) + \rho(z')$ where $\rho$ is a non-decreasing real-valued function that we define below. We think of $\rho(z)$ as the contribution of $u$, and $\rho(z')$ the contribution of $v$ to $\varphi(e)$. The function $\rho$ will be chosen to give a small (compared to 1) contribution to $\varphi(e)$ depending on the out-degree of the clusters that $e$ connects.

For any integer $z > 0$, we define a potential $\rho(z)$, as follows. For $z < h_0$, $\rho(z) = 4\alpha \log z$. In order to define $\rho(z)$ for $z \geq h_0$, we consider the sequence $\{n_0, n_1, \ldots\}$ of numbers, where $n_i = \left(\frac{3}{2}\right)^i h_0$. The potentials for these numbers are $\rho(n_0) = \rho(h_0) = 4\alpha \log h_0 + 4\alpha$, and for $i > 0$, $\rho(n_i) = 4\frac{\alpha h_0}{n_i} + \rho(n_{i-1})$. Notice that for all $i$, $\rho(n_i) \leq 12\alpha + 4\alpha \log h_0 \leq 8\alpha \log h_0 \leq \frac{1}{2^i h_0}$. We now partition all integers $z > h_0$ into sets $Z_1, Z_2, \ldots$, where set $Z_i$ contains all integers $z$ with $n_{i-1} \leq z < n_i$. For $z \in Z_i$, we define $\rho(z) = \rho(n_{i-1})$. This finishes the definition of $\rho$. Clearly, for all $z$, $\rho(z) \leq \frac{1}{2^i h_0}$.

**Observation 5.1** For any partition $C$ of the vertices of $G$ and for any edge $e = (u, v) \in E(G)$, if $u, v$ belong to the same cluster of $C$, then $\varphi(e) = 0$. Otherwise, $1 \leq \varphi(e) \leq 1.1$.

Suppose we are given any partition $C$ of $V(G)$. The following theorem allows us to partition any small cluster $C$ into a collection of sub-clusters, each of which has the $\alpha_{\text{BW}}$-bandwidth property, without increasing the overall potential. We call this procedure a bandwidth decomposition.

**Theorem 5.1** Let $C$ be any partition of $V(G)$, and let $C \in C$ be any small cluster, such that $G[C]$ is connected. Then there is an efficient algorithm that finds a partition $W$ of $C$ into small clusters, such that each cluster $R \in W$ has the $\alpha_{\text{BW}}$-bandwidth property, and additionally, if $C'$ is a partition obtained from $C$ by removing $C$ and adding the clusters of $W$ to it, then $\varphi(C') \leq \varphi(C)$.

**Proof:** We maintain a partition $W$ of $C$ into small clusters, where at the beginning, $W = \{C\}$. We then perform a number of iterations.

In each iteration, we select a cluster $S \in W$, and set up the following instance of the sparsest cut problem. Let $\Gamma$ be the set of the interface vertices of $S$ in $G$. We then consider the graph $G[S]$, where the vertices of $\Gamma$ serve as terminals. We run the algorithm $A_{\text{ARV}}$ on the resulting instance of the sparsest cut problem. If the sparsity of the cut produced by the algorithm is less than $\alpha$, then we obtain a partition $(X, Y)$ of $S$, with $|E(X, Y)| < \alpha \cdot \min \{|\Gamma \cap X|, |\Gamma \cap Y|\} \leq \alpha \cdot \min \{|\text{out}(S) \cap \text{out}(X)|, |\text{out}(S) \cap \text{out}(Y)|\}$. In this case, we remove $S$ from $W$, and add $X$ and $Y$ to $W$ instead. Notice that since $S$ is a small cluster, $X$ and $Y$ are also small clusters. The algorithm ends when for every cluster $S \in W$, algorithm $A_{\text{ARV}}$ returns a partition of sparsity at least $\alpha$. We are then guaranteed that every cluster in $W$ has the $\alpha/\beta_{\text{ARV}}(k) = \alpha_{\text{BW}}$-bandwidth property, and it is easy to verify that all resulting clusters are small.
It now only remains to show that the potential does not increase. Each iteration of the algorithm is associated with a partition of the vertices of \( G \), obtained from \( C \) by removing \( C \) and adding all clusters of the current partition \( W \) of \( C \) to it. It suffices to show that if \( C' \) is the current partition of \( V(G) \), and \( C'' \) is the partition obtained after one iteration, where a set \( S \subset C \) was replaced by two sets \( X \) and \( Y \), then \( \varphi(C'') \leq \varphi(C') \).

Assume without loss of generality that \(|\text{out}(X)| \leq |\text{out}(Y)|\), so \(|\text{out}(X)| \leq 2|\text{out}(S)|/3\). Let \( z = |\text{out}(S)|, z_1 = |\text{out}(X)|, z_2 = |\text{out}(Y)| \), and recall that \( z, z_1, z_2 < h_0 \). The changes to the potential are the following:

- The potential of the edges in \( \text{out}(Y) \cap \text{out}(S) \) only goes down.
- The potential of every edge in \( \text{out}(X) \cap \text{out}(S) \) goes down by \( \rho(z) - \rho(z_1) = 4\alpha \log z - 4\alpha \log z_1 = 4\alpha \log z \geq 4\alpha \log 1.5 \geq 2.3\alpha \), since \( z_1 \leq 2z/3 \). So the total decrease in the potential of the edges in \( \text{out}(X) \cap \text{out}(S) \) is at least \( 2.3\alpha \cdot |\text{out}(X) \cap \text{out}(S)| \).
- The edges in \( E(X, Y) \) did not contribute to the potential initially, and now contribute \( 1 + \rho(z_1) + \rho(z_2) \leq 2 \) each. Notice that \(|E(X, Y)| \leq \alpha \cdot |\text{out}(X) \cap \text{out}(S)| \), and so they contribute at most \( 2\alpha \cdot |\text{out}(X) \cap \text{out}(S)| \) in total.

Clearly, the overall potential decreases.

Assume that we are given an acceptable clustering \( C \) of \( G \). We now define two operations on \( G \), each of which produces a new acceptable clustering of \( G \), whose potential is strictly smaller than \( \varphi(C) \).

**Action 1: Partitioning a large cluster.** Suppose we are given a large cluster \( C \), and let \( \Gamma \) be the interface of \( C \) in \( G \). We say that a partition \((X, Y)\) of \( C \) is an \((h_0, \alpha)\)-violating partition, iff there are two subsets \( \Gamma_X \subseteq \Gamma \cap X, \Gamma_Y \subseteq \Gamma \cap Y \) of vertices, with \(|\Gamma_X| + |\Gamma_Y| \leq h_0 \), and \(|E(X, Y)| < \alpha \cdot \min \{|\Gamma_X|, |\Gamma_Y|\} \).

Suppose we are given an acceptable clustering \( C \) of \( G \), a large cluster \( C \subseteq C \), and an \((h_0, \alpha)\)-violating partition \((X, Y)\) of \( C \). In order to perform this operation, we first replace \( C \) with \( X \) and \( Y \) in \( C \). If, additionally, any of the clusters, \( X \) or \( Y \), become small, then we perform a bandwidth decomposition of that cluster using Theorem 5.1 and update \( C \) with the resulting partition. Clearly, the final partitioning \( C' \) is an acceptable clustering. We denote this operation by \( \text{PARTITION}(C, X, Y) \).

**Claim 5.2** Let \( C' \) be the outcome of operation \( \text{PARTITION}(C, X, Y) \). Then \( \varphi(C') < \varphi(C) - 1/n \).

**Proof:** Let \( C'' \) be the clustering obtained from \( C \), by replacing \( C \) with \( X \) and \( Y \). From Theorem 5.1 it is sufficient to prove that \( \varphi(C'') < \varphi(C) - 1/n \).

Assume without loss of generality that \(|\text{out}(X)| \leq |\text{out}(Y)|\). Let \( z = |\text{out}(C)|, z_1 = |\text{out}(X)|, z_2 = |\text{out}(Y)| \), so \( z_1 < 2z/3 \). Assume that \( z \in Z_i \). Then either \( z \in Z_i \) for \( i' \leq i - 1 \), or \( z < h_0 \). The potential of the edges in \( \text{out}(Y) \cap \text{out}(C) \) does not increase. The only other changes in the potential are the following: the potential of each edge in \( \text{out}(X) \cap \text{out}(C) \) decreases by \( \rho(z) - \rho(z_1) \), and the potential of every edge in \( E(X, Y) \) increases by \( 0 \) to at most \( 1.1 \). We consider two cases.

First, if \( z_1 < h_0 \), then \( \rho(z) \geq 4\alpha + \rho(z_1) \). So the potential of each edge in \( \text{out}(X) \cap \text{out}(C) \) decreases by at least \( 4\alpha \), and the overall decrease in potential due to these edges is at least \( 4\alpha |\text{out}(X) \cap \text{out}(C)| \).

The total increase in potential due to the edges in \( E(X, Y) \) is bounded by \( 1.1|E(X, Y)| < 1.1\alpha \cdot |\Gamma_X| \leq 1.1\alpha |\text{out}(X) \cap \text{out}(C)| \), so the overall potential decreases by at least \( 2\alpha |\text{out}(X) \cap \text{out}(C)| > 1/n \)

The second case is when \( z_1 \geq h_0 \). Assume that \( z_1 \in Z_{i'} \). Then \( n_{i'} \leq 3z_1/2 \), and, since \( i' \leq i - 1 \) must hold, \( \rho(z) \geq \frac{4\alpha h_0}{n_{i'}} + \rho(n_{i'-1}) = \frac{4\alpha h_0}{n_{i'}} + \rho(z_1) \geq \frac{8\alpha h_0}{3z_1} + \rho(z_1) \). So the potential of each edge in
out(\(X\)) \cap \text{out}(\(C\)) decreases by at least \(\frac{8\alpha h_0}{351}\), and the total decrease in potential due to these edges is at least \(\frac{8\alpha h_0}{351} \cdot |\text{out}(\(X\)) \cap \text{out}(\(C\))| \geq \frac{4\alpha h_0}{351}\), since \(|\text{out}(\(X\)) \cap \text{out}(\(C\))| \geq z_1/2\). The total increase in the potential due to the edges in \(E(\(X\), \(Y\))\) is bounded by \(1.1|E(\(X\), \(Y\))| < 0.55\alpha h_0\), since \(|E(\(X\), \(Y\))| \leq \alpha h_0/2\). Overall, the total potential decreases by at least \(\frac{2\alpha h_0}{3} > 1/n\).

\[\square\]

**Action 2: Separating a large cluster.** Let \(\mathcal{C}\) be any acceptable clustering, and let \(C \in \mathcal{C}\) be a large cluster in \(\mathcal{C}\). Assume further that we are given a partition \((A, B)\) of \(V(G)\), with \(C \subseteq A\), \(\mathcal{T} \subseteq B\), and \(|E_G(A, B)| < h_0/2\). We perform the following operation, that we denote by \(\text{SEPARATE}(C, A)\).

Consider some cluster \(S \in \mathcal{C}\). If \(S\) is a small cluster, but \(S \setminus A \neq \emptyset\), and it is a large cluster, then we modify \(A\) by removing all vertices of \(S\) from it. Notice that in this case, the number of edges in \(E(S)\) that originally contributed to the cut \((A, B), |E(S \cap A, S \cap B)| > |\text{out}(S) \cap E(A)|\) must hold, so \(|\text{out}(A)|\) only goes down as a result of this modification. We assume from now on that if \(S \in \mathcal{C}\) is a small cluster, and \(S \setminus A \neq \emptyset\), then \(S \setminus A\) is also a small cluster. We build a new partition \(\mathcal{C}'\) of \(V(G)\) as follows. First, we add every connected component of \(G[A]\) to \(\mathcal{C}\). Notice that all these clusters are small, as \(|\text{out}(A)| < h_0/2\). Next, for every cluster \(S \in \mathcal{C}\), such that \(S \setminus A \neq \emptyset\), we add every connected component of \(G[S \setminus A]\) to \(\mathcal{C}'\). Notice that every terminal \(t \in \mathcal{T}\) is added as a separate cluster to \(\mathcal{C}'\). So far we have defined a new partition \(\mathcal{C}'\) of \(V(G)\). This partition may not be acceptable, since we are not guaranteed that every small cluster of \(\mathcal{C}'\) has the bandwidth property. In our final step, we perform the bandwidth decomposition of every small cluster of \(\mathcal{C}'\), using Theorem 5.1 and obtain the final acceptable partition \(\mathcal{C}''\) of vertices of \(G\). Notice that if \(S \in \mathcal{C}''\) is a large cluster, then there must be some large cluster \(S'\) in the original partition \(\mathcal{C}\) with \(S \subseteq S'\).

**Claim 5.3** Let \(\mathcal{C}''\) be the outcome of operation \(\text{SEPARATE}(C, A)\). Then \(\varphi(\mathcal{C}'') \leq \varphi(\mathcal{C}) - 1\).

**Proof:** In order to prove the claim, it is enough to prove that \(\varphi(\mathcal{C}') \leq \varphi(\mathcal{C}) - 1\), since, from Theorem 5.1, bandwidth decompositions of small clusters do not increase the potential.

We now show that \(\varphi(\mathcal{C}') \leq \varphi(\mathcal{C}) - 1\). We can bound the changes in the potential as follows:

- Every edge in \(\text{out}(A)\) contributes at most 1.1 to the potential of \(\mathcal{C}''\), and there are at most \(\frac{h_0 - 1}{2}\) such edges. These are the only edges whose potential in \(\mathcal{C}''\) may be higher than their potential in \(\mathcal{C}\).

- Every edge in \(\text{out}(C)\) contributed at least 1 to the potential of \(\mathcal{C}'\), and there are at least \(h_0\) such edges, since \(C\) is a large cluster.

Therefore, the decrease in the potential is at least \(h_0 - \frac{1.1(h_0 - 1)}{2} \geq 1\).

To summarize, given any acceptable clustering \(\mathcal{C}\) of the vertices of \(G\), let \(E'\) be the set of edges whose endpoints belong to distinct clusters of \(\mathcal{C}\). Then \(|E'| \leq \varphi(\mathcal{C}) \leq 1.1|E'|\). So the potential is a good estimate on the number of edges connecting the different clusters. We have also defined two actions on large clusters of \(\mathcal{C}\): \(\text{PARTITION}(C, X, Y)\) can be performed if we are given a large cluster \(C \in \mathcal{C}\) and an \((h_0, \alpha)\)-violating partition \((X, Y)\) of \(C\), and \(\text{SEPARATE}(C, A)\), where \((A, V(G) \setminus A)\) is a partition of \(V(G)\) with \(|\text{out}(A)| < h_0/2\), separating a large cluster \(C\) from the terminals. Each such action returns a new acceptable clustering, whose potential goes down by at least \(1/n\). Both operations ensure that if \(S\) is any large cluster in the new clustering, then there is some large cluster \(S'\) in the original clustering with \(S \subseteq S'\).
5.3 The Algorithm

We maintain, throughout the algorithm, a good clustering \( C \) of \( G \). Initially, \( C \) is a partition of \( V(G) \), where every vertex of \( G \) belongs to a distinct cluster, that is, \( C = \{ \{ v \} \mid v \in V(G) \} \). Clearly, this is a good clustering, as \( \Delta < h_0 \). The algorithm consists of a number of phases. In every phase, we start with some good clustering \( C \) and the corresponding legal contracted graph \( H_C \). The phase output is either a good tree-of-sets system, or another good clustering \( C' \), such that \( \varphi(C') \leq \varphi(C) - 1 \). In the former case, we terminate the algorithm, and output the tree-of-sets system. In the latter case, we continue to the next phase. After \( O(|E(G)|) \) phases, our algorithm will then successfully terminate with a good tree-of-sets system. It is therefore enough to prove the following theorem.

**Theorem 5.2** Let \( C \) be any good clustering of the vertices of \( G \), and let \( H_C \) be the corresponding legal contracted graph. Then there is an efficient randomized algorithm that with high probability either computes a good tree-of-sets system, or finds a new good clustering \( C' \), such that \( \varphi(C') \leq \varphi(C) - 1 \).

The rest of this subsection is dedicated to proving Theorem 5.2. We assume that we are given a good clustering \( C \) of the vertices of \( G \), and the corresponding legal contracted graph \( G' = H_C \).

Let \( m = |E(G' \setminus T)| \). From Claim 5.1 \( m \geq k/3 \). As a first step, we randomly partition the vertices in \( G' \setminus T \) into \( r_0 \) subsets, where each vertex \( v \in (V(G') \setminus T) \) selects an index \( 1 \leq j \leq r_0 \) independently uniformly at random, and is then added to \( X_j \). We need the following claim.

**Claim 5.4** With probability at least \( 1/2 \), for each \( 1 \leq j \leq r_0 \), \( |\text{out}_{G'}(X_j)| < \frac{10m}{r_0} \), while \( |E_{G'}(X_j)| \geq \frac{m}{2r_0} \).

**Proof:** Let \( H = G' \setminus T \). Fix some \( 1 \leq j \leq r_0 \). Let \( E_1(j) \) be the bad event that \( \sum_{v \in X_j} d_H(v) > \frac{2m}{r_0} \left( 1 + \frac{1}{r_0} \right) \). In order to bound the probability of \( E_1(j) \), we define, for each vertex \( v \in V(H) \), a random variable \( x_v \), whose value is \( \frac{d_H(v)}{h_0} \) if \( v \in X_j \) and 0 otherwise. Notice that \( x_v \in [0,1] \), and the random variables \( \{ x_v \}_{v \in V(H)} \) are pairwise independent. Let \( B = \sum_{v \in V(H)} x_v \). Then the expectation of \( B \), \( \mu_1 = \sum_{v \in V(H)} \frac{d_G(v)}{h_0 r_0} = \frac{2m}{r_0 h_0} \). Using the standard Chernoff bound (see e.g. Theorem 1.1 in [DP00]),

\[
\Pr[E_1(j)] = \Pr[B > (1 + 1/r_0) \mu_1] \leq e^{-\mu_1/(3r_0^2)} = e^{-\frac{2m}{3r_0^2 h_0}} < \frac{1}{6r_0}
\]

since \( m \geq k/3 \) and \( h_0 = \frac{k}{192r_0^3 \log k} \).

For each terminal \( t \in T \), let \( e_t \) be the unique edge adjacent to \( t \) in graph \( G' \), and let \( u_t \) be its other endpoint. Let \( U = \{ u_t \mid t \in T \} \). For each vertex \( u \in U \), let \( w(u) \) be the number of terminals \( t \), such that \( u = u_t \). Notice that \( w(u) \leq h_0 \) must hold. We say that a bad event \( E_2(j) \) happens iff \( \sum_{u \in U \cap X_j} w(u) \geq \frac{k}{r_0} \left( 1 + \frac{1}{r_0} \right) \). In order to bound the probability of the event \( E_2(j) \), we define, for each \( u \in U \), a random variable \( y_u \), whose value is \( w(u)/h_0 \) if \( u \in X_j \), and is 0 otherwise. Notice that \( y_u \in [0,1] \), and the variables \( y_u \) are independent for all \( u \in U \). Let \( Y = \sum_{u \in U} y_u \). The expectation of \( Y \) is \( \mu_2 = \frac{k}{h_0 r_0} \), and event \( E_2(j) \) holds iff \( Y \geq \frac{k}{h_0 r_0} \cdot \left( 1 + \frac{1}{r_0} \right) \geq \mu_2 \cdot \left( 1 + \frac{1}{r_0} \right) \). Using the standard Chernoff bound again, we get that:

\[
\Pr[E_2(j)] \leq e^{-\mu_2/(3r_0^2)} \leq e^{-k/(3h_0^3r_0^2)} \leq \frac{1}{6r_0}
\]

since \( h_0 = \frac{k}{192r_0^3 \log k} \). Notice that if events \( E_1(j), E_2(j) \) do not hold, then:
\[ |\text{out}_G'(X_j)| \leq \sum_{v \in X_j} d_H(v) + \sum_{u \in U \cap X_j} w(u) \leq \left(1 + \frac{1}{r_0}\right) \left(\frac{2m}{r_0} + \frac{k}{r_0}\right) < \frac{10m}{r_0} \]

since \( m \geq k/3 \).

Let \( \mathcal{E}_3(j) \) be the bad event that \( |E_G'(X_j)| < \frac{m}{2r_0} \). We next prove that \( \Pr[\mathcal{E}_3(j)] \leq \frac{1}{6r_0} \). We say that two edges \( e, e' \in E(G' \setminus \mathcal{T}) \) are independent if they do not share any endpoints. Our first step is to compute a partition \( U_1, \ldots, U_z \) of the set \( E(G' \setminus \mathcal{T}) \) of edges, where \( z \leq 2h_0 \), such that for each \( 1 \leq i \leq z \), \( |U_i| \geq \frac{m}{4h_0} \), and all edges in set \( U_i \) are mutually independent. In order to compute such a partition, we construct an auxiliary graph \( Z \), whose vertex set is \( \{v_e \mid e \in E(H)\} \), and there is an edge \( (v_e, v_{e'}) \) iff \( e \) and \( e' \) are not independent. Since the maximum vertex degree in \( G' \) is at most \( h_0 \), the maximum vertex degree in \( Z \) is bounded by \( 2h_0 - 2 \). Using the Hajnal-Szemerédi Theorem [HS70], we can find a partition \( V_1, \ldots, V_{r_0} \) of the vertices of \( Z \) into \( z \leq 2h_0 \) subsets, where each subset \( V_i \) is an independent set, and \( |V_i| \geq \frac{|V(Z)| - 1}{2} \geq \frac{m}{8h_0} \). The partition \( V_1, \ldots, V_{r_0} \) of the vertices of \( Z \) gives the desired partition \( U_1, \ldots, U_z \) of the edges of \( G' \setminus \mathcal{T} \). For each \( 1 \leq i \leq r_0 \), we say that the bad event \( \mathcal{E}_3^i(j) \) happens if \( |U_i \cap E(X_j)| < \frac{|U_i|}{2r_0} \). Notice that if \( \mathcal{E}_3(j) \) happens, then event \( \mathcal{E}_3^i(j) \) must happen for some \( 1 \leq i \leq z \). Fix some \( 1 \leq i \leq z \). The expectation of \( |U_i \cap E(X_j)| \) is \( \mu_3 = \frac{|U_i|}{r_0} \). Since all edges in \( U_i \) are independent, we can use the standard Chernoff bound to bound the probability of \( \mathcal{E}_3^i(j) \), as follows:

\[ \Pr[\mathcal{E}_3^i(j)] = \Pr[|U_i \cap E(X_j)| < \frac{\mu_3}{2}] \leq e^{-\mu_3/8} = e^{-\frac{|U_i|}{8r_0}}. \]

Since \( |U_i| \geq \frac{m}{4h_0} \), \( m \geq k/3 \), \( h_0 = \frac{k}{192r_0 \log k} \), this is bounded by \( \frac{1}{k^2} \leq \frac{1}{12h_0r_0} \). We conclude that \( \Pr[\mathcal{E}_3(j)] \leq \frac{1}{12h_0r_0} \), and by using the union bound over all \( 1 \leq i \leq z \), \( \Pr[\mathcal{E}_3(j)] \leq \frac{1}{6r_0} \).

Using the union bound over all \( 1 \leq j \leq r_0 \), with probability at least \( \frac{1}{7} \), none of the events \( \mathcal{E}_1(j), \mathcal{E}_2(j), \mathcal{E}_3(j) \) for \( 1 \leq j \leq r_0 \) happen, and so for each \( 1 \leq j \leq r_0 \), \( |\text{out}_G'(X_j)| < \frac{10m}{r_0} \), and \( |E_G'(X_j)| \geq \frac{m}{2r_0} \). \( \square \)

Given a partition \( X_1, \ldots, X_{r_0} \), we can efficiently check whether the conditions of Claim 5.4 hold. If they do not hold, we repeat the randomized partitioning procedure. From Claim 5.4 we are guaranteed that with high probability, after \( \text{poly}(n) \) iterations, we will obtain a partition with the desired properties. Assume now that we are given the partition \( X_1, \ldots, X_{r_0} \) of \( V(G') \setminus \mathcal{T} \), for which the conditions of Claim 5.4 hold. Then for each \( 1 \leq j \leq r_0 \), \( |E_G'(X_j)| > \frac{|\text{out}_G'(X_j)|}{2h_0} \). Let \( X'_j \subseteq V(G) \setminus \mathcal{T} \) be the set obtained from \( X_j \), after we un-contract each cluster, that is, for each super-node \( v_C \in X_j \), we replace \( v_C \) with the vertices of \( C \). Notice that \( \{X'_j\}_{j=1}^{r_0} \) is a partition of \( V(G) \setminus \mathcal{T} \).

The plan for the rest of the proof is as follows. For each \( 1 \leq j \leq r_0 \), we will maintain an acceptable clustering \( C_j \) of the vertices of \( G \). That is, for each \( 1 \leq j \leq r_0 \), \( C_j \) is a partition of \( V(G) \). In addition to being an acceptable clustering, it will have the following property:

**P1.** If \( C \in C_j \) is a large cluster, then \( C \subseteq X'_j \).

The initial partition \( C_j \), for \( 1 \leq j \leq r_0 \) is obtained as follows. Recall that \( C \) is the current good clustering of the vertices of \( G \), and every cluster \( C \in C \) is either contained in \( X'_j \), or it is disjoint from it. First, we add to \( C_j \) all clusters \( C \in C \) with \( C \cap X'_j = \emptyset \). Next, we add to \( C_j \) all connected components of \( G[X'_j] \). If any of these components is a small cluster, then we perform the bandwidth decomposition of this cluster, using Theorem 5.1 and update \( C_j \) accordingly. Let \( C_j \) be the resulting
final partition. Clearly, it is an acceptable clustering, with property 1. Moreover, the following claim shows that ϕ(C_j) ≤ ϕ(C) − 1:

Claim 5.5 For each 1 ≤ j ≤ r_0, ϕ(C_j) ≤ ϕ(C) − 1.

Proof: Let C_j' be the partition of V(G), obtained as follows: we add to C_j all clusters C ∈ C with C ∩ X'_j = ∅, and we add all connected components of G[X'_j] to C_j' (that is, C_j' is obtained like C_j, except that we do not perform the bandwidth decompositions of the small clusters). From Theorem 5.1, it is enough to prove that ϕ(C_j') ≤ ϕ(C) − 1. The changes of the potential from C to C_j' can be bounded as follows:

- The edges in E_{G'}(X_j) contribute at least 1 to ϕ(C) and contribute 0 to ϕ(C_j').
- The potential of the edges in out_G(X'_j) may increase. The increase is at most ρ(n) ≤ 1/(2πr_0) per edge. So the total increase is at most |out_G(X'_j)| ≤ |E_{G'}(X_j)|. These are the only edges whose potential may increase.

Overall, the decrease in the potential is at least \( \frac{|E_{G}(X_j)|}{2} \geq \frac{m}{4r_0} \geq \frac{k}{12\pi r_0} \geq 1. \)

If any of the partitions C_1, ..., C_{r_0} is a good clustering, then we have found a good clustering C' with ϕ(C') ≤ ϕ(C) − 1. We terminate the algorithm and return C'. Otherwise, we select an arbitrary large cluster S_j ∈ C_j for each j. We then consider the resulting collection S_1, ..., S_{r_0} of large clusters, and try to exploit them to construct a good tree-of-sets system. Since for each 1 ≤ j ≤ r_0, S_j ⊆ X'_j, the sets S_1, ..., S_{r_0} are mutually disjoint and they do not contain terminals. Our algorithm performs a number of iterations, using the following theorem.

Theorem 5.3 Suppose we are given a collection \{S_1, ..., S_{r_0}\} of disjoint vertex subsets of G, where for all 1 ≤ j ≤ r_0, S_j ∩ T = ∅. Then there is an efficient randomized algorithm, that with high probability computes one of the following:

- either a good tree-of-sets system in G;
- or an \((h_0, α)\)-violating partition \((X, Y)\) of S_j, for some 1 ≤ j ≤ r_0;
- or a partition \((A, B)\) of V(G) with S_j ⊆ A, T ⊆ B and |E_G(A, B)| < h_0/2, for some 1 ≤ j ≤ r_0.

We provide the proof of Theorem 5.3 in the following subsection, and complete the proof of Theorem 5.2 here. Suppose we are given a good clustering C of the vertices of G. For each 1 ≤ j ≤ r_0, we compute an acceptable clustering C_j of V(G) as described above. If any of the partitions C_j is a good clustering, then we terminate the algorithm and return C_j. From the above discussion, ϕ(C_j) ≤ ϕ(C) − 1. Otherwise, for each 1 ≤ j ≤ r_0, we select any large cluster S_j ∈ C_j, and apply Theorem 5.3 to the current family \{S_1, ..., S_{r_0}\} of large clusters. If the outcome of Theorem 5.3 is a good tree-of-sets system, then we terminate the algorithm and return this tree-of-sets system, and we say that the iteration is successful. Otherwise, we apply the appropriate action: \textsc{Partition}(S_j, X, Y), or \textsc{Separate}(S_j, A) to the clustering C_j. As a result, we obtain an acceptable clustering C_j', with ϕ(C_j') ≤ ϕ(C_j) − 1/n. Moreover, it is easy to see that this clustering also has Property 1: if the \textsc{Partition} operation is performed, then we only partition S_j; if the \textsc{Separate} operation is performed, then for every large cluster S in the new partition C_j', there is a large cluster S' ∈ C_j with S ⊆ S'.

If all clusters in C_j' are small, then we can again terminate the algorithm with a good clustering C_j', with ϕ(C_j') ≤ ϕ(C_j) − 1/n ≤ ϕ(C) − 1 − 1/n (recall that Claim 5.5 shows that ϕ(C_j) ≤ ϕ(C) − 1). Otherwise,
we select any large cluster $S_j' \in \mathcal{C}_j'$, and continue to the next iteration. Overall, as long as we do not complete a successful iteration, and we do not find a good clustering $\mathcal{C}'$ of $V(G)$ with $\varphi(\mathcal{C}') \leq \varphi(\mathcal{C}) - 1$, we make progress in each iteration by decreasing the potential of one of the partitions $\mathcal{C}_j$ by at least $1/n$, by performing either a SEPARATE or a PARTITION operation on one of the large clusters of $\mathcal{C}_j$. After at most $1.1|E(G)| \cdot n \cdot r_0$ iterations we are then guaranteed to complete a successful iteration, or find a good clustering $\mathcal{C}'$ with $\varphi(\mathcal{C}') \leq \varphi(\mathcal{C}) - 1$, and finish the algorithm. Therefore, in order to complete the proof of Theorem 5.2 it is now enough to prove Theorem 5.3.

### 5.4 Proof of Theorem 5.3

Let $\mathcal{R} = \{S_1, \ldots, S_{r_0}\}$. We start by checking that for each $1 \leq j \leq r_0$, the vertices of $S_j$ can send $h_0/2$ flow units in $G$ to the terminals with no edge-congestion. If this is not the case for some set $S_j$, then there is a partition $(A, B)$ of $V(G)$ with $S_j \subseteq A$, $\mathcal{T} \subseteq B$ and $|E_G(A, B)| < h_0/2$. We then return the partition $(A, B)$ of $V(G)$ and finish the algorithm. From now on we assume that each set $S_j$ can send $h_0/2$ flow units in $G$ to the terminals with no edge-congestion.

Since the set $\mathcal{T}$ of terminals is node-well-linked, every pair $(S_j, S_{j'})$ of vertex subsets from $\mathcal{R}$ can send $h_0/2$ flow units to each other with edge-congestion at most 3: concatenate the flows from $S_j$ to a subset $\mathcal{T}_1$ of the terminals, from $S_{j'}$ to a subset $\mathcal{T}_2$ of the terminals, and the flow between the two subsets of the terminals. Scaling this flow down by factor $3\Delta$ and using the integrality of the $s$-$t$ flow, for each such pair $(S_j, S_{j'})$, there are at least $\lceil \frac{h_0}{2}\rceil$ node-disjoint paths connecting $S_j$ to $S_{j'}$ in $G$. We can assume that these paths do not contain any terminals, as the degree of every terminal in $G$ is 1.

The algorithm consists of two phases. In the first phase, we attempt to construct a tree-of-sets system, using the collection $\mathcal{R}$ of vertex subsets. If we fail to do so, we will either return an $(h_0, \alpha)$-violating cut in some cluster $S_j \in \mathcal{R}$, or we will identify a subset $\mathcal{R}' \subset \mathcal{R}$ of $r$ clusters, and for each cluster $S_j \in \mathcal{R}'$, a large subset $E_j \subset \text{out}(S_j)$ of edges, such that for each $S_i, S_j \in \mathcal{R}'$, there are many disjoint paths connecting the edges in $E_j$ to the edges in $E_i$ in $G$. In the second phase, we exploit the clusters in $\mathcal{R}'$ to build the tree-of-sets system.

Given any graph $G$ and a subset $\tilde{T}$ of vertices called terminals, we say that a pair $(t, t')$ of terminals is $\lambda$-edge-connected, iff there are at least $\lambda$ paths connecting $t$ to $t'$ in $G$, that are mutually edge-disjoint. Let $\lambda(t, t')$ be the largest value $\lambda$, such that $t$ and $t'$ are $\lambda$-edge-connected, and let $\lambda_G(\tilde{T}) = \min_{t, t' \in \tilde{T}} \lambda(t, t')$. We say that a pair $t, t'$ of terminals is $\mu$-element-connected, iff there are $\mu$ paths connecting $t$ to $t'$ that are pairwise disjoint in both the edges and the non-terminal vertices of $G$ (but they are allowed to share terminals). Let $\mu(t, t')$ be the largest value $\mu$, such that $t$ and $t'$ are $\mu$-element-connected, and denote $\mu_G(\tilde{T}) = \min_{t, t' \in \tilde{T}} \mu(t, t')$. Clearly, $\lambda_G(\tilde{T}) \geq \mu_G(\tilde{T})$ always holds. We use the following theorem several times.

**Theorem 5.4** Let $G$ be any graph, and $\tilde{T}$ a set of $\kappa$ vertices called terminals in $G$, such that $\mu_G(\tilde{T}) \geq \mu$, for some $\mu \geq 1$. Then there is an efficient algorithm to construct another graph $H$, whose vertex set is $\tilde{T}$, and for each edge $e = (t, t') \in E(H)$, find a path $P_e$ connecting $t$ to $t'$ in $G$, such that:

- $\lambda_H(\tilde{T}) \geq 2\mu$;
- For each terminal $t$, $d_H(t) \leq 2d_G(t)$;
- For each $e \in E(H)$, path $P_e$ does not contain terminals as inner vertices.
- We can compute a partition $\mathcal{U}$ of $E(H)$ into groups of size at most $\kappa$, such that, if we select, for each group $U \in \mathcal{U}$, an arbitrary edge $e_U \in U$, then the corresponding paths $\{P_{e_U} \mid U \in \mathcal{U}\}$ are disjoint in $G$, except for possibly sharing endpoints.
Proof: We use the following theorem of Hind and Oellermann [HO96] (see also [CK09]).

**Theorem 5.5** Let \( G \) be any graph, \( \mathcal{T} \) a set of terminals in \( G \), and assume that \( \mu_G(\mathcal{T}) = \mu \) for some \( \mu \geq 0 \). Let \((p,q)\) be any edge with \( p, q \in V \setminus \mathcal{T} \). Let \( G_1 \) be the graph obtained from \( G \) by deleting the edge \((p,q)\), and let \( G_2 \) be obtained from \( G \) by contracting it. Then either \( \mu_{G_1}(\mathcal{T}) \geq \mu \) or \( \mu_{G_2}(\mathcal{T}) \geq \mu \).

While our graph \( G \) contains any edge \((p,q)\) connecting two non-terminal vertices \( p \) and \( q \), we apply Theorem 5.5 to \( G, \mathcal{T} \) and the edge \((p,q)\), and replace \( G \) with the resulting graph, where the edge \((p,q)\) is either deleted or contracted. Let \( G' \) be the graph obtained at the end of this procedure. For simplicity, we call the terminal vertices of \( G' \) black vertices, and the non-terminal vertices white vertices. Let \( W \) denote the set of all white vertices. Notice that every edge in \( G' \) either connects two black vertices, or it connects a white vertex to a black vertex. Moreover, we can assume without loss of generality that for each \( t \in \mathcal{T}, \ v \in W \), there is at most one edge \((t,v)\) in \( G' \); otherwise, if several such parallel edges are present, we delete all but one such edge. This does not affect the element-connectivity of any pair \( t, t' \) of terminals, since the paths connecting them are not allowed to share \( v \). So we will assume from now on that every such pair \((t,v)\) at most one edge \((t,v)\) is present in \( G' \). Notice that for each terminal \( t \), \( d_{G'}(t) \leq d_G(t) \). For every pair \((t,t')\) of terminals, an edge \((t,t')\) is present in \( G' \) iff it was present in \( G \). Every white vertex \( v \) is naturally associated with a connected subgraph \( C_v \) of \( G \), containing all edges that were contracted into \( v \), and all subgraphs \( \{C_v\}_{v \in W} \) are completely disjoint. For each edge \((v,t)\) connecting \( v \) to some terminal \( t \) in \( G' \), there is an edge \((u,t)\) in \( G \), where \( u \) is some vertex in \( C_v \). Notice that \( \lambda_{G'}(\mathcal{T}) \geq \mu_{G'}(\mathcal{T}) \geq \mu \).

Next, we replace every edge in \( G' \) by two parallel edges, and denote the resulting graph by \( G'' \), so \( G'' \) is Eulerian. The degree of every terminal \( t \) now becomes at most \( 2d_G(t) \), and \( \lambda_{G''}(\mathcal{T}) \geq 2\mu \). We now start constructing the final graph \( H \) and the partition \( \mathcal{U} \) of its edges. We start with \( H = G'' \), and for every edge \((t,t') \in E(G')\) connecting a pair \( t,t' \in \mathcal{T} \) of terminals, we add a new group \( U \), containing the two copies of the edge \((t,t')\) in \( G'' \), to \( \mathcal{U} \). Next, we take care of the white vertices, by using the following edge-splitting operation due to Mader [Mad78].

**Theorem 5.6** Let \( G \) be any undirected multi-graph, \( s \) any vertex of \( G \) whose degree is not 3, such that \( s \) is not incident to a cut edge of \( G \). Then \( s \) has two neighbors \( u \) and \( v \), such that the graph \( G' \), obtained from \( G \) by replacing the edges \((s,u)\) and \((s,v)\) with the edge \((u,v)\), satisfies \( \lambda_{G'}(x,y) = \lambda_G(x,y) \) for all \( x, y \neq s \).

We process the white vertices \( v \in W \) one-by-one. Consider some such vertex \( v \). Recall that there are at most \( 2\kappa \) edges incident on \( v \) in \( G'' \). We apply Theorem 5.6 to vertex \( v \) repeatedly, until it becomes an isolated vertex (since the degree of \( v \) is even due to the doubling of all edges, and the terminals are \( 2\mu \)-edge-connected, the conditions of the theorem are always satisfied). Let \( U_v \) be the set of all resulting new edges in graph \( H \). We add \( U_v \) to \( \mathcal{U} \). Notice that \( |U_v| \leq \kappa \). Once all vertices \( v \in W \) are processed, we obtain the final graph \( H \). It is easy to see that the degree of every terminal \( t \in \mathcal{T} \) is at most \( 2d_G(t) \), since the edge-splitting operation does not change the degrees of the terminals.

Every edge \( e = (t,t') \) in \( H \) is naturally associated with a path \( P_e \) connecting \( t \) and \( t' \) in \( G \): if edge \( e = (t,t') \) is present in \( G \), then \( P_e = e \). Otherwise, edge \( e \) was obtained by replacing a pair \((t,v),(v,t')\) of vertices in \( G'' \) with edge \((t,t')\). In this case, there must be vertices \( u,u' \in C_v \) (possibly \( u = u' \)), with edges \( e_1 = (u,t), e_2 = (u',t') \in E(G) \). We let \( P \) be any path connecting \( u \) to \( u' \) in \( C_v \), and set \( P_e = (e_1, P, e_2) \). Given two edges \( e,e' \in E(H) \), the only possibility that the paths \( P_e \) and \( P_{e'} \) share inner vertices or edges is when \( e,e' \) are two copies of the same edge connecting some pair of terminals in \( G'' \), or both edges belong to some set \( U_v \), for some \( v \in W \). Therefore, choosing at most one edge form each group \( U \in \mathcal{U} \) ensures that the resulting paths are internally node- and edge-disjoint. 

30
5.4.1 Phase 1

Let $H$ be the graph obtained from $G$ by contracting each cluster $S_i \in \mathcal{R}$ into a super-node $v_i$. Let $\mathring{T} = \{v_1, \ldots, v_r\}$ be the resulting set of super-nodes that we will refer to as terminals in this phase. As observed above, every pair $S_i, S_j \in \mathcal{R}'$ of clusters has at least $\mu = \lceil \frac{h_0}{2\Delta} \rceil$ node-disjoint paths connecting them in $G$. Therefore, $\mu_H(\mathring{T}) \geq \mu$. We apply Theorem 5.4 to graph $H$, set $\mathring{T}$ of terminals and the value $\mu$. Let $H'$ denote the resulting graph, and $\mathcal{U}$ the resulting partition of the edges of $H'$, where each group $U \in \mathcal{U}$ contains at most $r_0$ edges. Recall that each edge $e = (v_i, v_j)$ in $H'$ corresponds to a path $P_e$ connecting $v_i$ to $v_j$ in $H$, where $P_e$ does not contain any vertex of $\mathring{T}$, except for its endpoints. In turn, path $P_e$ defines a path $P'_e$ in graph $G$, connecting a vertex of $S_j$ to a vertex of $S_i$ directly, that is, $P'_e$ does not contain the vertices of $\bigcup_{S \in \mathcal{R}} S$ as inner vertices.

Let $h_1 = \lceil \frac{h_0}{2\Delta r_0} \rceil$. We define a new graph $Z$, whose vertex set is $\{v_1, \ldots, v_r\}$. We add an edge $(v_i, v_j)$ to $Z$ iff there are at least $h_1$ parallel edges connecting $v_i$ to $v_j$ in $H'$. It is easy to verify that graph $Z$ is connected: indeed, assume otherwise. Let $A$ be any connected component of $Z$, and let $B$ contain the rest of the vertices. Let $v_j \in A$, $v_j' \in B$. Since there are at least $2\mu \geq \lceil \frac{h_0}{2\Delta} \rceil$ edge-disjoint paths connecting $v_j$ and $v_j'$ in $H'$, $|E_{H'}(A, B)| \geq \lceil \frac{h_0}{2\Delta} \rceil$ must hold. Since $|A| + |B| = r_0$, at least one pair $(v_j, v_j')$ with $v_j \in A$, $v_j' \in B$, has at least $h_1$ parallel edges connecting them in $H'$.

Let $T$ be any spanning tree of $Z$ that is rooted at some arbitrary node. We say that Case 1 happens if $T$ contains a root-to-leaf path of length at least $r$, and we say that Case 2 happens otherwise. Since $|V(T)| = r_0 = r^2$, and every vertex of $T$ lies on some root-to-leaf path, if Case 2 happens, then $T$ contains at least $r$ leaves (see Claim 2.2). We now consider each of the two cases separately. For Case 1, we build a good tree-of-sets system directly, and we only apply Phase 2 of the algorithm if Case 2 happens.

**Case 1** Let $P$ be any path of $T$ of length exactly $r$. Assume without loss of generality that $P = \{v_1, \ldots, v_r\}$. Let $\mathcal{R}' = \{S_1, \ldots, S_r\}$ be the set of corresponding clusters. We build a tree-of-sets system $(\mathcal{R}', T^*, \bigcup_{e \in T^*} P'_e)$. The tree $T^*$ is just a path connecting $v_1, \ldots, v_r$ in this order. In order to define the sets $P'_e$ of edges, assume otherwise. Let $A$ be any connected component of $Z$, and let $B$ contain the rest of the vertices. Let $v_j \in A$, $v_j' \in B$. Since there are at least $2\mu \geq \lceil \frac{h_0}{2\Delta} \rceil$ edge-disjoint paths connecting $v_j$ and $v_j'$ in $H'$, $|E_{H'}(A, B)| \geq \lceil \frac{h_0}{2\Delta} \rceil$ must hold. Since $|A| + |B| = r_0$, at least one pair $(v_j, v_j')$ with $v_j \in A$, $v_j' \in B$, has at least $h_1$ parallel edges connecting them in $H'$.

In order to solve the first problem, we will define, for each $1 \leq i < r$, a subset $\mathcal{P}'_i \subset \mathcal{P}_1$ of $h$ paths, such that every vertex of $\bigcup_{i=1}^{r-1} S_i$ belongs to at most one path in $\bigcup_{i=1}^{r-1} \mathcal{P}'_i$ (or in other words, the paths in $\bigcup_{i=1}^{r-1} \mathcal{P}'_i$ have distinct endpoints).

We start with the set $\mathcal{P}_1$. Notice that every vertex of $S_1$ may serve as an endpoint of at most $\Delta$ paths in $\mathcal{P}_1$, since these paths cannot share edges. Therefore, we can select a subset $\mathcal{P}_1'' \subset \mathcal{P}_1$ of $\lfloor \frac{h_0}{2\Delta} \rfloor$ paths.
that do not share endpoints in \( S_1 \). Similarly, we can select a subset \( \mathcal{P}'_1 \subseteq \mathcal{P}''_1 \) of \( \lfloor \frac{h_2}{2^3} \rfloor \) paths that do not share endpoints in \( S_2 \). We then consider the set \( \mathcal{P}_2 \) of paths. We delete from \( \mathcal{P}_2 \) all paths that share endpoints with paths in \( \mathcal{P}'_1 \). Since \( |\mathcal{P}'_1| = \lfloor \frac{h_2}{2^3} \rfloor \), at most \( \frac{h_2}{2^3} \) paths can share endpoints with paths in \( \mathcal{P}'_1 \), so at least half the paths in \( \mathcal{P}_2 \) remain. As before, we select a subset \( \mathcal{P}''_2 \) of \( \lfloor \frac{h_2}{2^3} \rfloor \) such paths that do not share vertices in \( S_2 \), and a subset \( \mathcal{P}'_2 \) of \( \lfloor \frac{h_2}{2^3} \rfloor \) paths that do not share vertices in \( S_3 \), obtaining a set \( \mathcal{P}'_3 \) of paths that are completely disjoint. We continue this process for all \( 1 \leq i < r \), until all paths in set \( \mathcal{P}' = \bigcup_i \mathcal{P}'_i \) are mutually disjoint. The sets \( \mathcal{P}'_i \) of paths are then used to define the sets \( \mathcal{P}'_e \) of paths in the tree-of-sets system. Notice that the size of each set is \( \lfloor \frac{h_2}{2^3} \rfloor \geq \frac{h_1}{8r_0 \Delta^2} \geq \frac{h_0}{2^2 r_0} > h \) from Equation (1).

Let \( G^* \) be the sub-graph of \( G \) obtained by the union of \( G[S] \) for \( S \in \mathcal{R}' \) and \( \bigcup'_{S \in \mathcal{T}'} \mathcal{P}^*_e \). We need to verify that each set \( S_i \) has the \( \alpha_{gw} \)-bandwidth property in \( G^* \). Let \( \Gamma_i \) be the interface of the set \( S_i \) in \( G^* \). We set up a sparsest cut problem instance with the graph \( G[S_i] \) and the set \( \Gamma_i \) of terminals, and apply algorithm \( \mathcal{A}_{ARV} \) to it. If the outcome is a cut of sparsity less than \( \alpha \), then, since \( |\Gamma_i| < h_0 \), we obtain an \((h_0, \alpha)\)-violating cut of \( S_i \) in graph \( G \). We return this cut as the outcome of the algorithm. If \( \mathcal{A}_{ARV} \) returns a cut of sparsity at least \( \alpha \) for each set \( S_i \), for \( 1 \leq i \leq r \), then we are guaranteed that each such set has the \( \alpha_{gw} \)-bandwidth property in \( G^* \), and we have therefore constructed a good tree-of-sets system. (We are guaranteed that \( S_i \cap \mathcal{T} = \emptyset \) for each \( i \), since each set \( S_i \in \mathcal{R} \) only contains non-terminal vertices).

**Case 2** If Case 2 happens, then we will need to execute the second phase of the algorithm, but first we need to establish the following useful fact. Recall that we have found a tree \( T \) in graph \( Z \) containing at least \( r \) leaves. Let \( \mathcal{R}' \subseteq \mathcal{R} \) be any subset of \( r \) clusters, corresponding to the leaves of \( T \). For simplicity of notation, we assume that \( \mathcal{R}' = \{S_1, \ldots, S_r\} \). We say that a path \( P \) connects \( S_i \) to \( S_j \) directly, for \( S_i, S_j \in \mathcal{R}' \) iff no inner vertex of \( P \) belongs to \( \bigcup_{S \in \mathcal{R}'} S \) (but they may belong to clusters \( S \in \mathcal{R} \setminus \mathcal{R}' \)).

**Theorem 5.7** There is an efficient randomized algorithm, that with high probability either computes an \((h_0, \alpha)\)-violating cut of some set \( S \in \mathcal{R} \), or finds, for each set \( S_i \in \mathcal{R}' \), a subset \( E_i \subseteq \text{out}_{G}(S_i) \) of edges, and for each \( 1 \leq i \neq j \leq r \) a collection \( \mathcal{P}_{i,j} \) of paths that satisfy the following properties:

1. \(|E_i| = h_3 = \Omega \left( \frac{\alpha^2 h_1 r_0}{r \Delta^2 \log k} \right)\)
2. \( \mathcal{P}_{i,j} \) is a collection of \( h_3 \) node-disjoint paths that directly connect \( S_i \) to \( S_j \)
3. each path \( P \in \mathcal{P}_{i,j} \) contains an edge of \( E_i \) and an edge of \( E_j \) as its first and last edges, respectively.

Notice that the theorem implies that the edges in each set \( E_i \) do not share endpoints.

**Proof:** Recall that each edge \( e \in E(H') \) corresponds to a path \( P_e \) in \( H \). Since for any choice of edges \( e_U \in U \) for each \( U \in \mathcal{U} \), we are guaranteed that the corresponding set \( \{P_e \mid U \in \mathcal{U}\} \) of paths is edge-disjoint in \( H \), and \(|U| \leq r_0 \) for all \( U \in \mathcal{U} \), the total edge congestion in \( H \) due to the paths in \( \{P_e \mid e \in E(H')\} \) is at most \( r_0 \). Each such path \( P_e \) naturally defines a direct path \( P'_e \) in \( G \), and the total edge congestion due to the paths in \( \{P'_e \mid e \in E(H')\} \) is at most \( r_0 \) in \( G \).

Let \( v^* \) be the root of the tree \( T \), and let \( S^* \in \mathcal{R} \) be the corresponding vertex subset. We claim that for each set \( S_j \in \mathcal{R}' \), there is a flow \( F_j \) of value \( h_1 \), connecting the vertices of \( S^* \) to the vertices of \( S_j \) in \( G \), with edge-congestion at most \( \frac{4}{n} + r_0 \leq \frac{3}{n} \), such that the flow-paths in \( F_j \) do not contain any vertices of \( \bigcup_{S \in \mathcal{R}'} S \) as inner vertices. Indeed, consider the path \( (v^* = v_{i_1}, v_{i_2}, \ldots, v_{i_k} = v_j) \) in the tree \( T \), connecting \( v_{i^*} \) to \( v_j \). For each edge \( e_z = (v_{i_z}, v_{i_{z+1}}) \) on this path, there are \( h_1 \) parallel edges.
corresponding to \( e_z \) in graph \( H' \). Let \( Q_z \) be the corresponding set of \( h_1 \) direct paths connecting the vertices of \( S_{i_z} \) to the vertices of \( S_{i_{z+1}} \) in \( G \). Denote

\[
\Gamma_z^2 = \{ v \in S_{i_z} \mid v \text{ is the first vertex on some path in } Q_z \},
\]

and similarly

\[
\Gamma_{z+1}^1 = \{ v \in S_{i_{z+1}} \mid v \text{ is the last vertex on some path in } Q_z \}.
\]

For each \( 1 \leq z < x \), we have now defined two subsets \( \Gamma_z^1, \Gamma_z^2 \subseteq S_{i_z} \) of vertices of size \( h_1 < h_0 \) each. We now try to send \( h_1 \) flow units from \( \Gamma_z^1 \setminus \Gamma_z^2 \) to \( \Gamma_z^2 \setminus \Gamma_z^1 \) inside \( G[S_{i_z}] \) with edge-congestion at most \( 1/\alpha \). If such a flow does not exist, then the minimum edge-cut separating these two vertex subsets defines an \((h_0, \alpha)\)-violating cut of \( S_{i_z} \). We then terminate the algorithm and return this cut. We assume therefore that such a flow exists, and denote it by \( F'_z \).

Concatenating the flows \((Q_1, F'_1, Q_2, F'_2, \ldots, F'_{x-1}, Q_x)\), we obtain the desired flow \( F_j \) of value \( h_1 \). The total congestion caused by paths in \( \bigcup_{z=1}^r Q_z \) is at most \( r_0 \), while each flow \( F'_z \) causes congestion at most \( 1/\alpha \) inside the graph \( G[S_{i_z}] \). Therefore, the total congestion due to flow \( F_j \) is bounded by \( \frac{1}{\alpha} + r_0 \leq \frac{2}{\alpha} \).

Scaling all flows \( F_j \), for \( S_j \in R' \) down by factor \( 2r\Delta/\alpha \), we obtain a new flow \( F \), where every set \( S_j \in R' \) sends \( \frac{h_0}{2r\Delta} \) flow units to \( S^* \), and the total vertex-congestion due to \( F \) is at most 1. The flow-paths of \( F \) do not contain the vertices of \( \bigcup_{z=1}^r S_j \) as inner vertices. From the integrality of flow, there is a collection \( \{P_j\}_{j=1}^r \) of path sets, where for each \( 1 \leq j \leq r \), set \( P_j \) contains \( \lceil \frac{h_0}{2r\Delta} \rceil \) paths connecting \( S_j \) to \( S^* \), the paths in \( \bigcup_{j=1}^r P_j \) are node-disjoint, and they do not contain the vertices of \( \bigcup_{j=1}^r S_j \) as inner vertices (the latter property is achieved by setting the capacity of every edge with both endpoints in the same cut set \( S_j \in R' \) to 0). We will also assume without loss of generality that the paths in \( \bigcup_{j=1}^r P_j \) do not contain the vertices of \( S^* \) as inner vertices.

For each \( 1 \leq j \leq r \), let \( A_j \subseteq S^* \) be the set of vertices that serve as endpoints of paths in \( P_j \), and let \( A = \bigcup_{j=1}^r A_j \). Notice that \( |A| = r \cdot \lceil \frac{h_0}{2r\Delta} \rceil < h_0 \). We set up an instance of the sparsest cut problem in graph \( G[S^*] \), where the vertices of \( A \) act as terminals, and apply algorithm \( \mathcal{A}_{\text{ARV}} \) to this problem. If the algorithm returns a cut whose sparsity is less than \( \alpha \), then we have found a \((h_0, \alpha)\)-violating cut in \( S^* \in \mathcal{R} \). We return this cut, and terminate the algorithm. Otherwise, we are guaranteed that the set \( A \) is \( \alpha_{\text{GW}} \)-well-linked in \( G[S^*] \).

We apply Corollary 2.2 to graph \( G[S^*] \) and the sets \( A_1, \ldots, A_r \) of vertices, obtaining, for each \( 1 \leq j \leq r \), a subset \( A_j^* \subseteq A_j \) of \( \Omega \left( \frac{\alpha_{\text{GW}} |A_j|}{2r\Delta \log k} \right) = \Omega \left( \frac{\alpha^2 h_1}{2r\Delta \log k} \right) = h_3 \) vertices, such that for all \( 1 \leq i \neq j \leq r \), \( A_j^* \) and \( A_i^* \) are linked in \( G[S^*] \). Let \( Q_{i;j} \) be the set of \( h_3 \) node-disjoint paths connecting \( A_j^* \) to \( A_i^* \) in \( G[S^*] \).

For each \( 1 \leq j \leq r \), let \( P_j' \subseteq P_j \) be the subset of paths whose endpoint belongs to \( A_j^* \), and let \( E_j \subseteq \text{out}(S_j) \) be the set of edges \( e \), where \( e \) is the first edge on some path of \( P_j' \), so \( |E_j| = h_3 \). Consider any pair \( 1 \leq i < j \leq r \) of indices. The desired set of paths connecting \( S_i \) to \( S_j \) is obtained by concatenating the paths in \( P_i', Q_{i;j} \), and \( P_j' \).

To summarize, we have found a collection \( R' = \{ S_1, \ldots, S_r \} \) of \( r \) disjoint vertex subsets, and for each \( S_j \), a collection \( E_j \subseteq \text{out}(S_j) \) of \( h_3 \) edges, such that for each \( 1 \leq j \neq i \leq r \), there is a set of \( h_3 \) node-disjoint paths in \( G \), connecting \( S_i \) to \( S_j \) directly, such that each path contains an edge of \( E_j \) and an edge of \( E_i \) as its first and last edges, respectively.
5.4.2 Phase 2

We construct a new graph $\tilde{H}$, obtained from $G$ as follows. First, for each $1 \leq j \leq r$, we delete all edges in $\text{out}(S_j) \setminus E_j$. Let $B_j \subset S_j$ be the subset of vertices containing the endpoints of the edges in $E_j$ that belong to $S_j$. We delete all vertices of $S_j \setminus B_j$ and add a new super-node $v_j$ that connects to every vertex in $B_j$ with an edge. Recall that from the above discussion, $|B_j| = h_3$, so the degree of every super-node $v_j$ in $\tilde{H}$ is $h_3$, and every pair $v_j, v_i$ of super-nodes are connected by $h_3$ paths, that are completely disjoint, except for sharing the first and the last vertex. We will think of the super-node $v_j$ as representing the set $S_j \in \mathcal{R}'$. In this phase, the vertices of $\{v_1, \ldots, v_r\}$ are called terminals, and a path $P$ connecting a vertex of $S_i$ to a vertex of $S_j$ in $G$ is called direct if it does not contain the vertices of $\bigcup_{j=1}^{r} S_j$ as inner vertices.

We apply Theorem 5.4 to graph $\tilde{H}$, with the set $\{v_1, \ldots, v_r\}$ of terminals and $\mu = h_3$. Let $\tilde{H}'$ denote the resulting graph, and $U$ the resulting partition of the edges of $\tilde{H}$ into groups of size at most $r$. Recall that the degree of every vertex of $\tilde{H}'$ is at most $2h_3$, and every pair of vertices is $(2h_3)$-edge-connected. This can only happen if the degree of every vertex is exactly $2h_3$. The main difference between graph $\tilde{H}'$ and the graph $H'$ that we computed in Phase 1 is that now the degree of every terminal, and the edge-connectivity of every pair of terminals, are the same. It is this property that allows us to build the tree-of-sets system. To simplify notation, denote $\ell = 2h_3$.

Suppose we choose, for each group $U \in \mathcal{U}$, some edge $e_U \in U$. Then Theorem 5.4 guarantees that all paths in $\{P_{e_U} \mid U \in \mathcal{U}\}$ are node-disjoint in $\tilde{H}$, except for possibly sharing endpoints, and they do not contain terminals as inner vertices. For each such edge $e_U = (v_j, v_j') \in E(\tilde{H}')$, path $P_{e_U}$ in $\tilde{H}$ naturally defines a direct path $P'_{e_U}$, connecting a vertex of $S_i$ to a vertex of $S_j$ in $G$. Moreover, from the definition of graph $\tilde{H}$, the paths in $\{P'_{e_U} \mid U \in \mathcal{U}\}$ are completely node-disjoint in $G$. For each edge $e \in E(\tilde{H}')$, let $P'_e$ denote the path in graph $G$ corresponding to the path $P_e$ in $\tilde{H}$.

We build an auxiliary undirected graph $\tilde{Z}$ on the set $\{v_1, \ldots, v_r\}$ of vertices, as follows. For each pair $v_j, v_{j'}$ of vertices, there is an edge $(v_j, v_{j'})$ in graph $\tilde{Z}$ iff there are at least $\ell/r^3$ edges connecting $v_j$ and $v_{j'}$ in $\tilde{H}'$. If edge $e = (v_j, v_{j'})$ is present in graph $\tilde{Z}$, then its capacity $c(e)$ is set to be the number of edges connecting $v_j$ to $v_{j'}$ in $\tilde{H}'$. For each vertex $v_j$, let $C(v_j)$ denote the total capacity of edges incident on $v_j$ in graph $\tilde{Z}$. We need the following simple observation.

**Observation 5.2**

- For each vertex $v \in V(\tilde{Z})$, $(1 - 1/r^2)\ell \leq C(v) \leq \ell$.
- For each pair $(u, v)$ of vertices in graph $\tilde{Z}$, we can send at least $(1 - 1/r)\ell$ flow units from $u$ to $v$ in $\tilde{Z}$ without violating the edge capacities.

**Proof:** In order to prove the first assertion, recall that each vertex in graph $\tilde{H}'$ has $\ell$ edges incident to it. So $C(v) \leq \ell$ for all $v \in V(\tilde{Z})$. Call a pair $(v_j, v_{j'})$ of vertices bad iff there are fewer than $\ell/r^3$ edges connecting $v_j$ to $v_{j'}$ in $\tilde{H}'$. Notice that each vertex $v \in V(\tilde{Z})$ may participate in at most $r$ bad pairs, as $|V(\tilde{Z})| = r$. Therefore, $C(v) \geq \ell - r\ell/r^3 = \ell(1 - 1/r^2)$ must hold.

For the second assertion, assume for contradiction that it is not true, and let $(u, v)$ be a violating pair of vertices. Then there is a cut $(A, B)$ in $\tilde{Z}$, with $u \in A$, $v \in B$, and the total capacity of edges crossing this cut is at most $(1 - 1/r)\ell$. Since $u$ and $v$ were connected by $\ell$ edge-disjoint paths in graph $\tilde{H}'$, this means that there are at least $\ell/r$ edges in graph $\tilde{H}'$ that connect bad pairs of vertices. But since we can only have at most $r^2$ bad pairs, and each pair has fewer than $\ell/r^3$ edges connecting them, this is impossible. \hfill \qed
The following claim allows us to find a spanning tree of $\tilde{Z}$ with maximum vertex degree at most 3. This tree will be used to define the tree-of-sets system.

**Claim 5.6** There is an efficient algorithm to find a spanning tree $T^*$ of $\tilde{Z}$ with maximum vertex degree at most 3.

**Proof:** We use the algorithm of Singh and Lau [SL07] for constructing bounded-degree spanning trees. Suppose we are given a graph $G = (V, E)$. Notice that it suffices to establish Constraint (3) for subsets $S$ with $|S| \geq 2$. From Observation 5.2, the total capacity of edges in $E_\tilde{Z}(S, \bar{S})$ must be at least $(1 - 1/r)\ell$. Since for each $v \in S$, $C(v) \leq \ell$, the total contribution of the vertices in $S$ towards the LP-weights of edges in $E_\tilde{Z}(S, \bar{S})$ is at least $\frac{r-1}{r} \cdot (1 - 1/r)^2 = (1 - 1/r)^2$. Therefore,

$$\sum_{e \in E(S)} x_e \leq \frac{r-1}{r} |S| - (1 - 1/r)^2 = |S| - |S|/r - 1/r^2 - 2/r \leq |S| - 1$$

since we assume that $|S| \geq 2$. This establishes Constraint (3). Finally, we show that for each $v \in V(\tilde{Z})$, $\sum_{e \in \delta(v)} x_e \leq 1$. Next, recall that for each $u \in V(\tilde{Z})$, $C(u) \geq (1 - 1/r^2)\ell$, while the total capacity of edges in $\delta(v)$ is at most $\ell$. Therefore, the total contribution of other vertices to this summation is bounded by $\frac{r}{(1-1/r^2)\ell} \cdot \frac{r-1}{r} \leq \frac{r}{r+1} \leq 1$. The algorithm of Singh and Lau can now be used to obtain a spanning tree $T^*$ for $\tilde{Z}$ with maximum vertex degree at most 3.

We are now ready to define the tree-of-sets system $(R', T^*, \bigcup_{e \in E(T^*)} P^*(e))$. The tree $T^*$ is the tree computed by Claim 5.6. In order to define the sets $P^*(e)$ of paths, recall that each edge $e$ of $\tilde{Z}$...
Theorem 6.1 Suppose we are given a graph $G$ of at least $2h^3/r^3$ edges of $\tilde{H}$. For each group $U \in \mathcal{U}$, we randomly choose one edge $e_U \in \mathcal{U}$, and we let $E^* \subseteq E(\tilde{H})$ be the set of all selected edges. For each $e \in E(T^*)$, let $S_e = S_e \cap E^*$. The expected size of $S_e$ is at least $\frac{2h}{r^2}$, and using the standard Chernoff bound, with high probability, for each edge $e \in E(T^*)$, $|S_e| \geq \frac{h}{2r^2}$, since $h^3/r^4 \geq h \geq 4 \log k$. This is since $\frac{h^3}{2r^3} = \Omega \left( \frac{h^4}{2r^4} \right) = \Omega \left( \frac{h^2}{2r^2} \right) \geq h$ from Equation (1). The final set $P_e^*$ of paths is $\{P_e' | e' \in S_e^*\}$. Notice that $|P_e^*| \geq h^3/r^4 \geq h$. We delete paths from $P_e$ as necessary, until $|P_e^*| = h$.

From the definition of the graph $\tilde{H}$, and from Theorem 5.4, all paths in $\bigcup_{e \in E(T^*)} P_e^*$ are mutually node-disjoint.

Let $G^*$ be the sub-graph of $G$ obtained by taking the union of $G[S_j]$ for $S_j \in \mathcal{R}'$, and $\bigcup_{e \in E(T^*)} P_e^*$. We need to verify that each set $S_i$ has the $\alpha_{bw}$-bandwidth property in $G^*$. Let $\Gamma_i$ be the interface of the set $S_i$ in $G^*$. We set up a sparsest cut problem instance with the graph $G[S_i]$ and the set $\Gamma_i$ of terminals, and apply algorithm $A_{ARV}$ to it. If the outcome is a cut of sparsity less than $\alpha$, then, since $|\Gamma_i| < h_0$, we obtain an $(h_0, \alpha)$-violating cut of $S_i$ in graph $G$. We return this cut as the outcome of the algorithm. If $A_{ARV}$ returns a cut of sparsity at least $\alpha$ for each set $S_i$, for $1 \leq i \leq r$, then we are guaranteed that each such set has the $\alpha_{bw}$-bandwidth property in $G^*$, and we have therefore constructed a good tree-sets system.

6 Extensions

The following theorem gives a slightly stronger version of Theorem 4.1, that we believe may be useful in designing approximation algorithms for routing problems. The proof follows easily from the proof of Theorem 4.1.

Theorem 6.1 Suppose we are given a graph $G$ of maximum vertex degree $\Delta$, 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 any parameters $r > 1, h' > 4 \log k$, such that $k/\log^4 k > c'h'^{19}\Delta^8$, where $c'$ is a large enough constant. Assume that we are also given some subset $T' \subset T$ of $h'$ terminals. Then there is an efficient randomized algorithm that with high probability computes a subgraph $G^*$ of $G$, and a tree-of-sets system $(S, T, \bigcup_{e \in E(T')} P_e)$ in $G^*$, with parameters $[h'], [r]$ and $\alpha_{bw} = \Omega \left( \frac{1}{r^2 \log^{1.5} k} \right)$. Moreover,

- For all $S_i \subseteq S$, $S_i \cap T = \emptyset$, and $S_i$ has the $\alpha_{bw}$-bandwidth property in $G^*$; and
- There is a set $Q$ of $[2h'/3]$ node-disjoint paths connecting the terminals in $T'$ to the vertices of $\bigcup_{S_i \in S} S_i$ in graph $G^*$, such that the paths in $Q$ do not contain any vertices of $V \left( \bigcup_{e \in E(T')} P_e \right)$, and are internally disjoint from $\bigcup_{S_i \in S} S_i$.

(Notice that the definition of the tree-of-sets system only requires that each set $S_i \subseteq S$ has the $\alpha_{bw}$-bandwidth property in the sub-graph $G'$ of $G$ induced by the vertices of the tree-of-set system. The Theorem requires a slightly stronger property, that $S_i$ has the $\alpha_{bw}$-bandwidth property in the graph $G^*$, that contains both the tree-of-set system, and the set $Q$ of paths.)

Proof: Let $h = 3h'$. The proof very closely follows the proof of Theorem 4.1, using the parameters $h, r, \alpha_{bw}$. As before, if $(S, T, \bigcup_{e \in E(T')} P_e)$ is a tree-of-sets system in $G$, with parameters $h, r, \alpha_{bw}$, and for each $S_i \subseteq S$, $S_i \cap T = \emptyset$, then we say that it is a good tree-of-sets system. We define the potential function, acceptable clustering, and good clustering exactly as before, using the parameters $h, r, \alpha_{bw}$.
As before, the algorithm consists of a number of phases, where the input to every phase is a good clustering $C$ of $V(G)$, and the output is either another good clustering $C'$ with $\varphi(C') \leq \varphi(C) - 1$, or a sub-graph $G^*$ of $G$, together with a good tree-of-sets system in $G^*$, for which the conditions of the corollary hold. The initial clustering is defined exactly as before: $\{\{v\} \mid v \in V(G)\}$.

We now proceed to describe each phase. Suppose the input to the current phase is a good clustering $C_j$ together with a sub-graph $T \subseteq G$ for some $1 \leq j \leq r_0$, an acceptability parameter $\alpha$, and a legal contracted graph $h$ of $C_j$. Let $\{S_0, S_1, \ldots, S_r\}$ be the corresponding tree-of-sets system in $G^*$, and compute, for each $1 \leq j \leq r$, an acceptability parameter $\alpha$, and a valid output for Theorem 6.1, that is:

- either an $(h_0, \alpha)$-violating partition $(X, Y)$ of $S_j$, for some $1 \leq j \leq r_0$;
- or a partition $(A, B)$ of $V(G)$ with $S_j \subseteq A$, $T \subseteq B$ and $|E_G(A, B)| < h_0/2$, for some $1 \leq j \leq r_0$;
- or a valid output for Theorem 6.1 that is:
  - a subgraph $G^*$ of $G$;
  - a tree-of-sets system $(S, T, \bigcup_{e \in E(T)} P_e)$ in $G^*$ with parameters $h', r, \alpha_{BW}$, where for each $i \in S$, $S_i \cap T = \emptyset$, and $S_i$ has the $\alpha_{BW}$-bandwidth property in $G^*$; and
  - a set $Q$ of $\lfloor 2h'/3 \rfloor$ node-disjoint paths connecting the terminals in $T'$ to the vertices of $\bigcup_{S_i \in S} S_i$ in $G^*$, such that the paths in $Q$ do not contain any vertices of $V \left( \bigcup_{e \in E(T)} P_e \right)$, and are internally disjoint from $\bigcup_{S_i \in S} S_i$.

Just as in the proof of Theorem 4.1, the proof of Theorem 6.1 follows from the proof of Theorem 6.2. We start with the initial collection $C_1, \ldots, C_{r_0}$ of acceptable clusterings, where for each $1 \leq j \leq r_0$, $\varphi(C_j) \leq \varphi(C) - 1$. If any of these clusterings $C_j$ is a good clustering, then we terminate the phase and return this clustering. Otherwise, each clustering $C_j$ must contain a large cluster $S_j \in C_j$. We then iteratively apply Theorem 6.2 to clusters $\{S_1, \ldots, S_{r_0}\}$. If the outcome is a valid output for Theorem 6.1, then we terminate the algorithm and return this output. Otherwise, we obtain either an $(h_0, \alpha)$-violating partition of some cluster $S_j$, or a partition $(A, B)$ of $V(G)$ with $S_j \subseteq A$, $T \subseteq B$ and $|E_G(A, B)| < h_0/2$, for some $1 \leq j \leq r_0$. We then apply the appropriate action: PARTITION$(S_j, X, Y)$, or SEPARATE$(S_j, A)$ to the clustering $C_j$, and obtain an acceptable clustering $C'_j$, with $\varphi(C'_j) \leq \varphi(C_j) - 1/n$. If $C'_j$ is a good clustering, then we terminate the phase and return $C'_j$. Otherwise, we select any large cluster $S'_{j'}$ in $C'_j$, replace $S_j$ with $S'_{j'}$ and continue to the next iteration. As before, we are guaranteed that after polynomially-many iterations, the algorithm will terminate with the desired output.

From now on we focus on proving Theorem 6.2. Given the input collection $\{S_1, \ldots, S_{r_0}\}$ of vertex subsets, we run the algorithm from Theorem 5.3 on it. If the outcome is an $(h_0, \alpha)$-violating partition $(X, Y)$ of $S_j$, for some $1 \leq j \leq r_0$, or a partition $(A, B)$ of $V(G)$ with $S_j \subseteq A$, $T \subseteq B$ and $|E_G(A, B)| < h_0/2$, for some $1 \leq j \leq r_0$, then we terminate the algorithm and return this partition.

Therefore, we can assume from now on that the algorithm from Theorem 5.3 has computed a good tree-of-sets system $(S, T, \bigcup_{e \in E(T)} P_e)$ in $G$, where $S = \{S_1, \ldots, S_{r_0}\}$. Let $U = \bigcup_{S_i \in S} S_i$. Recall that the algorithm also ensures that each set $S_j$ can send $h_0/2$ flow units to the terminals with no edge-congestion. Scaling the flow for $S_1$ down by factor $\Delta$ and using the integrality of flow, we conclude that there is a set $T_1$ of $2h' < h < h_0/2\Delta$ terminals, and a set $Q_1$ of $2h'$ node-disjoint paths connecting...
the terminals in $T_1$ to the vertices of $U$. Partition the set $T_1$ into two equal-sized subsets, $T'_1$ and $T'_2$, each containing $h'$ terminals. Since the set $T$ of terminals is node-well-linked, there is a set $Q_2$ of $h'$ node-disjoint paths from the terminals of $T'$ to the terminals of $T'_1$, and a set $Q'_2$ of $h'$ node-disjoint paths from the terminals of $T'$ to the terminals of $T'_2$. Taking the union of $Q_2$ and $Q'_2$, and concatenating them with the paths in $Q_1$, we obtain a collection $2h'$ paths, connecting the terminals in $T'$ to the vertices of $U$ with total vertex-congestion of at most 3. By sending $1/3$ flow unit along each such path, we obtain a flow of value $2h'/3$ from $T'$ to $U$ with no vertex-congestion. From the integrality of flow, there is a collection $\hat{Q}$ of $\lfloor 2h'/3 \rfloor$ node-disjoint paths from $T'$ to $U$. We assume w.l.o.g. that the paths in $\hat{Q}$ do not contain the vertices of $U$ as inner vertices, by suitably truncating them if necessary.

Let $G^*$ be the graph obtained from the union of $G[S_j]$ for all $1 \leq j \leq r_0$, the paths in $\hat{Q}$, and $\bigcup_{e \in E(T)} P_e$, so the good tree-of-sets system that we have computed is contained in $G^*$. For each $1 \leq j \leq r_0$, $|\text{out}_{G^*}(S_j)| \leq h + h' < h_0$. Therefore, we can check whether each set $S_i \subseteq S$ has the $\alpha_{bw}$-bandwidth property in $G^*$ as before: Let $\Gamma_i$ be the interface of the set $S_i$ in $G^*$. We set up a sparsest cut problem instance on the graph $G[S_i]$ and the set $\Gamma_i$ of terminals, and apply algorithm $A_{ARV}$ to it. If the outcome is a cut of sparsity less than $\alpha$, then, since $|\Gamma_i| < h_0$, we obtain an $(h_0, \alpha)$-violating partition of $S_i$ in graph $G$. We return this cut as the outcome of the algorithm. If $A_{ARV}$ returns a cut of sparsity at least $\alpha$ for each set $S_i$, for $1 \leq i \leq r$, then we are guaranteed that each such set has the $\alpha_{bw}$-bandwidth property in $G^*$.

The only remaining problem is that the paths in $\hat{Q}$ may not be disjoint from the paths in $\mathcal{P}^* = \bigcup_{e \in E(T)} P_e$. In order to overcome this difficulty, we re-route the paths in $\hat{Q}$, and discard up to $2h'$ paths from $\mathcal{P}^*$. This will give us a tree-of-sets system with slightly weaker parameters, where $h$ is replaced by $h'$, but now the paths in $\hat{Q}$ will be disjoint from the paths in sets $P_e$ of the new tree-of-sets system.

If any vertex $v \in U$ serves as an endpoint of some path in $\hat{Q}$, and some path $P \in \mathcal{P}^*$, then we discard $P$ from $\mathcal{P}^*$, and delete the edges of $P$ that do not belong to paths in $\hat{Q}$ from the graph $G^*$. Notice that we discard at most $h'$ paths from $\mathcal{P}^*$ in this step. For every set $S_j \subseteq S$, for every vertex $v \in S_j$, $v$ may now be incident on at most one edge in $\text{out}_{G^*}(S_j)$.

If some path $P \in \mathcal{P}^*$ contains only one edge, then $P$ is disjoint from every path in $\hat{Q}$. We ignore all such paths $P$. Let $\mathcal{P}' \subseteq \mathcal{P}^*$ be the set of all remaining paths, that have not been deleted so far, and which contain more than one edge. For each path $P \in \mathcal{P}'$, let $v$ be any inner vertex of $P$. We split $P$ into two sub-paths at the vertex $v$, $P_1, P_2$, both of which are directed away from $v$. We let $X$ be the resulting set of paths, after we unify all vertices of $U$ into a destination vertex $s$. We let $\mathcal{Y}$ be the set $\hat{Q}$ of paths, which are all directed towards $s$. We now use Lemma 2.3 to find a subset $\mathcal{X}' \subseteq X$ of $|X| - h'$ paths, and for each $Q \in \mathcal{Y}$, a path $\hat{Q}$ with the same endpoints as $Q$, such that, if we denote $\mathcal{Y}' = \{ \hat{Q} \mid Q \in \mathcal{Y} \}$, then all paths in $\mathcal{X}' \cup \mathcal{Y}'$ are pairwise disjoint, except for sharing the last vertex $s$.

The final set $Q$ of $\lfloor 2h'/3 \rfloor$ node-disjoint paths, connecting the terminals of $T'$ to the vertices of $U$ in $G^*$ is defined by the set $\mathcal{Y}'$ of paths. For each edge $e \in E(T)$, we now define the corresponding subset $P'_e \subseteq P_e$ of $h'$ paths, as follows. Consider some path $P \in P_e$. If $P$ has only one edge, we do nothing. Otherwise, consider the two corresponding paths $P_1, P_2$ that we have constructed from $P$. If $P_1$ or $P_2$ do not belong to $\mathcal{X}'$, then we discard $P$ from $P_e$. Notice that the total number of discarded paths is now at most $2h'$. Therefore, at least $h'$ paths remain in $P_e$. We let $P'_e$ be any subset of $h'$ of the remaining paths.

This completes the construction of the tree-of-sets system $(\mathcal{S}, T, \bigcup_{e \in E(T)} P'_e)$ with parameters $r, h', \alpha_{bw}$, and the set $Q$ of $\lfloor 2h'/3 \rfloor$ paths connecting the terminals of $T'$ to the vertices of $\bigcup_{S_j \subseteq S} S_j$. The paths in $Q$ are now guaranteed to be disjoint from the paths in $\bigcup_{e \in E(T)} P'_e$, and they do not contain the vertices.
of \( \bigcup_{j \in S} S_j \) as inner vertices. From the above discussion, each set \( S_j \in S \) has the \( \alpha_{BW} \)-bandwidth property in graph \( G^* \).

\[ \square \]

**Acknowledgement.** 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**


A Proofs Omitted from Section 2

A.1 Proof of Claim 2.3

Let $C$ be the collection of all connected components of $G \setminus E'$. We start by showing that there is a connected component $C \in C$ containing more than $|T|/2$ terminals. Assume otherwise. Then we can use Claim 2.1 to find a partition $(A, B)$ of $C$, such that $\sum_{C' \in A} |C' \cap T|, \sum_{C' \in B} |C' \cap T| \geq |T|/3$. Let $A = \bigcup_{C' \in A} C'$, and $B = \bigcup_{C' \in B} C'$. Then $|A \cap T|, |B \cap T| \geq |T|/3$. From the well-linkedness of $T$, $|E(A, B)| \geq \alpha |T|/3$ must hold. However, $E(A, B) \subseteq E'$ and $|E'| < \alpha |T|/3$, a contradiction.

Let $C \in C$ be the cluster containing more than $|T|/2$ terminals. We now claim that $|C \cap T| \geq |T| - \frac{m'}{\alpha}$. Assume otherwise, and let $Z = V(G) \setminus C$. Then $|Z \cap T'| > \frac{m'}{\alpha}$, and from the well-linkedness of $T$, $|E(C, Z)| \geq \alpha |Z \cap T| > m'$. But $E(C, Z) \subseteq E'$ and $|E'| = m'$, a contradiction.

A.2 Proof of Theorem 2.2

We start with a non-constructive proof, since it is much simpler and gives better parameters. This proof can be turned into an algorithm with running time $\text{poly}(n) \cdot 2^c$. We then show a constructive proof with running time $\text{poly}(n, \kappa)$.

A.2.1 A non-constructive proof

A separation in graph $G$ is two subgraphs $Y, Z$ of $G$, such that every edge of $G$ belongs to exactly one of $Y, Z$. The order of the separation is $|V(Y) \cap V(Z)|$. We say that a separation $(Y, Z)$ is balanced if $|V(Y) \cap T|, |V(Z) \cap T| \geq |T|/4$. Let $(Y, Z)$ be a balanced separation of $G$ of minimum order, and let $X = V(Y) \cap V(Z)$. Assume without loss of generality that $|V(Y) \cap T| \geq |V(Z) \cap T|$, so $|V(Y) \cap T| \geq |T|/2$. We claim that $X$ is node-well-linked in graph $Y$.

Claim A.1 Set $X$ of vertices is node-well-linked in graph $Y$.

Proof: Let $A, B$ be any two equal-sized subsets of $X$, and assume that $|A| = |B| = z$. It is enough to show that there is a set $P$ of $z$ disjoint paths connecting $A$ to $B$ in $Y$. Assume otherwise. Then there is a set $S$ of at most $z - 1$ vertices separating $A$ from $B$ in $Y$. 


Let $C$ be the set of all connected components of $Y \setminus S$. We partition $C$ into three subsets: $C_1$ contains all clusters containing the vertices of $A$; $C_2$ contains all clusters containing the vertices of $B$, and $C_3$ contains all remaining clusters (notice that all three sets of clusters are pairwise disjoint). Let $R_1 = \bigcup_{C \in C_1} C$, and define $R_2$ and $R_3$ for $C_2$ and $C_3$, respectively. Assume without loss of generality that $|R_1 \cap T| \geq |R_2 \cap T|$. We define a new separation $(Y', Z')$, as follows. The set of vertices $V(Y') = R_1 \cup R_3 \cup S$, and $V(Z') = V(Z) \cup R_2 \cup S$. Let $X' = V(Y') \cap V(Z')$. The edges of $Y'$ include all edges of $G$ with both endpoints in $V(Y') \setminus X'$, and all edges of $G$ with one endpoint in $V(Y') \setminus X'$ and the other endpoint in $X'$. The edges of $Z'$ include all edges with both endpoints in $Z'$.

We claim that $(Y', Z')$ is a balanced separation. Clearly, $|V(Z') \cap T| \geq |T|/4$, since $V(Z) \subseteq V(Z')$, and $|V(Z) \cap T| \geq |T|/4$. We next claim that $|V(Y') \cap T| \geq |T|/4$. Assume otherwise. Then, from our assumption, $|R_2 \cap T| < |T|/4$, and so $|V(Y) \cap T| < |T|/2$, a contradiction. Therefore, $(Y', Z')$ is a balanced separator. Finally, we claim that its order is less than $|X|$, contradicting the minimality of $X$. Indeed, $|V(Y') \cap V(Z')| \leq |X| - |B| + |S| < |X|$.

Let $T_1 = T \cap V(Z)$ and $T_2 = T \cap V(Y)$. Recall that we have assumed that $|T_1| \leq |T_2|$. Add a source vertex $s$ and connect it to every terminal in $T_1$ with a directed edge, and add a sink vertex $t$ and connect every vertex in $T_2$ to it with a directed edge. Set all edge and vertex capacities to 1, except for $s$ and $t$ whose capacities are infinite. Since $T$ is $\alpha$-well-linked in $G$, there is an $s$-$t$ flow of value at least $|T_1| \geq \kappa/4$ in this network, with edge congestion at most $1/\alpha$. Scaling this flow down by factor $\Delta/\alpha$, we obtain an $s$-$t$ flow of value at least $\frac{\kappa \alpha}{4 \Delta}$ and vertex congestion at most 1. From the integrality of flow, there is a set of $\left\lceil \frac{\kappa \alpha}{4 \Delta} \right\rceil$ internally disjoint $s$-$t$ paths. This gives a collection $P'$ of $\kappa' = \left\lceil \frac{\kappa \alpha}{4 \Delta} \right\rceil$ disjoint paths connecting terminals in $T_1$ to terminals in $T_2$. Each such path has to go through a vertex of $X$. For each path $P' \in P'$, we truncate the path $P'$ to the first vertex of $X$ on $P'$ (where the path is directed from $T_1$ to $T_2$). Let $P$ be the resulting set of truncated paths. Then $P$ is a set of $\kappa'$ disjoint paths, connecting the vertices of $T_1$ to the vertices of $X$; every path in $P$ is completely contained in graph $Z$, and is disjoint from $X$ except for its last endpoint that belongs to $X$.

Let $T' \subseteq T_1$ be the set of terminals from which the paths in $P$ originate, and let $X' \subseteq X$ be the set of vertices where they terminate. We claim that $T'$ is node-well-linked in $G$. Indeed, let $A, B \subseteq T'$ be any pair of equal-sized subsets of terminals. Let $U = A \cap B$, $A' = A \setminus U$ and $B' = B \setminus U$.

We define the set $\tilde{A}' \subseteq X'$ as follows: for each terminal $t \in A$, let $P_t \in P$ be the path originating at $t$, and let $x_t$ be its other endpoint, that belongs to $X$. We then set $\tilde{A}' = \{x_t \mid t \in A\}$. We define a set $\tilde{B}' \subseteq X$ similarly for $B$. Let $P_A \subseteq P$ be the set of paths originating at the vertices of $A'$, and let $P_B \subseteq P$ be the set of paths originating at the vertices of $B'$, Notice that both sets of paths are contained in $Z$, and are internally disjoint from $X$. The paths in $P_A \cup P_B$ are also mutually disjoint, and they avoid $U$.

Let $U' = U \cap X$, and consider the two subsets $\tilde{A} = \tilde{A}' \cup U'$ and $\tilde{B} = \tilde{B}' \cup U'$ of vertices of $X$. Denote $|\tilde{A}| = |\tilde{B}| = z$. Since $X$ is node-well-linked in $Y$, there is a set $Q$ of $z$ disjoint paths connecting $\tilde{A}$ to $\tilde{B}$ in $Y$. The paths in $Q$ are then completely disjoint from the paths in $P_1, P_2$ (except for sharing endpoints with them). The final set of paths connecting $A$ to $B$ is obtained by concatenating the paths in $P_1, Q, P_2$.

### A.2.2 A Constructive Proof

We start by reducing the problem to a problem where the set of the terminals is $\frac{1}{2}$-well-linked. Let $T$ be any spanning tree of $G$. We build a collection $S$ of disjoint subtrees of $T$, each of which contains at least $\lceil 1/\alpha \rceil$ and at most $\Delta \cdot \lceil 1/\alpha \rceil$ terminals, as follows. Start with $S = \emptyset$. While $T$ is non-empty, proceed as follows. If $T$ contains at most $\Delta \cdot \lceil 1/\alpha \rceil$ terminals, add $T$ to $S$ and finish the algorithm. Otherwise, let $v$ be the lowest vertex of $T$, whose sub-tree contains at least $\lceil 1/\alpha \rceil$ terminals, and let
\( T_v \) be the sub-tree of \( T \) rooted at \( v \). Since the degree of \( v \) is at most \( \Delta \), and \( v \) is not the root of \( T \), tree \( T_v \) contains at most \((\Delta - 1) \cdot \lfloor 1/\alpha \rfloor\) terminals, and so \( T \setminus T_v \) contains at least \lfloor 1/\alpha \rfloor\) terminals. We add \( T_v \) to \( S \), and delete all vertices of \( T_v \) from \( T \). We then continue to the next iteration. It is easy to see that at the end of this procedure, we obtain a collection \( S \) of disjoint trees, each of which contains at least \lfloor 1/\alpha \rfloor\) and at most \( \Delta \cdot \lfloor 1/\alpha \rfloor\) terminals, while every terminal belongs to one of the trees. Let \( \mathcal{T}^* \) be any subset of terminals, containing exactly one terminal from each tree in \( S \). Denote \(|\mathcal{T}^*| = \kappa'\).

Then \( \kappa' \geq \kappa/(\Delta \cdot \lfloor 1/\alpha \rfloor) \), and from Claim 2.4 the set \( \mathcal{T}^* \) of terminals is \( 1/2 \)-well-linked. Notice that for each terminal \( t \in \mathcal{T}^* \), we have defined a tree \( T_t \in S \), and all these trees are disjoint. Our final collection \( \mathcal{T}' \) of terminals will be chosen from among the terminals of \( \mathcal{T}^* \), and the trees \( \{T_t\}_{t \in \mathcal{T}'} \) will remain unchanged. Therefore, it now remains to find a subset \( \mathcal{T}' \subseteq \mathcal{T}^* \) of \( \Omega \left( \frac{\alpha}{\Delta} \beta_{\text{MW}}(\kappa') \cdot \kappa \right) \) terminals, such that \( \mathcal{T}' \) is node-well-linked. To simplify the notation, from now on, we denote \( \mathcal{T}^* \) by \( \mathcal{T} \), and when we use the term “terminals” to refer to the vertices of the new set \( \mathcal{T} \) only. We also denote by \( \alpha' = \frac{1}{2} \) the well-linkedness of the new set \( \mathcal{T} \) of terminals.

For every subset \( C \subseteq V \) of vertices, let \( \mathcal{T}_C = C \cap \mathcal{T} \). We say that a partition \((A, B)\) of \( V \) is balanced (with respect to \( \mathcal{T} \)), iff \(|T_A|, |T_B| \geq \frac{\kappa'}{2\Delta} \).

We need the following lemma that follows from the well-linkedness of \( \mathcal{T} \).

**Lemma A.1** Let \((A, B)\) be any balanced partition of \( V \). There is an efficient algorithm that computes a collection \( \mathcal{P} \) of disjoint paths from \( \mathcal{T}_B \) to \( \mathcal{T}_A \) where \(|\mathcal{P}| \geq \left\lceil \frac{\kappa'\alpha'}{2\Delta^2} \right\rceil \).

**Proof:** We build the following flow network. Add a source vertex \( s \) and connect it to every terminal in \( \mathcal{T}_B \) with a directed edge, and add a sink vertex \( t \) and connect every terminal in \( \mathcal{T}_A \) to it with a directed edge. Set all edge and vertex capacities to 1, except for \( s \) and \( t \) whose capacities are infinite. Since \( \mathcal{T} \) is \( \alpha' \)-well-linked in \( G \), there is an \( s \)-\( t \)-flow \( F \) of value \( \min\{|T_A|, |T_B|\} \geq \frac{\kappa'}{2\Delta} \) in this network, with edge congestion at most \( 1/\alpha' \), so the amount of flow through any vertex (except \( s \) and \( t \)) is at most \( \Delta/\alpha' \). Scaling this flow down by factor \( \Delta/\alpha' \), we obtain an \( s \)-\( t \)-flow of value at least \( \frac{\kappa'\alpha'}{2\Delta^2} \) and vertex congestion at most 1. From the integrality of flow, there is a set of \( \left\lceil \frac{\kappa'\alpha'}{2\Delta^2} \right\rceil \) internally disjoint \( s \)-\( t \) paths. This gives a collection \( \mathcal{P} \) of \( \left\lceil \frac{\kappa'\alpha'}{2\Delta^2} \right\rceil \) disjoint paths connecting terminals in \( \mathcal{T}_B \) to terminals in \( \mathcal{T}_A \).

Given a balanced partition \((A, B)\) and a collection of paths \( \mathcal{P} \) from \( \mathcal{T}_B \) to \( \mathcal{T}_A \), we think of the paths as being directed from \( \mathcal{T}_B \) to \( \mathcal{T}_A \). Each path in \( \mathcal{P} \) has to go through an edge \( e = (v, u) \in E(A, B) \) at least once. For a path \( P \in \mathcal{P} \) that is directed as above the notion of the first vertex of \( P \) in \( A \) is well defined. Given \( \mathcal{P} \), let \( \Gamma_A(\mathcal{P}) \) denote the set of all vertices \( v \in A \), where for some \( P \in \mathcal{P} \), \( v \) is the first vertex of \( P \) that belongs to \( A \).

We now show an algorithm to construct a balanced partition with some useful properties.

**Theorem A.1** There is an efficient algorithm to find a balanced partition \((A, B)\) of \( V \) and a collection \( \mathcal{P} \) of \( \left\lceil \frac{\kappa'\alpha'}{2\Delta^2} \right\rceil \) disjoint paths from \( \mathcal{T}_B \) to \( \mathcal{T}_A \) such that:

- \( G[B] \) is connected, and
- The set \( \Gamma_A(\mathcal{P}) \) is \( 1/\beta_{\text{MW}}(\kappa') \)-well-linked in \( G[A] \).

**Proof:** We say that a balanced partition \((A, B)\) is good iff both \( G[A] \) and \( G[B] \) are connected. We start with some initial good balanced partition \((A, B)\) and apply Lemma A.1 to find a collection of paths \( \mathcal{P} \), and then perform a number of iterations. In every iteration, we will either find a new good balanced partition \((A', B')\) with \(|E(A', B')| < |E(A, B)|\), or we will establish that the current partition
has the required properties (after possibly switching $A$ and $B$). In the former case, we continue to the next iteration, and in the latter case we terminate the algorithm and return the current partition $(A, B)$ and $\mathcal{P}$. Clearly, after at most $|E|$ iterations, our algorithm is guaranteed to terminate with the desired output.

The initial partition $(A, B)$ is computed as follows. Let $T$ be any spanning tree of $G$, rooted at any vertex. Let $v$ be the lowest vertex of $T$ whose subtree contains at least $\frac{\kappa}{2\Delta}$ terminals. Since the degree of every vertex is at most $\Delta$, the subtree of $T$ rooted at $v$ contains at most $\frac{\kappa}{2} + 1$ terminals. We let $A$ contain all vertices in the subtree of $T$ rooted at $v$ (including $v$), and we let $B$ contain all remaining vertices. Then both $A$ and $B$ contain at least $\frac{\kappa}{2\Delta}$ terminals, and both $G[A]$ and $G[B]$ are connected.

Given any good balanced partition $(A, B)$ of $V$, we perform an iteration as follows. Assume without loss of generality that $|T_A| \geq |T_B|$ (otherwise, we switch $A$ and $B$). First, we apply Lemma A.1 to find a collection $\mathcal{P}$ of $\lceil \frac{\kappa \rho^*}{2\Delta} \rceil$ disjoint paths from $T_B$ to $T_A$. Let $S = \Gamma_A(\mathcal{P})$; note that $|S| \leq \kappa/2$. For any subset $Z \subseteq A$ of vertices, we denote $S_Z = Z \cap S$. We set up an instance of the sparsest cut problem in graph $G[A]$ with the set $S$ of terminals. Let $(X, Y)$ be the partition of $A$ returned by the algorithm $\mathcal{A}_{ARV}$ on this instance. If $\frac{|E(X,Y)|}{\min\{|S_X|,|S_Y|\}} \geq 1$, then we are guaranteed that $S$ is $1/\beta_{ARV}(\kappa')$-well-linked in $G[A]$. We then return $(A, B)$ and $\mathcal{P}$ which satisfy the requirements of the theorem. We now assume that $\frac{|E(X,Y)|}{\min\{|S_X|,|S_Y|\}} = \rho < 1$.

Our next step is to show that there is a partition $(X', Y')$ of $A$, such that $G[X']$ and $G[Y']$ are both connected, and the sparsity of the cut $(X', Y')$ (with respect to $S$) is at most $\rho$. In order to show this, we start with the cut $(X, Y)$, and perform a number of iterations. Let $\mathcal{C}$ be the set of all connected components of $G[A] \setminus E(X, Y)$. Each iteration will reduce the number of the connected components in $\mathcal{C}$ by at least 1, while preserving the sparsity of the cut. Let $\mathcal{C}_1 \subseteq \mathcal{C}$ be the set of all connected components contained in $X$, and let $\mathcal{C}_2 \subseteq \mathcal{C}$ be the set of connected components contained in $Y$. Assume without loss of generality that $|S_X| \leq |S_Y|$. If there is some component $C \in \mathcal{C}$ with $|S_C| = 0$, then we can move the vertices of $C$ to the opposite side of the partition $(X, Y)$, and obtain a new partition $(X', Y')$ whose sparsity is less than $\rho$, and the number of connected components in $G[A] \setminus E(X', Y')$ is strictly smaller than $|\mathcal{C}|$. Therefore, we assume from now on that for each $C \in \mathcal{C}$, $|S_C| > 0$.

Assume first that $|\mathcal{C}_1| > 1$. Then $|E(X,Y)| = \rho \cdot |S_X|$, and so there is a connected component $C \in \mathcal{C}_1$ with $|E(C, Y)| \geq \rho \cdot |S_C|$. Moreover, $|S_X| > |S_C|$, since we have assumed that for each $C' \in \mathcal{C}$, $|S_{C'}| > 0$. Consider a new partition $(X', Y')$ of $A$, with $X' = X \setminus C$ and $Y' = Y \cup C$. Notice that the number of the connected components in $G[A] \setminus E(X', Y')$ is strictly smaller than $|\mathcal{C}|$. We claim that the sparsity of the new cut is at most $\rho$. Indeed, the sparsity of the new cut is:

$$\frac{|E(X', Y')|}{|S_X'|} = \frac{|E(X,Y)| - |E(C, Y)|}{|S_X| - |S_C|} \leq \rho|S_X| - \rho|S_C| = \rho.$$

Assume now that $|\mathcal{C}_2| > 1$, and denote $|E(X,Y)|/|S_Y| = \rho'$. Then $\rho' \leq \rho$. As before, there is a connected component $C \in \mathcal{C}_2$ with $|E(C, X)| \geq \rho'|S_C|$ and $|S_C| < |S_Y|$. Consider a new partition $(X', Y')$ of $A$, where $X' = X \cup C$ and $Y' = Y' \setminus C$. As before, the number of connected components in $G[A] \setminus E(X', Y')$ is strictly smaller than $|\mathcal{C}|$. We now show that the sparsity of the new cut is at most $\rho$. If $|S_{Y'}| \leq |S_{X'}|$, then the sparsity of the new cut is:

$$\frac{|E(X', Y')|}{|S_{Y'}|} = \frac{|E(X,Y)| - |E(C, X)|}{|S_Y| - |S_C|} \leq \rho'|S_Y| - \rho'|S_C| = \rho' \leq \rho.$$

Otherwise, $|S_{X'}| < |S_{Y'}|$, and the sparsity of the new cut is:
Each iteration is performed as follows. If \( q \) the vertices \( v \to C \) notation \( r \) \( v \mid \) and finish the algorithm. If \( \nabla \mid u \) There is an efficient algorithm to find a subset \( G \) \( S \) the set of the special neighbors. Recall that from Theorem A.1, \((G \mid P \setminus u) \) of tree \( \{A,B\} \) be the first edge of \( P \). Let \( \beta \) be the set of vertices of \( T \). We continue this procedure, until \( \nabla \mid \) must contain at least \( \nabla \mid P \), \( \nabla \setminus |P| \). Let \( \nabla \mid u \) be the children of \( \nabla \mid \) for some path \( \nabla \mid P \). If \( \nabla \mid u \) of disjoint connected subgraphs of \( G \). We add \( |T| \) \( v \in T \). Denote \( G \mid v \subseteq G[S_v] \). Let \( r \) be the root of \( T \). \( \nabla \mid u \) is always computed with respect to the most current tree \( T \). We start with \( C = \emptyset \), \( \nabla \setminus P \), and then iterate.

Each iteration is performed as follows. If \( q \leq |V(T) \cap U| \leq 4\Delta q \), then we add a new cluster \( G_v \) to \( C \), and finish the algorithm. If \( |V(T) \cap U| < q \), then we also finish the algorithm (we will show later that \( \nabla \setminus P \) must contain at least \( |P| / 2 \) paths at this point). Otherwise, let \( v \setminus \nabla \) be the lowest vertex of \( T \) with \( |T_v \cap U| \geq q \). If \( v \notin U \), then, since the degree of every vertex is at most \( \Delta \), \( |T_v \cap U| \leq \Delta q \). We add \( G_v \) to \( C \), and all paths in \( \{P \mid u_P \in T_v\} \setminus P \). We then delete all vertices of \( T_v \) from \( T \), and continue to the next iteration.

Assume now that \( v = u_P \) for some path \( P \in P \). If \( |T_v \cap U| \leq 4\Delta q \), then we add \( G_v \) to \( C \), and all paths in \( \{P \mid u_P \in T_v\} \setminus P \) and continue to the next iteration. So we assume that \( |T_v \cap U| > 4\Delta q \).

Let \( v_1, \ldots, v_r \) be the children of \( v \) in \( T \). Build a new tree \( T' \) as follows. Start with the path \( P \), and add the vertices \( v_1, \ldots, v_r \) to \( T' \). For each \( 1 \leq i \leq r \), let \((x_i, y_i) \in E(G[B])\) be any edge connecting some

$$\frac{|E(X', Y')|}{|S_X'|} = \frac{|E(X, Y)| - |E(X, C)|}{|S_X| + |S_C|} < \frac{|E(X, Y)|}{|S_X|} = \rho.$$
$x_i \in P$ to some $y_i \in G_{v_i}$. (Such an edge must exist from the definition of $G_{v_i}$ and $T$). Add the edge $(v_i, x_i)$ to $T'$. Therefore, $T'$ is the union of the path $P$, and a number of disjoint stars whose centers lie on the path $P$, and whose leaves are the vertices $v_1, \ldots, v_r$. The degree of every vertex of $P$ is at most $\Delta$. The weight of the vertex $v_i$ is defined to be the number of paths in $P$ contained in $G_{v_i}$. Recall that the weight of each vertex $v_i$ is at most $q$, by the choice of $v$. For each vertex $x \in P$, the weight of $x$ is the total weight of its children in $T'$. Recall that the total weight of the vertices of $P$ is at least $4\Delta q$, and the weight of every vertex is at most $\Delta q$. We partition $P$ into a number of disjoint segments $\Sigma = (\sigma_1, \ldots, \sigma_\ell)$ of weight at least $q$ and at most $2\Delta q$ each, as follows. Start with $\Sigma = \emptyset$, and then iterate. If the total weight of the vertices of $P$ is at most $2\Delta q$, we build a single segment, containing the whole path. Otherwise, find the shortest segment $\sigma$ starting from the first vertex of $P$, whose weight is at least $q$. Since the weight of every vertex is at most $\Delta q$, the weight of $\sigma$ is at most $q + \Delta q$. We then add $\sigma$ to $\Sigma$, delete it from $P$ and continue. Consider the final set $\Sigma$ of segments. For each segment $\sigma$, we add a new cluster $C_\sigma$ to $C$. Cluster $C_\sigma$ consists of the union of $\sigma$, the graphs $G_{v_i}$ for each $v_i$ that is connected to a vertex of $\sigma$ with an edge in $T'$, and the corresponding edge $(x_i, y_i)$. Clearly, $C_\sigma$ is a connected subgraph of $G[B]$, containing at least $q$ and at most $2\Delta q$ paths in $P$. We add all those paths to $\tilde{P}$, delete all vertices of $T_v$ from $T$, and continue to the next iteration.

At the end of this procedure, we obtain a collection $\tilde{P}$ of paths, and a collection $C$ of disjoint connected subgraphs of $G$, such that each path $P \in \tilde{P}$ is contained in some $C \in C$, and each $C \in C$ contains at least $q$ and at most $4\Delta q$ paths from $\tilde{P}$. It now remains to show that $|\tilde{P}| \geq |P|/2$. We discard at most $q$ paths in the last iteration of the algorithm. Additionally, when $v = u_p$ is processed, if $|T_v \cap U| > 4\Delta q$, then path $P$ is also discarded, but at least $4\Delta q$ paths are added to $\tilde{P}$. Therefore, overall, $|\tilde{P}| \geq |P| - \frac{|P|}{4\Delta q + 1} - q \geq |P|/2$, since $|P| \geq \left(\frac{\kappa'}{2\Delta}\right)^2 \geq \frac{\kappa}{3\Delta^2}$, while $q = 2\Delta \beta_{\text{ARV}}(\kappa')$, and we have assumed that $\kappa \geq \frac{64\Delta^4 \beta_{\text{ARV}}(\kappa)}{\alpha}$.

For each cluster $C \in C$, we select one path $P_C \in \tilde{P}$ that is contained in $C$, and we let $t_C$ be the terminal that serves as an endpoint of $P_C$. Let $\Gamma'_C \subset \Gamma'$ be the set of all vertices of $\Gamma'$ that serve as endpoints of paths in $\tilde{P}$ that are contained in $C$. Then $|\Gamma'_C| \geq q$. We delete vertices from $\Gamma'_C$ as necessary, until $|\Gamma'_C| = q$ holds. Our final set $\mathcal{T}'$ of terminals is $\mathcal{T}' = \{t_C \mid C \in C\}$. Observe that $|\mathcal{T}'| \geq \frac{|\tilde{P}|}{4\Delta q} \geq \frac{|P|}{16\Delta^2 \beta_{\text{ARV}}(\kappa')} \geq \frac{\kappa' \kappa}{32\Delta^2 \beta_{\text{ARV}}(\kappa')} = \Omega\left(\frac{\kappa}{\Delta^2 \beta_{\text{ARV}}(\kappa)}\right)$, as required.

It now only remains to show that $\mathcal{T}'$ is node-well-linked in $G$. Let $\mathcal{T}_1, \mathcal{T}_2$ be any pair of equal-sized subsets of $\mathcal{T}'$. Let $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{T}_1' = \mathcal{T}_1 \setminus \mathcal{T}_2$, and $\mathcal{T}_2' = \mathcal{T}_2 \setminus \mathcal{T}_1$. We set up an $s$-$t$ flow network, by adding a source $s$ and connecting it to every vertex of $\mathcal{T}_1'$ with a directed edge, and adding a sink $t$, and connecting every vertex of $\mathcal{T}_2'$ to it. We also delete all vertices of $T'$ from the graph, and set all vertex capacities, except for $s$ and $t$, to $1$. From the integrality of flow, it is enough to show a valid $s$-$t$ flow of value $|\mathcal{T}_1'| = |\mathcal{T}_2'|$ in this flow network. This flow will be a concatenation of three flows, $F_1, F_2, F_3$.

We start by defining the flows $F_1$ and $F_3$. Consider some terminal $t \in \mathcal{T}_1' \cup \mathcal{T}_2'$. and let $C \in \mathcal{C}$ be the cluster to which $t$ belongs. Let $T_C$ be any spanning tree of $C$. Terminal $t$ sends one flow unit towards the vertices of $\Gamma'_C$, along the tree $T_C$, such that every vertex in $\Gamma'_C$ receives $1/q$ flow units. Let $F_1$ be the union of all these flows for all $t \in \mathcal{T}_1'$, and $F_3$ be the union of all these flows for all $t \in \mathcal{T}_2'$ (we will eventually think of the flow in $F_3$ as directed towards the terminals). Notice that for every vertex $v \in B \setminus \mathcal{T}'$, the total flow that goes through vertex $v$ or terminates at $v$ is at most 1. We say that the flow is of type 1 if it originates at a terminal in $\mathcal{T}_1'$, and it is of type 2 otherwise.

We now proceed to define flow $F_2$. For every cluster $C \in \mathcal{C}$, each vertex $v \in \Gamma'_C$ sends the $1/q$ flow units it receives to its special neighbor $u \in \Gamma$, along the edge $(v, u)$. Recall that every vertex $u \in \Gamma$ serves as a special neighbor of at most one vertex in $\Gamma'$. Let $\Gamma_1 \subset \Gamma$ be the set of vertices that receive flow of type 1, and $\Gamma_2 \subset \Gamma$ is the set of vertices that receive flow of type 2. Then $|\Gamma_1| = |\Gamma_2|$, and we
denote $|\Gamma_1| = \kappa^*$. It is enough to show that there is a flow $F_2$ in $G[A]$, where every vertex in $\Gamma_1$ sends $1/q$ flow units, every vertex in $\Gamma_2$ receives $1/q$ flow units, and the total vertex congestion due to this flow is at most $1/2$.

In order to define this flow, we set up a new flow network, by starting with $G[A]$, adding a source vertex $s$ that connects to every vertex in $\Gamma_1$ with a directed edge, and a sink $t$, to which every vertex in $\Gamma_2$ is connected with a directed edge. All edge capacities are unit. Since $\Gamma_1 \cup \Gamma_2 \subset S$ and $S$ is $1/\beta_{\text{ARV}}(\kappa')$-well-linked in $G[A]$, there is an $s$-$t$ flow of value $\kappa^*$ in this network, with edge-congestion at most $1/\beta_{\text{ARV}}(\kappa')$. In this flow, each vertex in $\Gamma_1$ sends one flow unit, and each vertex in $\Gamma_2$ receives one flow unit. The total flow through any vertex is at most $\Delta_{\beta_{\text{ARV}}}(\kappa')$. Scaling this flow down by factor $q = 2\Delta_{\beta_{\text{ARV}}}(\kappa')$, we obtain the flow $F_2$, where every vertex of $\Gamma_1$ sends $1/q$ flow units, every vertex in $\Gamma_2$ receives $1/q$ flow units, and the total vertex congestion is at most $1/2$. Combining together the flows $F_1, F_2, F_3$, we obtain the final flow $F$. From the integrality of flow, there is a set of $|\Gamma_1| = |\Gamma_2|$ disjoint paths connecting the vertices of $\Gamma_1$ to the vertices of $\Gamma_2$ in $G$.

A.3 Proof of Theorem 2.3

Let $T$ be any spanning tree of the graph $Z$. If $T$ contains at least $\ell$ leaves, then we are done. Assume now that $T$ contains fewer than $\ell$ leaves. We will next try to perform some improvement steps in order to increase the number of leaves in $T$.

Assume first that $T$ contains three vertices $a, b, c$, that have degree 2 in $T$ each, where $b$ is the unique child of $a$ and $c$ is the unique child of $b$, and assume further that there is an edge $(a, c)$ in $Z$. We can then delete the edge $(b, c)$ and add the edge $(a, c)$ to $T$. It is easy to see that the number of leaves increases, with the new leaf being $b$ (see Figure 5).

Assume now that $v$ is a degree-2 vertex in $T$, such that both its father $v_1$ and grandfather $v_2$ are degree-2 vertices. Moreover, assume that the unique child $v'_1$ of $v$ is a degree-2 vertex, and so is the unique grandchild $v'_2$ of $v$. Assume that an edge $(v, u)$ belongs to $Z$, where $u \neq v_1, v'_1$. Notice that if $u = v_2$ or $u = v'_2$, then we can apply the transformation outlined above. Therefore, we assume that $u \neq v_2$ and $u \neq v'_2$. Two cases are possible. First, if $u$ is not a descendant of $v$, then we add the edge $(u, v)$ to $T$, and delete the edge $(v_1, v_2)$ from $T$. Notice that the number of leaves increases, as two new vertices become leaves - $v_1, v_2$, while in the worst case at most one vertex stops being a leaf (vertex $u$). The second case is when $u$ is a descendant of $v$. Then we add an edge $(u, v)$ to $T$, and delete the edge $(v'_1, v'_2)$ from $T$. Again, the number of leaves increases by at least 1, since both $v'_1$ and $v'_2$ are now leaves. (See Figure 5).

We perform the above improvement step while possible. Let $T$ denote the final tree, where no such operation is possible. If $T$ has at least $\ell$ leaves, then we are done. Assume therefore that $T$ has fewer than $\ell$ leaves. Then the number of inner vertices of $T$ whose degree is more than 2 in $T$ is at most $\ell$. If $P$ is a maximal 2-path in $T$, then the child of its lowermost vertex must be either a leaf or a vertex whose degree is more than 2 in $T$. Therefore, there are at most $2\ell$ maximal 2-paths in $T$, and at least one such path must contain at least $\frac{n-2\ell}{2\ell} \geq p + 4$ vertices. Let $P'$ be the path obtained from $P$ by deleting the first two and the last two vertices. Then $P'$ contains at least $p$ vertices, and, since no improvement step was possible, $P'$ must be a 2-path in $Z$.

A.4 Proof of Lemma 2.3

We prove the lemma via the stable matching theorem. We use arguments very similar to those used by Conforti, Hassin and Ravi [CHR03] for re-routing flow-paths. We set up an instance of the stable matching problem in a multi-graph. In this problem, we are given a complete bipartite multigraph
Given the sets \( \mathcal{X} \) and \( \mathcal{Y} \) of paths, construct a bipartite multigraph \( G = (A, B, E) \), where \( A = \{v_P \mid P \in \mathcal{X}\} \), and initially \( B = \{v_Q \mid Q \in \mathcal{Y}\} \). Additionally, we add \(|X| - |Y|\) dummy vertices to \( B \) to ensure that \(|A| = |B|\). If paths \( P \in \mathcal{X} \) and \( Q \in \mathcal{Y} \) share \( k \) vertices (in addition to \( s \)), then we add \( k \) parallel edges between \( v_P \) and \( v_Q \). Each such edge is labeled with the corresponding vertex common to \( P \) and \( Q \). We then add dummy edges to \( G \) as needed to turn it into a complete bipartite graph.

We now define the preference lists of each vertex of \( G \), over the edges incident to it. For each dummy vertex \( v \in B \), its preference list contains all edges incident to it in an arbitrary order.

Given a vertex \( v_P \in A \), its preference order over the non-dummy edges incident on it is defined as follows. Recall that each such non-dummy edge \( e \) corresponds to some vertex \( u_e \) that appears on \( P \). The ordering of such edges \( e \) is in the reverse order of the appearance of the corresponding vertices \( u_e \) on path \( P \). The dummy edges incident on \( v_P \) appear in an arbitrary order at the end of the list. The preference lists for the vertices \( v_Q \in B \) are defined as follows. The non-dummy edges \( e \) incident on \( v_Q \) are ordered in the same order as the appearance of the corresponding vertices \( u_e \) on path \( Q \). The dummy edges incident on \( v_Q \) appear in an arbitrary order at the end of the preference list.

Let \( M \) be any stable matching in \( G \). We let \( \mathcal{X}' \subset \mathcal{X} \) contain all paths \( P \in \mathcal{X} \), such that the edge incident on \( v_P \) in \( M \) is a dummy edge. Notice that if \( e \in M \) is a non-dummy edge, then both its endpoints must be non-dummy vertices, so \(|X'|| \geq |X| - |Y|\) must hold.

Next, consider some path \( Q \in \mathcal{Y} \). If the edge \( e \) incident on \( v_Q \) in \( M \) is a dummy edge, then we define \( \hat{Q} = Q \), and we say that \( \hat{Q} \) is a type-1 path. Otherwise, let \( e = (v_Q, v_P) \), and let \( u \in Q \) be the vertex corresponding to \( e \). (Notice that \( P \notin \mathcal{X}' \) in this case). We let \( \hat{Q} \) be the union of the prefix of \( Q \), between its start vertex and \( u \), and the suffix of \( P \) between \( u \) and \( s \), and we say that \( \hat{Q} \) is a type-2 path. Notice that \( \hat{Q} \) has the same endpoints as \( Q \).
Let $\mathcal{Y}' = \{ \hat{Q} \mid Q \in \mathcal{Y} \}$. Given a pair $P, Q$ of directed paths that share the same destination vertex $s$, we say that $P$ and $Q$ intersect if they share any vertex in addition to $s$.

It now only remains to show that the paths in $\mathcal{X}' \cup \mathcal{Y}'$ do not intersect each other. Clearly, the paths in $\mathcal{X}'$ do not intersect each other, as $\mathcal{X}' \subseteq \mathcal{X}$, and the paths in $\mathcal{X}$ do not intersect each other.

Consider some pair of paths $P \in \mathcal{X}'$ and $\hat{Q} \in \mathcal{Y}'$. If $\hat{Q}$ is a type-1 path, then $P$ and $\hat{Q}$ do not intersect. Indeed, assume for contradiction that they share a vertex $u \neq s$, and let $e = (v_P, v_Q)$ be the edge of $G$ corresponding to the vertex $u$. Then both $P$ and $Q$ currently participate in $M$ via dummy edges, while both of them prefer the edge $e$, a contradiction.

Assume now that $\hat{Q}$ is a type-2 path, and assume that it is a union of a suffix of some path $P' \in \mathcal{X}$ and a prefix of $Q$, performed via a vertex $u$ shared by these paths, where the edge $e = (v_{P'}, v_Q)$ representing the vertex $u$ belongs to $M$. Assume for contradiction that $P$ and $\hat{Q}$ share some vertex $u' \neq s$. Since $P$ and $P'$ do not intersect, $u'$ must lie before the vertex $u$ on $Q$. Let $e' = (v_P, v_{Q'})$ be the edge of $G$ corresponding to $u'$. Then $Q$ prefers $e'$ over $e$, as $u'$ appears before $u$ on $Q$, and $P$ prefers $e'$ to its current edge, since currently the edge incident on $v_P$ that belongs to $M$ must be a dummy edge (as $v_P \in \mathcal{X}'$).

Finally, consider any pair $\hat{Q}, \hat{Q}' \in \mathcal{Y}'$ of paths and assume for contradiction that they intersect. Assume that $\hat{Q}$ is a union of a prefix $\sigma_1$ of $Q$ and a suffix $\sigma_2$ of some (possibly empty) path $P \in \mathcal{X}$, connected via a vertex $u$ (if $P = \emptyset$, then $u = s$). Similarly, assume that $\hat{Q}'$ is a union of a prefix $\sigma'_1$ of $Q'$ and a suffix $\sigma'_2$ of some path $P' \in \mathcal{X}$, connected via a vertex $u'$. Since the paths in $\mathcal{X}$ do not intersect each other, and the same is true for all paths in $\mathcal{Y}$, this can only happen if either $\sigma_1$ and $\sigma'_2$ share a vertex different from $s$, or $\sigma_2$ and $\sigma'_1$ share a vertex different from $s$. Assume without loss of generality that the former happens, and let $u''$ be the shared vertex. Then $u''$ appears before $u$ on $Q$, and it appears after $u'$ on $P'$. Let $e = (v_{P'}, v_{P''})$ be the edge representing $u''$ in graph $G$. Then $v_Q$ prefers $u''$ to the current edge incident on it that belongs to $M$, and so does $v_{P'}$, contradicting the fact that $M$ is a stable matching.

### A.5 Proof of Theorem 2.5

Since $G$ has treewidth $k$, we can efficiently find a set $X$ of $\Omega(k)$ vertices of $G$ with properties guaranteed by Lemma 2.2. Assume for simplicity that $|X|$ is even.

Using the cut-matching game and Theorem 2.4, we can embed an expander $H = (X, F)$ into $G$ as follows. Each iteration $j$ of the cut-matching game requires the matching player to find a matching $M_j$ between a given partition of $X$ into two equal-sized sets $Y_j, Z_j$. From Lemma 2.2, there exist a collection $\mathcal{P}_j$ of paths from $Y_j$ to $Z_j$, that cause congestion at most $1/\alpha^*$ on the vertices of $G$; these paths naturally define the required matching $M_j$. The game terminates in $\gamma_{\text{CMG}}(|X|)$ steps. Consider the collection of paths $\mathcal{P} = \bigcup_j \mathcal{P}_j$ and let $G'$ be the subgraph of $G$ induced by the union of the edges in these paths. Let $H = (X, F)$ be the expander on $X$ created by the union of the edges in $\bigcup_j M_j$. By the construction, for each $j$, any node $v$ of $G$ appears in at most $1/\alpha^*$ paths in $\mathcal{P}_j$. Therefore, the maximum degree in $G'$ is at most $2\gamma_{\text{CMG}}(|X|)/\alpha^* = O(\log^3 k)$, and moreover the node (and hence also edge) congestion caused by the set $\mathcal{P}$ of paths in $G$ is also upper bounded by the same quantity. We apply the algorithm $\mathcal{A}_{\text{ARV}}$ to the sparsest cut instance defined by the graph $X$, where all vertices of $X$ serve as terminals. If the outcome is a cut whose sparsity is less than $\alpha_{\text{CMG}}(|X|)$, then the algorithm fails; we discard the current graph $X$ and repeat the algorithm again. Otherwise, if the outcome is a cut of sparsity at least $\alpha_{\text{CMG}}(|X|)$, then we are guaranteed that $X$ is an $\alpha_{\text{CMG}}(|X|)/\beta_{\text{ARV}}(|X|) = \Omega(\sqrt{\log |X|})$-expander, and in particular, it is an $\alpha$-expander, for $\alpha = 1/2$.

Since each execution of the cut-matching game is guaranteed to succeed with a constant probability,
after $|X|$ such executions, the algorithm is guaranteed to succeed with high probability.

Since $H = (X, F)$ is an $\alpha$-expander, $X$ is $\alpha$-well-linked in $H$. Since $H$ is embedded in $G'$ with congestion at most $2\gamma_{\text{CMG}}(|X|)/\alpha^*$, $X$ is $\frac{\alpha^*}{\gamma_{\text{CMG}}(|X|)}$-well-linked in $G'$. Since the maximum degree in $G'$ is at most $2\gamma_{\text{CMG}}(|X|)/\alpha^* = O(\log^3 k)$, we can apply Theorem 2.2 to find a subset $X' \subseteq X$ of $\Omega\left(\frac{k}{\log^{\frac{r}{r+k}} k}\right)$ vertices, such that $X'$ is node-well-linked in $G'$.

## B Proof of Theorem 3.1

Let $h_0 = \lceil\sqrt{\frac{\ell}{2}}\rceil$. We assume that $h$ is large enough, so $h_0 > 128$ (otherwise, we can return a $1 \times 1$ grid - a single vertex). Fix some cluster $S_i$, for $2 \leq i \leq h - 1$. A set $L_i$ of $h$ disjoint paths, that are completely contained in $S_i$, and connect the vertices of $A_i$ to the vertices of $B_i$, is called an $A_i$-$B_i$ linkage. Since the sets $A_i, B_i$ of vertices are linked in $G[S_i]$, such a linkage $L_i$ exists and can be found efficiently.

Given an $A_i$-$B_i$ linkage $L_i$, we associate a graph $H_i = H(S_i, L_i)$ with the set $S_i$ as follows. The vertices of $H_i$ are $U_i = \{u_P \mid P \in L_i\}$, and there is an edge between $u_P$ and $u_{P'}$ iff there is a path $\beta_{P,P'}$, completely contained in $G[S_i]$, whose first vertex belongs to $P$, last vertex belongs to $P'$, and the inner vertices do not belong to any paths in $L_i$. Notice that since $G[S_i]$ is a connected graph, so is $H(S_i, L_i)$ for any $A_i$-$B_i$ linkage $L_i$. We say that a linkage $L_i$ is good for set $S_i$ if the corresponding graph $H(S_i, L_i)$ contains no 2-path of length $h_0$. Recall that Theorem 2.3 guarantees that if $L_i$ is good for $S_i$, then we can find a spanning tree $T_i$ of $H(S_i, L_i)$ with at least $\frac{\ell}{2(h_0 + 3)} \geq h_0$ leaves.

We say that a set $S_i \in \mathcal{S}$ is even if $i$ is even with $i < h$, and otherwise we say that $S_i$ is odd. In the following theorem, we show how to find a grid minor of size $(\Omega(\sqrt{h}) \times \Omega(\sqrt{h}))$ if we are given a good linkage $L_i$ for every even cluster $S_i$.

**Theorem B.1** Assume that we are given, for every even cluster $S_i$, a good linkage $L_i$. Then we can efficiently find a grid minor of size $(\Omega(\sqrt{h}) \times \Omega(\sqrt{h}))$ in $G$.

**Proof:** Let $\mathcal{H}$ be the set of $h$ disjoint paths, obtained by concatenating $P_1, L_2, P_2, L_3, \ldots, L_{h-1}, P_h$. Consider some even cluster $S_i$, and the corresponding path graph $H_i = H(S_i, L_i)$. Since $H_i$ contains no 2-path of length $h_0/4$, from Theorem 2.3, we can find a spanning tree $T_i$ of $H_i$ with at least $h_0$ leaves. Let $L_i' \subseteq L_i$ be the set of paths whose corresponding vertices serve as leaves of $T_i$. If $L_i'$ contains more than $h_0$ paths, remove paths from $L_i'$ arbitrarily, until $|L_i'| = h_0$ holds. Observe that any pair $P, P' \in L_i'$ of paths can be connected by a path $\beta_{P,P'}$ contained in $G[S_i]$, that does not intersect any other paths in $L_i'$ (by the construction of the graph $H_i$).

Let $S_{2i}, S_{2i+2}$ be any pair of consecutive even clusters, and let $S_{2i+1}$ be the odd cluster lying between $S_{2i}$ and $S_{2i+2}$. We call $S_{2i+1}$ the connector cluster for $S_{2i}$ and $S_{2i+2}$. We define $L_{2i}^+ \subseteq P_{2i}$ to contain all paths $P \in P_{2i}$, such that one endpoint of $P$ belongs to $V(L_{2i}^-)$ (where it must serve as the last endpoint of some path). Let $A'_{2i+1} \subseteq A_{2i+1}$ be the set of the endpoints of paths in $L_{2i}^+$ that belong to $S_{2i+1}$. Similarly, we define $L_{2i+2}^- \subseteq P_{2i+1}$ to contain all paths $P \in P_{2i+1}$, whose last vertex belongs to $V(L_{2i+2}^+)$ (where it must serve as the first endpoint of some path). Let $B'_{2i+1} \subseteq B_{2i+1}$ be the set of the endpoints of paths in $L_{2i+2}^-$ that belong to $S_{2i+1}$. (See Figure 6).

Using the fact that $A_{2i+1}$ and $B_{2i+1}$ are linked in $G[S_{2i+1}]$, we can find a set $R_{2i+1}$ of $h_0$ disjoint paths that are contained in $S_{2i+1}$ and connect $A'_{2i+1}$ to $B'_{2i+1}$. To make the notation consistent, for the first even cluster $S_2$, we define $L_2^+ = \emptyset$, and for the last even cluster $S_{\ell}$ (where $\ell = h - 1$ or $\ell = h - 2$), we define $L_{\ell}^- = \emptyset$.  

50
Figure 6: Reconnecting paths in $L'_{2i}$ and $L'_{2i+2}$ using cluster $S_{2i+1}$ as a connector cluster.

We now define a new set $\mathcal{H}'$ of horizontal paths, obtained by the concatenation of all paths $L_i^-, L_i^+, L_i^+$ where $S_i$ is an even cluster, and paths $R_j$, where $S_j$ is a connector cluster. The resulting set $\mathcal{H}'$ contains $h_0$ disjoint paths, and it has the property that the paths in $\mathcal{H}'$ traverse the even clusters $S_{2i}$ in their natural order; for each such cluster $S_{2i}$, every path in $\mathcal{H}'$ contains exactly one path in $L_i^+$ as a sub-path, and, except for the paths in $L_i^+$, the paths in $\mathcal{H}'$ do not contain any other vertices of $S_{2i}$.

Let $z$ be the largest integer, such that $\sqrt{z}$ is an integer; $z^2 \leq h/16$; and $2z + \sqrt{z} \leq h_0$. It is easy to see that $z = \Omega(\sqrt{h})$. We will construct a $(z \times z)$-grid minor in $G$. We build a new path-of-sets system $(S' = (S'_1,\ldots,S'_{4z}),\bigcup_{i=1}^{4z^2-1} P'_i)$, as follows. For $1 \leq i \leq 4z^2$, we let $S'_i = S_{2i}$, and for $1 \leq i < 4z^2$, set $P'_i$ consists of all the segments of the paths in $\mathcal{H}'$ that connect $S'_i$ to $S'_{i+1}$. In other words, $P'_i = L_{2i}^+ \cup R_{2i+1} \cup L_{2i+2}$. Notice that each path $P \in \mathcal{H}'$ visits the clusters $S'_1,\ldots,S'_{4z^2}$ in this order, and for each $1 \leq i \leq 4z^2$, $P \cap G[S'_i]$ is a path. Moreover, for each $P, P' \in \mathcal{H}'$, and for each $1 \leq i \leq 4z^2$, there is a path $\beta'_{P,P'}$ in $G[S'_i]$, connecting a vertex of $P$ to a vertex of $P'$, so that $\beta'_{P,P'}$ is internally disjoint from all paths in $\mathcal{H}'$.

Finally, we partition the path-of-sets system $(S' = (S'_1,\ldots,S'_{4z}),\bigcup_{i=1}^{4z^2-1} P'_i)$ into $z$ smaller disjoint systems of width $4z$ each: for $1 \leq j \leq 4z$, the $j$th path-of-set system consists of the clusters $S'_{4z(j-1)+1},\ldots,S'_{4zj}$, and the set $\bigcup_{i=4z(j-1)+1}^{4zj-1} P'_i$ of paths. Let $\mathcal{H}'_j$ denote the set of the segments of $\mathcal{H}'$ that are contained in the $j$th path-of-set system.
Let $X$ be the $(z \times z)$-grid. The high-level plan is to show an embedding of $X$ into $G$ which certifies that $X$ is a minor of $G$. Each vertex $v \in V(X)$ will be mapped to a sub-path $H_v$ of a path $P \in \mathcal{H}'$. The sub-path $H_v$ will correspond to an interval $[a, b]$ where $a < b$ are indices in $\{1, \ldots, 4z\}$: that is, $H_v$ is the segment of $P$ starting from the first vertex of $P$ lying in the $a$th path-of-sets system, and ending with the last vertex of $P$ lying in the $b$th path-of-sets system. For all $v, v' \in V(X)$, where $v \neq v'$, we will have the property that $H_v$ and $H_{v'}$ are disjoint. Note that that $H_v$ and $H_{v'}$ can be sub-paths of the same path $P$ from $\mathcal{H}'$, but their corresponding intervals will be disjoint. Each edge $(v, v') \in E(X)$ will be mapped to a path connecting $H_v$ and $H_{v'}$ in some cluster; that is, if $H_v$ is a sub-path of $P$ and $H_{v'}$ is a sub-path of $P'$, there will be a cluster $S_j$ such that $(v, v')$ is mapped to the path $\beta_{P, P'}^{j}$. For this to be feasible, the intervals $[a, b]$ and $[a', b']$ for $H_v$ and $H_{v'}$ need to overlap. Each cluster $S_j$ will be used for at most one edge of $X$. It is easy to see that such an embedding proves that $X$ is a minor of $G$.

Now we describe the details of the embedding. We partition the $(z \times z)$-grid $X$ into $z$ disjoint $(\sqrt{z} \times \sqrt{z})$ grids, numbered $X_1, \ldots, X_z$ in the top-to-bottom and then left-to-right order (see Figure 7). For $1 \leq j \leq z$, let $U_j = \bigcup_{i=1}^{j} V(X_i)$, and let $\Gamma_j \subseteq U_j$ be the set of vertices incident on the edges of $E(X) \setminus U_j$. Notice that $|\Gamma_j| \leq z + \sqrt{z}$ for all $j$, and for all $j < j' < j''$, if a vertex of $X$ belongs to $\Gamma_j \cap \Gamma_{j''}$, then it must also belong to $\Gamma_j$.

Let $V_j = \Gamma_{j-1} \cup V(X_j)$, and let $Y_j$ be the sub-graph of $X$, whose vertex set is $V_j$, and the edge set consists of all the edges of $E(X_j)$, and $E(X_j, \Gamma_{j-1})$. We observe that $|V_j| \leq 2z + \sqrt{z} \leq h_0$, and $|E(Y_j)| \leq 4z$. Notice that for $j < j' < j''$, if $v \in V_j \cap V_{j''}$, then $v \in V_j$ must hold. Notice also that for each edge $e$ of $X$, there is a unique index $1 \leq j \leq z$, such that $e \in E(Y_j)$. See Figure 7.

The idea is to use the $j$th path-of-sets system to realize the graph $Y_j$. We will define a mapping $g_j : V_j \rightarrow \mathcal{H}_j$, where each vertex of $V_j$ is mapped to a distinct path of $\mathcal{H}_j$. Since $|V_j| \leq h_0 = |\mathcal{H}_j|$, such a mapping exists. Let $e_1, \ldots, e_N$ be the edges of $Y_j$, where $N \leq 4z$. We will use the $i$th cluster of the $j$th path-of-set system to implement the edge $e_i$: if $e_i = (v, v')$, $P = g_j(v)$ and $P' = g_j(v')$, then edge $e_i$ will be embedded into the path $\beta_{P, P'}^{j_1}\beta_{P, P'}^{j_2}\beta_{P, P'}^{j_3}\ldots\beta_{P, P'}^{j_N}$. Note that $Y_j$ may include isolated vertices with no edges incident to them (such a vertex would be in $\Gamma_{j-1}$) but these vertices are also mapped to paths in $\mathcal{H}_j$.

Finally, it remains to define the mappings $g_j : V_j \rightarrow \mathcal{H}_j$. We need to define these mappings in such a way, that for each $v \in V(X)$, all paths to which $v$ is mapped are sequential sub-paths of the same path in $\mathcal{H}'$. We achieve this as follows. Mapping $g_j$ : $V_j$ $\rightarrow$ $\mathcal{H}_j$ is an arbitrary injection. Assume now that we have defined $g_1, \ldots, g_j$. We show how to define $g_{j+1}$. For $v \in V_j \cap V_{j+1}$, let $H \in \mathcal{H}_j$ be the path, such that $g_j(v)$ is a sub-path of $H$. Let $H' \in \mathcal{H}_{j+1}$ be a sub-path of $H$. We then define $g_{j+1}(v) = H'$. The vertices in $V_{j+1} \setminus V_j$ are mapped to the remaining paths in $\mathcal{H}_{j+1}'$ arbitrarily, so that each vertex is mapped to a distinct path.
Theorem B.2 Fix some cluster $L$.

Fix some cluster $L$ and compute a good linkage $v(v)$. After applying this algorithm to each cluster $L$, we terminate the algorithm and return this grid minor. Otherwise, if $v(v)$ is not good for $v(v)$, either find an $(\Omega(\sqrt{h}) \times \Omega(\sqrt{h}))$-grid minor inside $G[S_i]$ or finds a new $A_i$-$B_i$ linkage $\mathcal{L}_i'$, such that the number of the degree-2 vertices in $H(S_i, \mathcal{L}_i')$ is strictly smaller than the number of the degree-2 vertices in $H(S_i, \mathcal{L}_i)$.

We prove Theorem B.2 below, and complete the proof of Theorem B.1 here. Fix some cluster $S_i$, for $2 \leq i \leq h - 1$. Let $\mathcal{L}_i$ be any $A_i$-$B_i$ linkage, and assume that $\mathcal{L}_i$ is not good for $S_i$. Then there is an efficient algorithm, that either finds an $(\Omega(\sqrt{h}) \times \Omega(\sqrt{h}))$-grid minor inside $G[S_i]$, or finds a new $A_i$-$B_i$ linkage $\mathcal{L}_i'$, such that the number of the degree-2 vertices in $H(S_i, \mathcal{L}_i')$ is strictly smaller than the number of the degree-2 vertices in $H(S_i, \mathcal{L}_i)$.

We prove Theorem B.2 below, and complete the proof of Theorem B.1 here. Fix some cluster $S_i$, for $2 \leq i \leq h - 1$. We start with an arbitrary $A_i$-$B_i$ linkage $\mathcal{L}_i$, and iterate. While $\mathcal{L}_i$ is not good for $S_i$, we apply Theorem B.2 to the current linkage $\mathcal{L}_i$. If the outcome is an $(\Omega(\sqrt{h}) \times \Omega(\sqrt{h}))$-grid minor, then we terminate the algorithm and return this grid minor. Otherwise, if $\mathcal{L}_i'$ is a good linkage for $S_i$, we terminate the algorithm and return $\mathcal{L}_i'$. Otherwise, we replace $\mathcal{L}_i$ with $\mathcal{L}_i'$ and continue to the next iteration. After $O(h)$ iterations, the number of degree-2 vertices in the graph $H(S_i, \mathcal{L}_i)$ is guaranteed to fall below $h_0$ (unless the algorithm terminates earlier). Therefore, we obtain an efficient algorithm, that, given any cluster $S_i$, for $2 \leq i \leq h - 1$, either finds an $(\Omega(\sqrt{h}) \times \Omega(\sqrt{h}))$-grid minor inside $G[S_i]$, or computes a good linkage $\mathcal{L}_i$ for $S_i$.

Applying Theorem B.2 will finish the proof. It now remains to prove Theorem B.2.

Figure 8: A $9 \times 9$ grid. In picture on left, the vertex set $\Gamma_4$ is shown as unfilled circles. In picture on right, the graph $Y_5$ is highlighted; the vertices are unfilled (circles are $\Gamma_4$ and squares are $V(X_5)$) and the edges are shown in dark.
B.1 Proof of Theorem B.2

We start with the $A_i$-$B_i$ linkage $L_i$, and assume that $L_i$ is not good for $S_i$. Then there is a 2-path $R^* = (u_{P_0}, \ldots, u_{P_{h_0-1}})$ in the corresponding graph $H_i = H(S_i, L_i)$. Let $z = \lfloor h_0/4 \rfloor - 1$, so $z = \Omega(\sqrt{h})$. Consider the following four subsets of paths: $P_1 = \{P_1, \ldots, P_z\}$, $P_2 = \{P_{z+1}, \ldots, P_{2z}\}$, $P_3 = \{P_{2z+1}, \ldots, P_{3z}\}$, and $P_4 = \{P_{3z+1}, \ldots, P_{4z}\}$, whose corresponding vertices participate in the 2-path $R^*$. (Notice that $P_0 \notin P_1$, but the degree of $u_{P_0} = 2$ in $H_i$ - we use this fact later). Let $X \subseteq A_i$ be the set of endpoints of the paths in $P_2$ that belong to $A_i$, and let $Y \subseteq B_i$ be the set of paths in $P_4$ that belong to $B_i$ (see Figure 9). Since $A_i, B_i$ are linked in $S_i$, we can find a set $Q$ of $z$ disjoint paths connecting $X$ to $Y$ in $G[S_i]$. We view the paths in $Q$ as directed from $X$ to $Y$.

Let $Q \in Q$ be any such path. Observe that, since $R^*$ is a 2-path in $H_i$, path $Q$ has to either intersect all paths in $P_1$, or all paths in $P_3$ before it reaches $Y$. Therefore, it must intersect $P_{z+1}$ or $P_{2z}$. Let $u$ be the last vertex of $Q$ that belongs to $P_{z+1} \cup P_{2z}$. Let $Q'$ be the segment of $Q$ starting from $v$ and terminating at a vertex of $Y$. Assume first that $v \in P_{z+1}$. We say that $Q$ is a type-1 path in this case. Let $u$ be the first vertex on $Q'$ that belongs to $P_0$. (Such a vertex must exist again due to the fact that $R^*$ is a 2-path.) Let $Q''$ be the segment of $Q'$ between $v$ and $u$. Then $Q''$ intersects every path in $P_1 \cup \{P_0, P_{z+1}\}$, and does not intersect any other path in $L_i$, while $|V(Q'') \cap V(P_0)| = |V(Q'') \cap P_{z+1}| = 1$. (see Figure 9).

Similarly, if $v \in P_{2z}$, then we say that $Q$ is a type-2 path. Let $u$ be the first vertex of $Q'$ that belongs to $P_{3z+1}$, and let $Q''$ be the segment of $Q'$ between $u$ and $v$. Then $Q''$ intersects every path in $P_3 \cup \{P_{2z} \cup P_{3z+1}\}$, and does not intersect any other path in $L_i$, while $|V(Q'') \cap V(P_{2z})| = |V(Q'') \cap V(P_{3z+1})| = 1$.

Figure 9: Two examples for paths in $Q$ - a type-1 and a type-2 path - are shown in red, with the $Q^*$ segment highlighted.
Clearly, either at least half the paths in $Q$ are type-1 paths, or at least half the paths in $Q$ are type-2 paths. We assume w.l.o.g. that the former is true. Let $Q'$ be the set of the sub-paths $Q^*$ for all type-1 paths $Q \in Q$, that is, $Q' = \{Q^* \mid Q \in Q \text{ and } Q \text{ is type-1}\}$. Then $|Q'| \geq z/2 = \Omega(\sqrt{h})$.

The rest of the proof is based on the following idea. We will show that either the graph obtained from the union of the paths in $Q' \cup P_1$ is a planar graph, in which case we recover a grid-minor directly, or we will find a new linkage $L_i'$, such that $H(S_i, L_i')$ contains fewer degree-2 vertices than $H(S_i, L_i)$. To accomplish this we will iteratively simplify the intersection pattern of the paths in $Q'$ and $P_1$.

The algorithm performs a number of iterations. Throughout the algorithm, the set $Q'$ of paths remains unchanged. The input to every iteration consists of a set $P_1'$ of paths, such that the following hold:

- $L_i' = (L_i \setminus P_1) \cup P_1'$ is a valid $A_i$-$B_i$ linkage;
- The graphs $H_i$ and $H_i' = H(S_i, L_i')$ are isomorphic to each other, where the vertices $u_P$ for $P \notin P_1$ are mapped to themselves;
- Every path in $Q'$ intersects every path in $P_1' \cup \{P_0, P_{z+1}\}$, and no other paths of $L_i'$.

The input to the first iteration is $P_1' = P_1$. Throughout the algorithm, we maintain a graph $\tilde{H}$ - the sub-graph of $G$ induced by the edges participating in the paths of $P_1' \cup Q'$. We define below two combinatorial objects: a bump and a cross. We show that if $\tilde{H}$ has either a bump or a cross, then we can find a new set $P_1''$ of paths, such that $L_i'' = (L_i \setminus P_1) \cup P_1''$ is a valid $A_i$-$B_i$ linkage. Moreover, either $H_i'' = H(S_i, L_i'')$ contains fewer degree-2 vertices than $H_i'$, or the two graphs are isomorphic to each other. In the former case, we terminate the algorithm and return the linkage $L_i''$. In the latter case, we show that we obtain a valid input to the next iteration, and $|E(Q') \cup E(P_1')| > |E(Q') \cup E(P_1'')|$. In other words, the number of edges in the graph $\tilde{H}$ goes down in every iteration. We also show that, if $\tilde{H}$ contains no bump and no cross, then a large sub-graph of $\tilde{H}$ is planar, and contains a grid minor of size $(\Omega(h_0) \times \Omega(h_0))$. Therefore, after $|E(G[S_i])|$ iterations the algorithm is guaranteed to terminate with the desired output. We now proceed to define the bump and the cross, and their corresponding actions. A useful observation is that for any $A_i$-$B_i$ linkage $L$, the corresponding graph $H(S_i, L)$ is a connected graph, since $G[S_i]$ is connected.

**A bump.** Let $P_1'$ be the current set of paths, and $L_i' = (L_i \setminus P_1) \cup P_1'$ the corresponding linkage. We say that the corresponding graph $\tilde{H}$ contains a bump, if there is a sub-path $Q'$ of some path $Q \in Q'$, whose endpoints, $s$ and $t$, both belong to the same path $P_j \in P_1'$, and all inner vertices of $Q'$ are disjoint from all paths in $P_1'$. (See Figure 10). Let $a_j \in A_i, b_j \in B_i$ be the endpoints of $P_j$, and assume that $s$ appears before $t$ on $P_j$, as we traverse it from $a_j$ to $b_j$. Let $P_j'$ be the path obtained from $P_j$, by concatenating the segment of $P_j$ between $a_j$ and $s$, the path $Q'$, and the segment of $P_j$ between $t$ and $b_j$.

![Figure 10: A bump and the corresponding action.](image)
Let $P_i'''$ be the set of paths obtained by replacing $P_j$ with $P_j'$ in $P_i'$, and let $L_i'' = (L_i' \setminus P_i') \cup P_i''' = (L_i \setminus P_i) \cup P_i''$. It is immediate to verify that $L_i''$ is a valid $A_i$-$B_i$ linkage. Let $H_i' = H(S_i, L_i')$, and $H_i'' = H(S_i, L_i'')$, and let $E'$ be the set of edges in the symmetric difference of the two graphs (that is, edges that belong to exactly one of the two graphs). Then for every edge in $E'$, both endpoints must belong to the set $\{u_{P_{i-1}}, u_{P_i}, u_{P_{i+1}}\}$. In particular, the only vertices whose degree may be different in the two graphs are $u_{P_{i-1}}, u_{P_i}, u_{P_{i+1}}$. If the degree of any one of these three vertices is different in $H_i''$ and $H_i'$, then, since their degrees are 2 in both $H_i'$ and the original graph $H_i$, we obtain a new $A_i$-$B_i$ linkage $L_i''$, such that $H(S_i, L_i'')$ contains fewer degree-2 vertices than $H_i$. Otherwise, if the degrees of all three vertices remain equal to 2, then it is immediate to verify that $H_i''$ is isomorphic to $H_i'$, where each vertex is mapped to itself, except that we replace $u_{P_j}$ with $u_{P_j'}$. It is easy to verify that all invariants continue to hold in this case. Let $\tilde{H}$ be the graph obtained by the union of the paths in $P_i'$ and $Q'$, and define $\tilde{H}'$ similarly for $P_i'''$ and $Q'$. Then $\tilde{H}'$ contains fewer edges than $\tilde{H}$, since the portion of the path $P_j$ between $s$ and $t$ belongs to $\tilde{H}$ but not to $\tilde{H}'$.

**A cross.** Suppose we are given two disjoint paths $Q_j', Q_j''$, where $Q_j'$ is a sub-path of some path $Q_1 \in Q'$, and $Q_j''$ is a sub-path of some path $Q_2 \in Q'$ (possibly $Q_1 = Q_2$). Assume that the endpoints of $Q_j'$ are $s_1, t_1$ and the endpoints of $Q_j''$ are $s_2, t_2$. Moreover, suppose that $s_1, s_2$ appear on some path $P_j \in P_i'$ in this order, and $t_2, t_1$ appear on $P_{j+1} \in P_i'$ in this order (where the paths in $P_i'$ are directed from $A_i$ to $B_i$), and no inner vertex of $Q_j'$ or $Q_j''$ belongs to any path in $P_i$. We then say that $Q_j', Q_j''$ are a cross. (See Figure 11)

![Figure 11: A cross and the corresponding action.](image)

Given a cross as above, we define two new paths, as follows. Assume that the endpoints of $P_j$ are $a_j \in A_i$, $b_j \in B_i$, and similarly the endpoints of $P_{j+1}$ are $a_{j+1} \in A_i$, $b_{j+1} \in B_i$. Let $P_j'$ be obtained by concatenating the segment of $P_j$ between $a_j$ and $s_1$, the path $Q_j'$, and the segment of $P_{j+1}$ between $t_1$ and $b_{j+1}$. Let $P_{j+1}'$ be obtained by concatenating the segment of $P_{j+1}$ between $a_{j+1}$ and $t_2$, the path $Q_j''$, and the segment of $P_j$ between $s_2$ and $b_j$. We obtain the new set $P_i''$ of paths by replacing $P_j, P_{j+1}$ with $P_j', P_{j+1}'$ in $P_i'$. Let $L_i'' = (L_i' \setminus P_i') \cup P_i''' = (L_i \setminus P_i) \cup P_i''$. It is immediate to verify that $L_i''$ is a valid $A_i$-$B_i$ linkage. As before, let $H_i' = H(S_i, L_i')$, and $H_i'' = H(S_i, L_i'')$, and let $E'$ be the set of edges in the symmetric difference of the two graphs. Then for every edge in $E'$, both endpoints must belong to the set $\{u_{P_{i-1}}, u_{P_i}, u_{P_{i+1}}, u_{P_{i+j+2}}\}$. The only vertices whose degree may be different in the two graphs are $u_{P_{i-1}}, u_{P_i}, u_{P_{i+1}}, u_{P_{i+j+2}}$. If the degree of any one of these four vertices is different in $H_i''$ and $H_i'$, then, since their degrees are 2 in both $H_i'$ and the original graph $H_i$, we obtain a new linkage $L_i''$, such that $H(S_i, L_i'')$ contains fewer degree-2 vertices than $H_i$. Otherwise, if the degrees of all four vertices remain equal to 2, then it is immediate to verify that $H_i''$ is isomorphic to $H_i'$, where each vertex is mapped to itself, except that we replace $u_{P_j}, u_{P_{j+1}}$ with $u_{P_j'}, u_{P_{j+1}'}$ (possibly switching them).

It is easy to verify that all invariants continue to hold in this case. Let $\tilde{H}$ be the graph obtained by the union of the paths in $P_i' \cup Q'$, and define $\tilde{H}'$ similarly for $P_i''' \cup Q'$. Then $\tilde{H}'$ contains fewer edges than $\tilde{H}$, since the portion of the path $P_j$ between $s_1$ and $s_2$ belongs to $\tilde{H}$ but not to $\tilde{H}'$. 

56
We are now ready to complete the description of our algorithm. We start with \( P'_1 = P_1 \), and then iterate. In every iteration, we construct the graph \( \tilde{H} \), the sub-graph of \( G \) induced by \( P'_1 \cup Q' \). If \( \tilde{H} \) contains a bump or a cross, we apply the appropriate action. If the resulting linkage, \( L''_i \), has the property that \( H(S_i, L''_i) \) has fewer degree-2 vertices than \( H(S_i, L_i) \), then we terminate the algorithm and return \( L''_i \). Otherwise, we obtain a valid input to the next iteration, and moreover, the number of edges in the new graph \( \tilde{H} \) strictly decreases. Therefore, we are guaranteed that within \( O(|E(G[S_i])|) \) iterations, either the algorithm terminates with the desired linkage \( L''_i \), or the graph \( \tilde{H} \) contains no bump and no cross.

Consider the final graph \( \tilde{H} \). For each path \( Q \in Q' \), let \( v_Q \) be the first vertex of \( Q \) that belongs to \( V(P_1) \), and let \( u_Q \) be the last vertex of \( Q \) that belongs to \( V(P_1) \). Let \( \tilde{Q} \) be the sub-path of \( Q \) between \( v_Q \) and \( u_Q \). Delete from \( \tilde{H} \) all vertices of \( V(Q) \setminus V(\tilde{Q}) \) for all \( Q \in Q' \), and let \( \tilde{H}' \) denote the resulting graph. Let \( \tilde{Q}' \) be the set of the sub-paths \( \tilde{Q} \) for all \( Q \in Q' \). We need the following claim.

**Claim B.1** If \( \tilde{H} \) contains no cross and no bump, then \( \tilde{H}' \) is planar.

**Proof:** Consider some path \( \tilde{Q} \in \tilde{Q}' \). Delete from \( \tilde{Q} \) all edges that participate in the paths in \( P_1 \), and let \( \Sigma(\tilde{Q}) \) be the resulting set of sub-paths of \( \tilde{Q} \). While any path \( \sigma \in \Sigma(\tilde{Q}) \) contains a vertex \( v \in V(P_1) \) as an inner vertex, we replace \( \sigma \) with two sub-paths, where each subpath starts at one of the endpoints of \( \sigma \) and terminates at \( v \). Let \( \Sigma = \bigcup_{\tilde{Q} \in \tilde{Q}'} \Sigma(\tilde{Q}) \) be the resulting set of paths. Then for each path \( \sigma \in \Sigma \), both endpoints of \( \sigma \) belong to \( V(P_1) \), and the inner vertices are disjoint from \( V(P_1) \). Moreover, since the paths in \( P'_1 \) induce a 2-path in the corresponding graph \( H(S_i, L''_i) \), and since there are no bumps, the endpoints of each such path \( \sigma \) connect two consecutive paths in \( P'_1 \). Since no crosses are allowed, it is easy to see that the graph \( \tilde{H}' \) is planar.

Since \( \tilde{H}' \) is planar, it must contain a grid minor of size \( \Omega(V(\sqrt{h}) \times \Omega(\sqrt{h})) \). One way to see this is to observe that the treewidth of \( \tilde{H} \) is \( \Omega(\sqrt{h}) \), as it contains a node-well-linked set of \( \Omega(\sqrt{h}) \) vertices - a collection of \( z/2 \) endpoints of the paths in \( P_1 \), that belong to \( A_i \). We can then use known results [RST94], that show that a planar graph of treewidth \( g \) contains a grid minor of size \( \Omega(g \times \Omega(g)) \). Alternatively, we can find a grid minor of size \( (z/2 \times z/2) = (\Omega(\sqrt{h}) \times \Omega(\sqrt{h})) \) directly, by ensuring that for each pair of paths \( Q \in Q' P \in P'_1 \), the intersection \( Q \cap P \) is a path. This can be done by re-routing the paths of \( Q' \) using standard techniques.