On Vertex Sparsifiers with Steiner Nodes

Julia Chuzhoy[*]

March 16, 2012

Abstract

Given an undirected graph $G = (V, E)$ with edge capacities $c_e \geq 1$ for $e \in E$ and a subset $\mathcal{T}$ of k vertices called terminals, we say that a graph H is a quality-q cut sparsifier for G iff $\mathcal{T} \subseteq V(H)$, and for any partition (A, B) of $\mathcal{T}$, the values of the minimum cuts separating A and B in graphs G and H are within a factor q from each other. We say that H is a quality-q flow sparsifier for G iff $\mathcal{T} \subseteq V(H)$, and for any set D of demands over the terminals, the values of the minimum edge congestion incurred by fractionally routing the demands in D in graphs G and H are within a factor q from each other.

So far vertex sparsifiers have been studied in a restricted setting where the sparsifier H is not allowed to contain any non-terminal vertices, that is $V(H) = \mathcal{T}$. For this setting, efficient algorithms are known for constructing quality-$O(\log k / \log \log k)$ cut and flow vertex sparsifiers, as well as a lower bound of $\tilde{\Omega}(\sqrt{\log k})$ on the quality of any flow or cut sparsifier.

We study flow and cut sparsifiers in the more general setting where Steiner vertices are allowed, that is, we no longer require that $V(H) = \mathcal{T}$. We show algorithms to construct constant-quality cut sparsifiers of size $O(C^3)$ in time $\text{poly}(n) \cdot 2^C$, and constant-quality flow sparsifiers of size $C^{O(\log \log C)}$ in time $n^{O(\log C)} \cdot 2^C$, where C is the total capacity of the edges incident on the terminals.

1 Introduction

Suppose we are given an undirected graph $G = (V, E)$ with edge capacities $c_e \geq 1$ for $e \in E$, and a subset $\mathcal{T} \subseteq V$ of k vertices called terminals. Assume further that we are interested in routing traffic across G between the terminals in $\mathcal{T}$. While the size of the graph G may be very large, the specific structure of G is largely irrelevant to our task, except where it affects our ability to route flow between the terminals. A natural question is whether we can build a smaller graph $H = (V', E')$, with $\mathcal{T} \subseteq V'$, that approximately preserves the routing properties of graph G with respect to $\mathcal{T}$. In this case, we say that H is a *vertex sparsifier* for G. Two types of vertex sparsifiers have been studied so far: cut sparsifiers, which preserve the minimum cuts between any partition of the terminals, and flow sparsifiers, which preserve the minimum edge congestion required for routing any set D of demands over $\mathcal{T}$.

More formally, given any graph G with capacities $c_e \geq 1$ for the edges $e \in E$, a subset $\mathcal{T}$ of vertices called terminals, and a partition $(\mathcal{T}_A, \mathcal{T}_B)$ of the terminals, let $\text{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)$ denote the capacity of the minimum cut separating the vertices of $\mathcal{T}_A$ from the vertices of $\mathcal{T}_B$ in G. We

say that a graph $H = (V', E')$ is a quality-q *vertex cut sparsifier*, or just cut sparsifier, for graph G with terminal set $\mathcal{T}$, iff $\mathcal{T} \subseteq V'$, and for every partition $(\mathcal{T}_A, \mathcal{T}_B)$ of $\mathcal{T}$, $\mathrm{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B) \leq \mathrm{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B) \leq q \cdot \mathrm{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)$.

Given a graph G with capacities $c_e \geq 1$ for every edge $e \in E$, and a subset $\mathcal{T} \subseteq V$ of vertices called terminals, a set D of demands over the terminals specifies, for every unordered pair (t, t') of terminals, a demand $D(t, t')$. A flow F is a *routing* of the set D of demands, iff for every pair (t, t') of terminals, t and t' send $D(t, t')$ flow units to each other. The congestion of F is the maximum, over all edges $e \in E$, of $F(e)/c_e$, where $F(e)$ is the flow sent along e. Given a set D of demands over the set $\mathcal{T}$ of terminals, let $\eta(G, D)$ denote the minimum congestion required for routing the demands in D in graph G. We say that a graph H is a *flow sparsifier of quality q* for G, iff $\mathcal{T} \subseteq V(H)$, and for any set D of demands over the set $\mathcal{T}$ of terminals, $\eta(H, D) \leq \eta(G, D) \leq q \cdot \eta(H, D)$.

For a vertex sparsifier H, we say that the vertices in $V(H) \setminus \mathcal{T}$ are *Steiner vertices*. Vertex cut sparsifiers were first introduced by Moitra [Moi09], and later Leighton and Moitra [LM10] defined flow sparsifiers and showed that they generalize cut sparsifiers. The main motivation in both papers was designing improved approximation algorithms for graph partitioning and routing problems. Specifically, if the solution value of some combinatorial optimization problem only depends on the values of the minimum cuts separating terminal subsets, then given any approximation algorithm for the problem, we can first compute a cut sparsifier H for graph G, and then run this algorithm on H, thus obtaining an algorithm whose performance guarantee is independent of the size of G, and only depends on the size of the sparsifier H. Flow sparsifiers can be similarly used for combinatorial optimization problems whose solution value only depends on the congestion required for routing various demand sets over the terminals in G. The definitions of the cut and the flow sparsifiers of [Moi09, LM10] however required that the sparsifier H does not contain any Steiner vertices, that is, $V(H) = \mathcal{T}$.

Moitra [Moi09] showed that there exist cut sparsifiers of quality $O(\log k / \log \log k)$ even when no Steiner vertices are allowed, and Leighton and Moitra [LM10] proved the existence of quality-$O(\log k / \log \log k)$ flow sparsifiers for the same setting, and obtained an efficient algorithm to construct quality-$O(\log^2 k / \log \log k)$ flow and cut sparsifiers. Recently, Charikar et al. [CLLM10], Englert et al. [EGK$^+$10] and Makarychev and Makarychev [MM10] have shown efficient algorithms to construct quality-$O(\log k / \log \log k)$ flow and cut sparsifiers that do not contain Steiner vertices. On the negative side, Leighton and Moitra [LM10] have shown a lower bound of $\Omega(\log \log k)$ on the quality of flow sparsifiers when no Steiner vertices are allowed. This bound was later improved to $\Omega(\sqrt{\log k / \log \log k})$ by Makarychev and Makarychev [MM10]. Englert et al. [EGK$^+$10] have shown a lower bound of $\Omega(\sqrt{\log k} / \log \log k)$ on the quality of flow sparsifiers when no Steiner vertices are allowed, and all edge capacities of the sparsifier are bounded from below by a constant. As for cut vertex sparsifiers with no Steiner nodes, [CLLM10] and [MM10] have shown a lower bound of $\Omega(\log^{1/4} k)$, and the results of [MM10] together with the results of [FJS88] give a lower bound of $\Omega(\sqrt{\log k} / \log \log k)$ on their quality.

It is therefore natural to ask whether we can obtain better quality vertex sparsifiers by allowing Steiner vertices. In particular, an interesting question is: what is the smallest size $S(k)$, such that for any graph G with a set $\mathcal{T}$ of k terminals, there is a constant-quality cut or flow sparsifier of size at most $S(k)$. Notice that if our goal is to obtain better and faster approximation algorithms via graph sparsifiers, then the presence of Steiner nodes may actually lead to improved performance if we can construct better quality sparsifiers, while keeping the graph size sufficiently low.

For simplicity, we first consider a special case where all edge capacities in the input graph G

are unit, and every terminal in $\mathcal{T}$ has degree 1 in G. In this case we show that there exist constant-quality cut sparsifiers of size $O(k^3)$, and constant-quality flow sparsifiers of size $k^{O(\log \log k)}$. We also show algorithms to construct these sparsifiers in time $\text{poly}(n) \cdot 2^k$ for cut sparsifiers and in time $n^{O(\log k)} \cdot 2^k$ for flow sparsifiers. We then generalize these algorithms to arbitrary edge capacities. Let C be the total capacity of the edges incident on the terminals, assuming that for each edge $e \in E$, $c_e \geq 1$. We show that there exist constant-quality cut sparsifiers of size $O(C^3)$, and constant-quality flow sparsifiers of size $C^{O(\log \log C)}$, and show algorithms to construct such sparsifiers, with running time $\text{poly}(n) \cdot 2^C$ for cut sparsifiers and $n^{O(\log C)} \cdot 2^C$ for flow sparsifiers.

We say that a graph H is a *restricted sparsifier* for graph G, if H is a sparsifier that is associated with a collection $\mathcal{C}$ of disjoint subsets of non-terminal vertices, and H is obtained from G by contracting every cluster $S \in \mathcal{C}$ into a vertex. All sparsifiers that we construct are restricted sparsifiers. Interestingly, Charikar et al. [CLLM10] showed that when Steiner vertices are not allowed, the ratio of the quality of the best possible restricted flow sparsifier to the quality of an optimal flow sparsifier is super-constant. Moreover, Englert et al. [EGK$^+$10] have shown an $\Omega(\sqrt{\log k})$ lower bound on the quality of sparsifiers that do not contain Steiner vertices, and can be obtained from convex combinations of 0-extensions in graph G.

We note that our techniques are very different from the techniques of [MM10, CLLM10, EGK$^+$10], who exploited the connection between vertex sparsifiers and 0-extensions. Instead, we use well-linked decompositions and other techniques that are often employed in the context of graph routing.

Our Results and Techniques We start with a simple construction of cut sparsifiers with Steiner vertices, which is summarized in the following theorem.

Theorem 1 *Let $G = (V, E)$ be any n-vertex graph with capacities $c_e \geq 1$ on edges $e \in E$, and a set $\mathcal{T} \subseteq V$ of terminals. Let C denote the total capacity of the edges incident on the terminals, and let $0 < \epsilon \leq 1$ be any constant. Then there is a quality-$(3 + \epsilon)$ vertex cut sparsifier $H = (V', E')$ for G, with $|V'| = O(C^3)$. Moreover, graph H can be constructed in time $\text{poly}(n) \cdot 2^C$.*

For simplicity, we give an outline of the construction for the special case where all edge capacities are unit, and the degree of every terminal is 1. Our algorithm relies on the notion of well-linkedness, and on a new procedure to compute a well-linked decomposition. Given any subset S of vertices, let $\text{out}(S)$ denote the set of edges with exactly one endpoint in S. We say that S is α-well-linked, iff for any partition (A, B) of S, if we denote $\mathcal{T}_A = \text{out}(S) \cap \text{out}(A)$ and $\mathcal{T}_B = \text{out}(S) \cap \text{out}(B)$, then $|E(A, B)| \geq \alpha \cdot \min\{|\mathcal{T}_A|, |\mathcal{T}_B|\}$. Informally, we can set up an instance of the sparsest cut problem, on graph $G[S] \cup \text{out}(S)$, where the edges of $\text{out}(S)$ serve as terminals. Set S being α-well-linked is roughly equivalent to the value of sparsest cut in this new graph being at least α. The notion of well-linkedness[1] has been used extensively in graph routing e.g. in [Räc02, CKS04, CKS05, RZ10, And10], and one of the useful tools for designing algorithms for routing problems is *well-linked decomposition*: a procedure that, given any subset S of vertices with $|\text{out}(S)| = z$, produces a partition $\mathcal{W}$ of S into well-linked subsets. In all standard well-linked decompositions, we can ensure that $|\mathcal{W}|$ is small (less than z), while each set $X \in \mathcal{W}$ is guaranteed to be α-well-linked, where $\alpha = 1/\text{poly} \log z$. We show a different well-linked decomposition, that instead ensures that every set $X \in \mathcal{W}$ is 1/3-well-linked, and we can still bound the number of clusters in $\mathcal{W}$ by $O(z^3)$. An algorithm for constructing a cut sparsifier then simply computes a well-linked decomposition

---

[1] Our definition of well-linkedness is very similar to what was called bandwidth property in [Räc02], and cut well-linkedness in [CKS05], where we use the graph $G[S] \cup \text{out}(S)$, and the set of terminals is the edges of $\text{out}(S)$.

$\mathcal{W}$ of the set $V(G) \setminus \mathcal{T}$ of vertices, and contracts every cluster $X \in \mathcal{W}$. Since every cluster $X \in \mathcal{W}$ is 1/3-well-linked, it is easy to verify that we obtain a constant-quality cut sparsifier.

We now turn to the more challenging task of constructing flow sparsifiers. We again first consider a special case where all edge capacities are unit, and each terminal $t \in \mathcal{T}$ has exactly one edge incident to it in G. We show that for this special case, there is a flow sparsifier H of quality 68 and size $k^{O(\log \log k)}$, where $k = |\mathcal{T}|$. Recall that a sparsifier H is called a restricted sparsifier iff it is associated with a collection $\mathcal{C}$ of disjoint subsets of non-terminal vertices, and graph H is obtained from G by contracting each cluster $S \in \mathcal{C}$ into a vertex.

Theorem 2 *Let* $G = (V, E)$ *be any n-vertex (multi-)graph with unit edge capacities and a set $\mathcal{T} \subseteq V$ of k terminals. Assume further that each terminal in $\mathcal{T}$ has exactly one edge incident to it in G. Then there is an algorithm that finds, in time $n^{O(\log k)} \cdot 2^k$, a quality-q restricted vertex flow sparsifier H for G, with $|V(H)| = k^{O(\log \log k)}$ and $q = 68$.*

It is then fairly easy to obtain the following corollary that extends the results of Theorem 2 to general graphs.

Corollary 1 *Let $G = (V, E)$ be any n-vertex graph with edge capacities $c_e \geq 1$ for $e \in E$, and a set $\mathcal{T} \subseteq V$ of terminals. Let C denote the total capacity of all edges incident on the terminals, and let $0 < \epsilon < 1$ be any constant. Then there is an algorithm that finds, in time $n^{O(\log C)} \cdot 2^C$, a quality-q vertex flow sparsifier H for G, with $|V(H)| = C^{O(\log \log C)}$ and $q = 68 + \epsilon$.*

We now outline our algorithm for constructing flow sparsifiers for the special case where all edge capacities are unit, and the degree of every terminal is 1. Let us assume for simplicity that the set $R = V(G) \setminus \mathcal{T}$ of vertices is 1/3-well-linked (we perform a well-linked decomposition as a pre-processing step to ensure this). One of the central notions in our algorithm is that of good routers. We say that a subset $S \subseteq R$ of vertices is a good router iff it is 1/3-well-linked, and moreover, every pair of edges in $\text{out}(S)$ can simultaneously send $1/z$ flow units to each other with constant congestion inside S, where $z = |\text{out}(S)|$. We say that a graph H is a legal contracted graph for G iff there is a collection $\mathcal{C}$ of disjoint good routers in graph G, and H is obtained from G by contracting every cluster $S \in \mathcal{C}$. It is easy to verify that if H is a legal contracted graph, then it is a constant quality flow sparsifier, since contracting the good routers in $\mathcal{C}$ may only affect the congestion of any routing by a constant factor. Our goal is then to find a legal contracted graph whose size is small enough.

Notice that we have assumed that $R = V(G) \setminus \mathcal{T}$ is 1/3-well-linked. However, this is not sufficient to ensure that R is a good router, as the ratio between the minimum sparsest cut and the maximum concurrent flow, known as the flow-cut gap, can be as large as logarithmic in undirected graphs. To overcome this difficulty, we define several special structures that we call witnesses. If graph G contains such a witness, then we are guaranteed that R is a good router. For example, suppose that for some value $0 < \alpha < 1$, graph G contains $r = \log k / \alpha$ disjoint subsets $S_1, \ldots, S_r$ of non-terminal vertices, where for each $1 \leq j \leq r$, subset S_j is α-well-linked, and there is a set $\mathcal{P}_j$ of edge-disjoint paths in G, connecting every terminal in $\mathcal{T}$ to some edge in $\text{out}(S_j)$. For each $1 \leq j \leq r$, let $E_j \subseteq \text{out}(S_j)$ be the set of k edges where the paths of $\mathcal{P}_j$ terminate. Since the flow-cut gap in undirected graphs is bounded by $O(\log k)$, and the set S_j is α-well-linked, every pair of edges in E_j can simultaneously send $\frac{\alpha}{k \log k} = \frac{1}{rk}$ flow units to each other with constant congestion inside S_j. If graph G contains such a witness $\{S_1, \ldots, S_r\}$, it is easy to verify that R must be a good router, since we can send, for each $1 \leq j \leq r$, $1/r$ flow units along each path in $\mathcal{P}_j$,

so that each edge $e \in \text{out}(S_j)$ receives at most $1/r$ flow units, and then send $\frac{1}{rk}$ flow units between every pair of edges in E_j, with constant congestion inside S_j. In this way, every pair of terminals sends $1/k$ flow units to each other with constant congestion in G.

Our algorithm proceeds as follows. Throughout the algorithm, we maintain a legal contracted graph G' of G, where at the beginning $G' = G$. As long as the number of vertices in G' is large enough, we perform an iteration, whose output is either a witness for set R being a good router, or another legal contracted graph G'' that contains fewer vertices than G'. In the former case, we stop the algorithm and output a sparsifier H obtained from G by contracting the set R, and in the latter case we proceed to the next iteration. In fact, we can efficiently check whether R is a good router beforehand, by computing an appropriate multicommodity flow in $G[R]$, to ensure that the former case never happens. Once the size of the current graph G' becomes small enough, we output it as our final sparsifier.

Organization: We start with preliminaries and notation in Section 2. We construct cut sparsifiers in Section 3 and flow sparsifiers in Section 4.

2 Preliminaries and Notation

In all our results, we start with a special case where all edges in G have unit capacities, and then extend our results to the general setting. Therefore, all definitions and results presented in this section are for graphs with unit edge capacities.

General Notation For a graph $G = (V, E)$, and subsets $V' \subseteq V$, $E' \subseteq E$ of its vertices and edges respectively, we denote by $G[V']$, $G \setminus V'$, and $G \setminus E'$ the sub-graphs of G induced by V', $V \setminus V'$, and $E \setminus E'$, respectively. For any subset $S \subseteq V$ of vertices, we denote by $\text{out}_G(S) = E_G(S, V \setminus S)$ the subset of edges with one endpoint in S and the other endpoint in $V \setminus S$. When clear from context, we omit the subscript G. All logarithms are to the base of 2.

Let $\mathcal{P}$ be any collection of paths in graph G. We say that paths in $\mathcal{P}$ cause congestion η in G, iff for each edge $e \in E(G)$, the number of paths in $\mathcal{P}$ containing e is at most η.

Given a graph $G = (V, E)$, and a subset $\mathcal{T} \subseteq V$ of vertices called terminals, a set D of demands is a function $D : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^+$, that specifies, for each pair $t, t' \in \mathcal{T}$ of terminals, a demand $D(t, t')$. For simplicity, we assume that the pairs t, t' of terminals are unordered, that is $D(t, t') = D(t', t)$ for all $t, t' \in \mathcal{T}$. We say that the set D of demands is γ-restricted, iff for each terminal $t \in \mathcal{T}$, the total demand $\sum_{t' \in \mathcal{T}} D(t, t') \leq \gamma$.

Given any set D of demands, a *routing* of D is a flow F, where for each unordered pair $t, t' \in \mathcal{T}$, the amount of flow sent from t to t' (or from t' to t) is $D(t, t')$. The *congestion* of the flow is the maximum, over all edges $e \in E$, of $F(e)$ — the amount of flow sent via the edge e.

Given any two subsets V_1, V_2 of vertices, we denote by $F : V_1 \rightsquigarrow_\eta V_2$ a flow that causes congestion at most η in G, where each vertex in V_1 sends one flow unit, and each flow-path starts at a vertex of V_1 and terminates at a vertex of V_2. We denote by $F : V_1 \overset{1:1}{\rightsquigarrow}_\eta V_2$ a flow with the above properties, where additionally each vertex in V_2 receives at most one flow unit. Similarly, we denote by $\mathcal{P} : V_1 \rightsquigarrow_\eta V_2$ a collection of paths $\mathcal{P} = \{P_v \mid v \in V_1\}$ in graph G, where each path P_v originates at v and terminates at some vertex of V_2, and the paths in $\mathcal{P}$ cause congestion at most η. We denote $\mathcal{P} : V_1 \overset{1:1}{\rightsquigarrow}_\eta V_2$ if additionally each vertex of V_2 serves as an endpoint of at most one path in $\mathcal{P}$. Similarly, we define flows and paths between subsets of edges. For example, given two collections E_1, E_2 of edges of G, we denote by $F : E_1 \rightsquigarrow_\eta E_2$ a flow that causes congestion at

most η in G, where each flow-path has an edge in E_1 as its first edge, and an edge in E_2 as its last edge, and moreover each edge in E_1 sends one flow unit. (Notice that it is then guaranteed that each edge in E_2 receives at most η flow units due to the bound on congestion). If additionally each edge in E_2 receives at most one flow unit, we denote this by $F : E_1 \overset{1:1}{\rightsquigarrow}_\eta E_2$. Collections of paths connecting subsets of edges to each other are defined similarly. We will often be interested in a scenario where we are given a subset $S \subseteq V(G)$ of vertices, and $E_1, E_2 \subseteq \mathrm{out}(S)$. In this case, we say that a flow $F : E_1 \rightsquigarrow_\eta E_2$ is *contained* in S, iff for each flow-path P in F, all edges of P belong to $G[S]$, except for the first and the last edges that belong to $\mathrm{out}(S)$. Similarly, we say that a set $\mathcal{P} : E_1 \rightsquigarrow_\eta E_2$ of paths is contained in S, iff all inner edges on paths in $\mathcal{P}$ belong to $G[S]$.

Sparsest Cut and the Flow-Cut Gap Suppose we are given a graph $G = (V, E)$, with non-negative weights w_v on vertices $v \in V$, and a subset $\mathcal{T} \subseteq V$ of k terminals, such that for all $v \notin \mathcal{T}$, $w_v = 0$. Given any partition (A, B) of V, the *sparsity* of the cut (A, B) is $\frac{|E(A,B)|}{\min\{W(A),W(B)\}}$, where $W(A) = \sum_{v \in A} w_v$ and $W(B) = \sum_{v \in B} w_v$. In the sparsest cut problem, the input is a graph G with non-negative weights on vertices, and the goal is 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 by $\mathcal{A}_{\mathrm{ARV}}$ this algorithm and by $\alpha_{\mathrm{ARV}}(k) = O(\sqrt{\log k})$ its approximation factor. We will usually work with a special case of the sparsest cut problem, where for each $t \in \mathcal{T}$, $w_t = 1$. We denote such an instance by $(G, \mathcal{T})$.

The dual of the sparsest cut problem is the maximum concurrent flow problem, where the goal is to find the maximum possible value λ, such that each pair (t, t') of terminals can simultaneously send λ/k flow units to each other with unit congestion (we assume that in the sparsest cut problem instance the weights $w_t = 1$ for all $t \in \mathcal{T}$). The flow-cut gap is the maximum possible ratio, in any graph, between the value of the minimum sparsest cut and the value λ of the maximum concurrent flow. The flow-cut gap in undirected graphs, that we denote by $\beta_{\mathrm{FCG}}(k)$ throughout the paper, is $\Theta(\log k)$ [LR99, GVY95, LLR94, AR98]. In particular, if the value of the sparsest cut in graph G is α, then every pair of terminals can send at least $\frac{\alpha}{k\beta_{\mathrm{FCG}}(k)}$ flow units to each other simultaneously with no congestion. It is also easy to see that any 1-restricted set D of demands on set $\mathcal{T}$ of terminals can be routed with congestion at most $2\beta_{\mathrm{FCG}}(k)/\alpha$. In order to find this routing, let F be the flow where every pair of terminals sends $\frac{\alpha}{k\beta_{\mathrm{FCG}}(k)}$ flow units to each other with no congestion, and let F' be the same flow scaled up by factor $\beta_{\mathrm{FCG}}(k)/\alpha$, so the flow in F' causes congestion at most $\beta_{\mathrm{FCG}}(k)/\alpha$, and every pair of terminals sends $1/k$ flow units to each other. For each pair (t, t') of terminals, vertex t sends $D(t, t')/k$ flow units to each terminal in $\mathcal{T}$ using the flow F' (scaled by factor $D(t, t')$), and vertex t' collects $D(t, t')/k$ flow units from each terminal in $\mathcal{T}$. It is easy to verify that, since the set D of demands is 1-restricted, the total congestion of this flow is bounded by $2\beta_{\mathrm{FCG}}(k)/\alpha$.

Well-Linked Decompositions

Definition 1 *Given a graph G, a subset S of its vertices, and a parameter $\alpha > 0$, we say that S is α-well-linked, iff for any partition (A, B) of S, if we denote by $T_A = \mathrm{out}(A) \cap \mathrm{out}(S)$, and by $T_B = \mathrm{out}(B) \cap \mathrm{out}(S)$, then $|E(A, B)| \geq \alpha \cdot \min\{|T_A|, |T_B|\}$.*

Given a subset S of vertices of G, we define a graph G_S associated with S, and a corresponding instance $(G_S, \mathcal{T}'_S)$ of the sparsest cut problem, that we use throughout the paper. We start by

sub-dividing every edge $e \in \text{out}_G(S)$ by a vertex t_e, and let $\mathcal{T}'_S = \{t_e \mid e \in \text{out}_G(S)\}$ be the set of these new vertices. We then let G_S be the sub-graph of the resulting graph, induced by $S \cup \mathcal{T}'_S$. Notice that set S is α-well-linked in G iff the value of the sparsest cut in instance $(G_S, \mathcal{T}'_S)$ is at least α (for $\alpha < 1$). In particular, if S is α-well-linked, and $|\text{out}(S)| = z$, then we have the following two properties:

P1. Any set D of 1-restricted demands on the edges of $\text{out}(S)$ can be routed inside S with congestion at most $2\beta_{\text{FCG}}(z)/\alpha$.

P2. For any two subsets $E_1, E_2 \subseteq \text{out}(S)$, where $|E_1| = |E_2|$, there is a collection $\mathcal{P} : E_1 \overset{1:1}{\rightsquigarrow}_{\lceil 1/\alpha \rceil} E_2$ of paths contained in S.

In order to obtain the latter property, we set up a single-source single-sink max-flow instance in graph G_S, where the edges of E_1 serve as the source and the edges of E_2 serve as the sink. The existence of the flow $F : E_1 \overset{1:1}{\rightsquigarrow}_{1/\alpha} E_2$ follows from the max-flow/min-cut theorem, and the existence of the set $\mathcal{P} : E_1 \overset{1:1}{\rightsquigarrow}_{\lceil 1/\alpha \rceil} E_2$ of paths follows from the integrality of flow.

A well-linked decomposition of an arbitrary subset S of vertices, is a partition of S into a collection of well-linked subsets. We use two different types of well-linked decomposition, that give slightly different guarantees. We start with a standard decomposition, that we refer to as the *weak well-linked decomposition*, and it is similar to the one used in [CKS05, Räc02]. The proof of the next theorem appears in the Appendix.

Theorem 3 (Weak well-linked decomposition) *Given any graph $G = (V, E)$, and any subset $S \subseteq V$ of vertices with $|\text{out}(S)| = z$, there is an efficient algorithm, that finds a partition $\mathcal{W}$ of S, such that for each set $R \in \mathcal{W}$, $|\text{out}(R)| \leq |\text{out}(S)|$, R is $\alpha_W(z)$-well-linked for $\alpha_W(z) = \Omega\left(\frac{1}{\log^{3/2} z}\right)$, and $\sum_{R \in \mathcal{W}} |\text{out}(R)| \leq 1.2 |\text{out}(S)|$.*

The next theorem gives what we call a *strong well-linked decomposition*. This decomposition gives a better guarantee for the well-linkedness of the resulting sets in the partition. The drawback is that the running time of the algorithm is exponential in $|\text{out}(S)|$, and the number of edges adjacent to the subsets in the partition is higher. The proof of the next theorem appears in the Appendix.

Theorem 4 (Strong well-linked decomposition) *Given any n-vertex graph $G = (V, E)$, and any subset $S \subseteq V$ of its vertices, where $G[S]$ is connected and $|\text{out}(S)| = z$, there is an algorithm running in time $2^z \cdot \text{poly}(n)$, that finds a partition $\mathcal{S}$ of S, such that:*

- *For each $R \in \mathcal{S}$, $|\text{out}(R)| \leq |\text{out}(S)|$, and R is $1/3$-well-linked;*

- *$\sum_{R \in \mathcal{S}} |\text{out}(R)| = O(z^3)$; and*

- *For each $i : 1 \leq i \leq 2\lfloor \log z \rfloor$, if $\mathcal{S}_i \subseteq \mathcal{S}$ denotes the collection of subsets $R \in \mathcal{S}$ with $z/2^i < |\text{out}(R)| \leq z/2^{i-1}$, then $|\mathcal{S}_i| \leq 2^{3i+3}$ for all i.*

We will sometimes use the notion of well-linkedness in a slightly different setting. Suppose we are given a graph $G = (V, E)$, and a subset $\mathcal{T} \subseteq V$ of vertices called terminals. We say that G is α-well-linked with respect to $\mathcal{T}$, iff for any partition (A, B) of V, if we denote $T_A = \mathcal{T} \cap A$ and

7

$T_B = \mathcal{T} \cap B$, then $|E(A,B)| \geq \alpha \cdot \min\{|T_A|, |T_B|\}$. A convenient way of viewing this consistently with the previous definition of well-linkedness is to augment the graph G, by adding an edge connecting each terminal $t \in \mathcal{T}$ to a new vertex v_t. Saying that G is α-well-linked for $\mathcal{T}$ is then equivalent to saying that the subset V of vertices of the new graph is α-well-linked.

3 Cut Sparsifiers

In this section we prove Theorem 1. We fist consider a simpler special case where all edge capacities are unit (but parallel edges are allowed), in the following theorem.

Theorem 5 *Let $G = (V, E)$ be any n-vertex (multi-)graph with unit edge capacities, and a set $\mathcal{T} \subseteq V$ of terminals. Let $k = \sum_{t \in \mathcal{T}} d_t$ be the sum of degrees of all terminals. Then there is a quality-3 vertex cut sparsifier $H = (V', E')$ for G, with $|V'| = O(k^3)$. Moreover, graph H can be constructed in time $\mathrm{poly}(n) \cdot 2^k$.*

Proof: We assume w.l.o.g. that G is a connected graph: otherwise, we construct a sparsifier for each of its connected components separately. Let $S = V \setminus \mathcal{T}$. Notice that $|\mathrm{out}(S)| = k$. In order to construct the sparsifier H, we compute a strong well-linked decomposition $\mathcal{S}$ of S, given by Theorem 4. Recall that the decomposition can be found in time $2^k \cdot \mathrm{poly}(n)$, and $|\mathcal{S}| = O(k^3)$. We now contract each set $R \in \mathcal{S}$ into a single super-node v_R. The resulting graph is the sparsifier H. Notice that H is an unweighted multi-graph, and $|V(H)| = O(k^3)$.

Assume that we are given any partition $(\mathcal{T}_A, \mathcal{T}_B)$ of the set $\mathcal{T}$ of terminals. It is easy to see that $\mathrm{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B) \leq \mathrm{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B)$: let (X', Y') be the minimum cut, separating $\mathcal{T}_A$ from $\mathcal{T}_B$ in graph H. Since H is obtained from G by contracting some subsets of its vertices, the cut (X', Y') naturally induces a cut (X, Y) separating $\mathcal{T}_A$ from $\mathcal{T}_B$ in G: for each cluster $R \in \mathcal{S}$, if $v_R \in X'$, then we add all vertices of R to X, and otherwise we add them to Y. The value of the cut, $|E_G(X, Y)| = |E_H(X', Y')|$, and so $\mathrm{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B) \leq \mathrm{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B)$.

We now prove that $\mathrm{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B) \leq 3\,\mathrm{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)$. Let (X, Y) be the minimum cut separating $\mathcal{T}_A$ from $\mathcal{T}_B$ in G. We define a cut (X', Y'), separating $\mathcal{T}_A$ from $\mathcal{T}_B$ in H, with $|E_H(X', Y')| \leq 3|E_G(X, Y)|$, as follows. we start with the cut (X, Y) in graph G, and we gradually change this cut, so that eventually, for each set $R \in \mathcal{S}$, all vertices of R are completely contained in either X or in Y. The resulting partition will then naturally define the cut (X', Y') in graph H.

We process the sets $R \in \mathcal{S}$ one-by-one. Let R be any such set. Partition the edges of $\mathrm{out}(R)$ into four subsets: E_X, E_Y, E_{XY}, E_{YX}, as follows. Let $e = (u, v) \in \mathrm{out}(R)$, where $u \in R$, $v \notin R$. If both u and v belong to X, then e is added to E_X. If both vertices belong to Y, then e is added to E_Y. If u belongs to X and v to Y, then e is added to E_{XY}. Otherwise, it is added to E_{YX} (see Figure 1). Let $E'_R = E_G(R \cap X, R \cap Y)$. If $|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|$, then we move all vertices of R to Y; otherwise we move them to X.

Assume w.l.o.g. that $|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|$, and so we have moved the vertices of R to Y. The only new edges that we have added to the cut are the edges of E_X. On the other hand, the edges of E'_R, that belonged to the cut before the current iteration, do not belong to the cut anymore. We charge the edges of E'_R for the edges of E_X. Since set R is 1/3-well-linked, $|E'_R| \geq |E_X|/3$ must hold, and so the charge to each edge of E'_R is at most 3. Moreover, since the edges of E'_R are the inner edges of the set R (that is, both endpoints of each such edge belong to R), we will never charge these edges again. Therefore, if $(\tilde{X}, \tilde{Y})$ denotes the final cut, after all clusters $R \in \mathcal{S}$ have been processed, then $|E_G(\tilde{X}, \tilde{Y})| \leq 3|E_G(X, Y)|$. Finally, the cut $(\tilde{X}, \tilde{Y})$ in
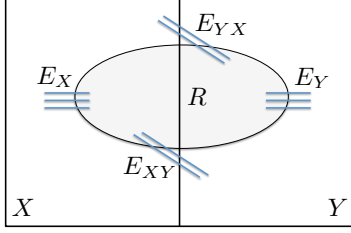
Figure 1: Illustration for Theorem 5

graph G naturally defines a cut (X', Y') in graph H: for each cluster $R \in \mathcal{S}$, if $R \subseteq \tilde{X}$, then we add v_R to X'; otherwise we add it to Y'. Clearly, $|E_H(X', Y')| = |E_G(\tilde{X}, \tilde{Y})| \le 3|E_G(X, Y)|$. We conclude that $\operatorname{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B) \le 3 \operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)$. □

We now complete the proof of Theorem 1. Suppose we are given a graph G with arbitrary edge capacities $c_e \ge 1$. For notational convenience, we denote the input parameter ϵ by ϵ', and we set $\epsilon = \epsilon'/3$. We perform the following transformation in graph G. Let C be the sum of the capacities of all edges incident on the terminals. For each edge $e \in E$, if the capacity $c_e > C$, then we set it to be C. Notice that this does not change the values $\operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)$ for any partition $(\mathcal{T}_A, \mathcal{T}_B)$ of the set $\mathcal{T}$ of the terminals, since $\operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B) \le C$ always holds. Finally, we replace each edge $e \in E$ with $\lceil c_e/\epsilon \rceil$ parallel unit-capacity edges. Let G' be the resulting graph. We now apply Theorem 5 to graph G', to obtain a sparsifier H' of size $O((C/\epsilon)^3) = O(C^3)$. We obtain a sparsifier H for graph G, by setting the capacity of every edge in H' to ϵ. We now show that H is a quality-$(3 + 3\epsilon) = (3 + \epsilon')$-sparsifier for G.

Notice that for each partition $(\mathcal{T}_A, \mathcal{T}_B)$ of $\mathcal{T}$, $\frac{\operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)}{\epsilon} \le \operatorname{MinCut}_{G'}(\mathcal{T}_A, \mathcal{T}_B) \le \operatorname{MinCut}_{H'}(\mathcal{T}_A, \mathcal{T}_B) = \frac{\operatorname{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B)}{\epsilon}$, and so $\operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B) \le \operatorname{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B)$. On the other hand, $\operatorname{MinCut}_{G'}(\mathcal{T}_A, \mathcal{T}_B) \le \frac{\operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)}{\epsilon}(1 + \epsilon)$, since all original edge capacities $c_e \ge 1$, and so $\lceil \frac{c_e}{\epsilon} \rceil \le \frac{c_e}{\epsilon}(1 + \epsilon)$. Therefore, $\operatorname{MinCut}_H(\mathcal{T}_A, \mathcal{T}_B) = \epsilon \cdot \operatorname{MinCut}_{H'}(\mathcal{T}_A, \mathcal{T}_B) \le 3\epsilon \operatorname{MinCut}_{G'}(\mathcal{T}_A, \mathcal{T}_B) \le 3(1 + \epsilon) \operatorname{MinCut}_G(\mathcal{T}_A, \mathcal{T}_B)$.

4 Flow Sparsifiers

In this section we prove Theorem 2 and Corollary 1. We start with the following definition.

Definition 2 *Let $S \subseteq V \setminus \mathcal{T}$ be any subset of non-terminal vertices, and let $|\operatorname{out}(S)| = z$. We say that S is a* good router *iff S is $1/3$-well-linked, and every pair $(e, e') \in \operatorname{out}(S)$ of edges can simultaneously send $1/z$ flow units to each other inside S, with congestion at most $\eta^* = 34$.*

Notice that we can efficiently check whether S is a good router by computing an appropriate multicommodity flow in the graph G_S. Notice also that if S is a good router, then any 1-restricted set D of demands on the edges of $\operatorname{out}(S)$ can be routed with congestion at most $2\eta^*$ inside S. Indeed, let F be the flow, where each pair $(e, e') \in \operatorname{out}(S)$ of edges sends $1/z$ flow units to each other with congestion at most η^* inside S. In order to route the set D of demands, consider any pair $(e, e') \in \operatorname{out}(S)$ of edges. Edge e sends $D(e, e')/z$ flow units to each edge $e'' \in \operatorname{out}(S)$, using the flow F (scaled by factor $D(e, e')$), while edge e' collects $D(e, e')/z$ flow units from each edge $e'' \in \operatorname{out}(S)$, using the flow F. In the end, we have $D(e, e')$ flow units sent from e to e', and since the set D of demands is 1-restricted, the total congestion of this routing is bounded by $2\eta^*$.

9

Definition 3 *We say that a graph G' is a* legal contracted graph *for G iff there is a collection $\mathcal{C}$ of disjoint good routers, where the clusters $S \in \mathcal{C}$ do not contain any terminals, and G' is obtained from G by contracting every cluster $S \in \mathcal{C}$ into a super-node v_S. (We remove self-loops, but leave parallel edges).*

It is easy to see that if G' is a legal contracted graph for G, then it is a quality-$2\eta^*$ flow sparsifier, as the next claim shows.

Claim 1 *If G' is a legal contracted graph for G, then it is a quality-$2\eta^*$ restricted flow sparsifier.*

Proof: Given any set D of demands on the terminals in $\mathcal{T}$, it is immediate to see that $\eta(G', D) \leq \eta(G, D)$, since G' is obtained from G by contracting some vertex subsets into super-nodes.

Assume now that we are given some set D of demands on $\mathcal{T}$, and $\eta(G', D) = \eta$. For simplicity, we scale the demands in D down by the factor of η, to obtain a new set D' of demands with $\eta(G', D') = 1$. It is now enough to show that we can route the demands in D' in graph G with congestion at most $2\eta^*$. Let F be the routing of D' in G' with congestion 1. For each cluster $S \in \mathcal{C}$, for each pair $(e, e') \in \text{out}(S)$ of edges, let $D_S(e, e')$ be the total amount of flow in F sent on flow-paths that enter v_S through edge e, and leave it through edge e'. We have thus obtained a set D_S of 1-restricted demands on the edges of $\text{out}(S)$. Since S is a good router, these demands can be routed inside S with congestion at most $2\eta^*$. Let F_S denote this routing. In order to obtain the final routing F' of the set D' of demands in G, we start with the flow F, and we augment it with the routings F_S that we have computed in each cluster $S \in \mathcal{C}$. Therefore, $\eta(G, D') \leq 2\eta^*$. It is immediate to see that G' is a restricted sparsifier for G, from the definition of a legal contracted graph. $\qquad \square$

Most of this section is devoted to proving the following theorem, which gives a construction of a flow sparsifier for the special case where the set $V \setminus \mathcal{T}$ is 1/3-well-linked.

Theorem 6 *Assume that we are given any (multi-)graph $G = (V, E)$, with unit edge capacities and a subset $\mathcal{T} \subseteq V$ of k terminals, where every vertex in $\mathcal{T}$ has degree 1. Let $R = V \setminus \mathcal{T}$, and assume further that R is 1/3-well-linked. Then there is an algorithm that finds, in time $2^k \cdot n^{O(\log k)}$, a restricted flow sparsifier H of quality $q = 2\eta^*$ for $(G, \mathcal{T})$, such that $|V(H)| = k^{O(\log \log k)}$, and H is a legal contracted graph for G.*

We defer the proof of Theorem 6 to Section 4.1, and complete the proof of Theorem 2 here. We assume w.l.o.g that G is a connected graph: otherwise, we compute a sparsifier for each of its connected components separately. Our first step is to compute a strong well-linked decomposition $\mathcal{S}$ of the set $V \setminus \mathcal{T}$ of vertices, given by Theorem 4. Recall that each set $X \in \mathcal{S}$ is 1/3-well-linked, $|\mathcal{S}| = O(k^3)$, and the decomposition can be computed in time $2^k \cdot \text{poly}(n)$. For each edge e in set $\bigcup_{X \in \mathcal{S}} \text{out}(X)$, we sub-divide e by a new vertex v_e, and we let G' denote the resulting graph. For each cluster $X \in \mathcal{S}$, let $\mathcal{T}_X = \{v_e \mid e \in \text{out}_G(X)\}$, and let $G_X = G'[X \cup \mathcal{T}_X]$. Notice that $|\mathcal{T}_X| \leq k$, and all vertices in $\mathcal{T}_X$ have degree 1 in G_X. For each cluster $X \in \mathcal{S}$, we use Theorem 6 on graph G_X and the set $\mathcal{T}_X$ of terminals, to find a restricted flow sparsifier H_X. Let $\mathcal{C}_X$ be the corresponding collection of disjoint subsets of $V(G_X) \setminus \mathcal{T}_X$, such that H_X is obtained from G_X by contracting every cluster in $\mathcal{C}_X$. Let $\mathcal{C} = \bigcup_{X \in \mathcal{S}} \mathcal{C}_X$. We obtain our final sparsifier H by contracting every cluster $S \in \mathcal{C}$ into a super-node v_S. Notice that since, for each cluster $X \in \mathcal{S}$, graph H_X is a legal contracted graph for G_X, each cluster $S \in \mathcal{C}$ is a good router, and so H is a legal contracted

graph for G. From Claim 1, H is a quality-$(2\eta^*)$ restricted sparsifier for G. It is easy to see that the running time of the algorithm is $2^k \cdot n^{O(\log k)}$. It now only remains to bound $|V(H)|$.

Recall that $|\mathcal{S}| \leq O(k^3)$, and for each $X \in \mathcal{S}$, $|\operatorname{out}(X)| \leq |\mathcal{T}| = k$. Therefore, $|V(H_X)| = k^{O(\log \log k)}$, and $|V(H)| = O(k^3) \cdot k^{O(\log \log k)} = k^{O(\log \log k)}$. This completes the proof of Theorem 2. The proof of Corollary 1 follows from Theorem 2 using standard techniques, and it appears in Section C of the Appendix. We now focus on the proof of Theorem 6, which is the main technical contribution of this section.

4.1 Proof of Theorem 6

We prove the theorem by induction on the value of k. Throughout the proof, we use two parameters: $r = O(\log^3 k)$, and $k^* = 2kr \log r = k \operatorname{poly} \log k$. We set the value r to be a large enough integer, so that the following inequality holds:

$$r > 24\beta_{\mathrm{FCG}}(k^*)/\alpha_W(k^*) \tag{1}$$

Notice that $\beta_{\mathrm{FCG}}(k^*)/\alpha_W(k^*) = O(\log^{5/2}(2kr \log r)) = O(\log^3 k) + O(\log^3(r \log r))$, so $r = O(\log^3 k)$ is sufficient.

Next, we define a function $F : \mathbb{R}^+ \to \mathbb{R}^+$, where $F(k')$ will roughly serve as an upper bound on the size of the sparsifier for any graph G with k' terminals. Function F is defined recursively, as follows. For $k' \leq 4$, $F(k') = 1$. If $k' > 4$ is an integral power of 2, then $F(k') = 2^{16} \cdot r^3 \log r \cdot F(k'/2) = O(\log^9 k \log \log k) \cdot F(k'/2)$. Otherwise, $F(k') = F(k'')$, where k'' is the smallest integral power of 2 with $k'' \geq k'$. Notice that for any integer $k' > 4$, $F(k'/2) = F(\lceil k'/2 \rceil)$, and we will sometimes use these values interchangeably.

Notice that for all values k', $F(k') = (\log k')^{O(\log k')}$, so $F(k) = k^{O(\log \log k)}$ as required. From now on, we focus on proving that if $G = (V, E)$ is a graph as in the theorem statement with k terminals, then we can find, in time $n^{O(\log k)} \cdot 2^k$, a restricted quality-$(2\eta^*)$ sparsifier H for G, such that $|V(H) \setminus \mathcal{T}| \leq F(k)$, and H is a legal contracted graph for G.

The proof is by induction on the values of k. If $k \leq 4$, then the set $R = V \setminus \mathcal{T}$ is a good router, so we can let $\mathcal{C} = \{R\}$, and return the corresponding contracted graph H as our sparsifier, so $|V(H) \setminus \mathcal{T}| = 1$. Assume now that the claim holds for values $k' < k$, and we now prove it for k.

Notice that if the set $R = V \setminus \mathcal{T}$ of vertices is a good router, then we can set $\mathcal{C} = \{R\}$, and output a sparsifier H, obtained from G, after we contract the cluster R into a super-node v_R. Therefore, we can assume from now on that R is not a good router. The main idea of the algorithm is as follows. Throughout the algorithm, we maintain a collection $\mathcal{C}$ of disjoint good routers in graph G and the corresponding legal contracted graph G'. At the beginning, $\mathcal{C} = \emptyset$, and $G' = G$. While the number of vertices in $V(G') \setminus \mathcal{T}$ is greater than $F(k)$, we perform an iteration, in which we obtain a new collection $\mathcal{C}'$ of disjoint good routers, such that the corresponding graph G'' contains strictly fewer vertices than G'. Once the number of vertices in $V(G') \setminus \mathcal{T}$ falls below $F(k)$, we stop and output G' as our sparsifier.

Notice that if G' is a legal contracted graph for G, then each edge of G' corresponds to some edge of G. We do not distinguish between these edges. For example, if $S \subseteq V(G')$ is any subset of vertices, and $S' \subseteq V(G)$ is obtained from S by replacing each super-node $v_C \in S$ by the vertices of C, then we view $\operatorname{out}_{G'}(S) = \operatorname{out}_G(S')$. We need the following definition.

Definition 4 *Let G' be the current legal contracted graph, and let $S \subseteq V(G') \setminus \mathcal{T}$ be any subset of non-terminal vertices, such that $G'[S]$ is connected. We say that S is a* contractible set *iff* $|\operatorname{out}(S)| \leq \lceil k/2 \rceil$, *and* $|S| > 128F(|\operatorname{out}_{G'}(S)|)$.

11

Let G' be the current contracted graph, and let $\mathcal{C}$ be the corresponding collection of good routers. Suppose we can find a contractible set S of vertices in the current graph G', with $|\operatorname{out}(S)| = k'$. We show that in this case we can compute a smaller legal contracted graph G''. We denote this procedure by $\mathsf{Contract}(G', S)$. Procedure $\mathsf{Contract}(G', S)$ is executed as follows. Let $\mathcal{C}_S \subseteq \mathcal{C}$ contain all clusters C with $v_C \in S$, and let S' be the subset of vertices of the original graph G obtained from S by replacing each super-node $v_C \in \mathcal{C}_S$ with the vertices of C. Clearly, $|S'| > 128F(k')$ still holds, $G[S']$ is a connected graph, and $|\operatorname{out}_G(S')| = k'$. Let $\mathcal{S}$ be the strong well-linked decomposition of S' given by Theorem 4. We now process the clusters in $\mathcal{S}$ one by one. Consider some cluster $Z \in \mathcal{S}$. We construct a new graph G_Z from graph G, by first sub-dividing every edge $e \in \operatorname{out}_G(Z)$ by a vertex v_e, setting $\mathcal{T}_Z = \{v_e \mid e \in \operatorname{out}_G(Z)\}$, and we let G_Z be the sub-graph of the resulting graph induced by $Z \cup \mathcal{T}_Z$. Let $k_Z = |\mathcal{T}_Z| = |\operatorname{out}_G(Z)|$, and observe that $k_Z \leq k' \leq \lceil k/2 \rceil < k$. Recall that G_Z is 1/3-well-linked for $\mathcal{T}_Z$, so by the induction hypothesis, we can find a sparsifier H_Z for $(G_Z, \mathcal{T}_Z)$, with $|V(H_Z) \setminus \mathcal{T}_Z| \leq F(k_Z)$. Let $\mathcal{C}_Z$ be the collection of the good routers corresponding to H_Z. Recall that each cluster $C \in \mathcal{C}_Z$ only contains vertices of Z. Let $\mathcal{C}' = (\mathcal{C} \setminus \mathcal{C}_S) \cup \left(\bigcup_{Z \in \mathcal{S}} \mathcal{C}_Z \right)$ be the new collection of good routers in graph G, and let G'' be the contracted graph corresponding to $\mathcal{C}'$. Graph G'' is the output of procedure $\mathsf{Contract}(G', S)$. In the next claim we show that $|V(G'')| < |V(G')|$.

Claim 2 *Let G'' be the output of Procedure $\mathsf{Contract}(G', S)$. Then $|V(G'')| < |V(G')|$.*

Proof: From the definition of G'',

$$
\begin{aligned}
V(G'') &= |V(G')| - |S| + \sum_{Z \in \mathcal{S}} |H_Z \setminus \mathcal{T}_Z| \\
&\leq |V(G')| - |S| + \sum_{Z \in \mathcal{S}} F(k_Z).
\end{aligned}
$$

Let $k' = |\operatorname{out}_{G'}(S)|$, and let k'' be the smallest power of 2, such that $k'' \geq k'$. Recall that $|S| > 128F(k') = 128F(k'')$, so in order to show that $|V(G'')| < |V(G')|$, it is enough to show that $\sum_{Z \in \mathcal{S}} F(k_Z) \leq 128F(k'')$. For each $i : 1 \leq i \leq \log k'' + 1$, let $\mathcal{S}_i \subseteq \mathcal{S}$ be the collection of subsets $Z \in \mathcal{S}$ with $k''/2^i < k_Z \leq k''/2^{i-1}$. Then from Theorem 4, $|\mathcal{S}_i| \leq 2^{3i+3}$ for all i. Therefore,

$$
\begin{aligned}
\sum_{Z \in \mathcal{S}} F(k_Z) &\leq \sum_{i=1}^{\log k'' + 1} |\mathcal{S}_i| \cdot F(k''/2^{i-1}) \\
&\leq \sum_{i=1}^{\log k'' + 1} 2^{3i+3} \cdot F(k''/2^{i-1})
\end{aligned}
$$

Let $T(i) = 2^{3i-3} F(k''/2^{i-1})$. Then

$$
T(i) = 8 \cdot 2^{3i-6} F(k''/2^{i-1}) < \frac{1}{2} \cdot 2^{3i-6} F(k''/2^{i-2}) = \frac{1}{2} T(i-1).
$$

Therefore, values $T(i)$ form a geometrically decreasing sequence, and $\sum_{i=1}^{\log k'' + 1} 2^{3i+3} \cdot F(k''/2^{i-1}) < 2^7 \cdot T(1) = 2^7 F(k'')$. $\qquad \square$

We now proceed to define two structures, that we call a type-1 and a type-2 witnesses. We show that if G' is a legal contracted graph for G, and G' contains either a type-1 or a type-2 witness, then $R = V(G) \setminus \mathcal{T}$ must be a good router. Finally, we show an algorithm, that, given a legal contracted graph G' with $|V(G') \setminus \mathcal{T}| > F(k)$, either finds a contractible subset $S \subseteq V(G') \setminus \mathcal{T}$ of vertices in G', or returns a type-1 or a type-2 witness in G'. Since we have assumed that R is not a good router, whenever we apply this algorithm to the current legal contracted graph G', we will obtain a contractible subset S of vertices, and by using procedure $\mathsf{Contract}(G', S)$, we can obtain a new legal contracted graph G'' with $|V(G'')| < |V(G')|$. We continue this process until $|V(G') \setminus \mathcal{T}| \le F(k)$ holds, and output G' as our sparsifier then. We now proceed to define the two types of witnesses.

Definition 5 *Let G' be a legal contracted graph, and let $\mathcal{F} = \{S'_1, \ldots, S'_r\}$ be a family of disjoint subsets of $V(G') \setminus \mathcal{T}$. We say that $\mathcal{F}$ is a type-1 witness, iff for each $1 \le j \le r$, S'_j is $\alpha_W(k^*)$-well-linked in graph G', and there is a collection $\mathcal{P}'_j$ of $\lceil k/2 \rceil$ edge-disjoint paths in graph G', where each path connects a distinct terminal in $\mathcal{T}$ to a distinct edge in $\mathrm{out}_{G'}(S'_j)$.*

Definition 6 *Let $\tilde{A} \subseteq V(G') \setminus \mathcal{T}$ be any subset of non-terminal vertices. We say that $\tilde{A}$ is a type-2 witness iff we are given a subset $\tilde{E} \subseteq \mathrm{out}_{G'}(\tilde{A})$ of $r \cdot \lceil k/4 \rceil$ edges, such that $\tilde{A}$ is $\alpha_W(r \cdot \lceil k/4 \rceil)$-well linked for $\tilde{E}$, and we are given a partition $E_1, \ldots, E_r$ of $\tilde{E}$ into r disjoint subsets of size $\lceil k/4 \rceil$ each, and a subset $\mathcal{T}^* \subseteq \mathcal{T}$ of $\lceil k/4 \rceil$ terminals, such that for each $1 \le j \le r$, there is a collection $\mathcal{P}'_j : \mathcal{T}^* \overset{1:1}{\rightsquigarrow}_2 E_j$ of paths in graph G'.*

(Here we say that $\tilde{A}$ is α-well-linked for $\tilde{E} \subseteq \mathrm{out}_{G'}(\tilde{A})$ iff for any partition (X, Y) of $\tilde{A}$, if we denote $T_X = \tilde{E} \cap \mathrm{out}_{G'}(X)$, and $T_Y = \tilde{E} \cap \mathrm{out}_{G'}(Y)$, then $|E_{G'}(X, Y)| \ge \alpha \cdot \min \{|T_X|, |T_Y|\}$.)

We start by showing that if a legal contracted graph G' contains a type-1 witness or a type-2 witness, then the set R is good router.

Theorem 7 *If any legal contracted graph G' contains a type-1 witness $\mathcal{F}$, or a type-2 witness $\tilde{A}$, then $R = V(G) \setminus \mathcal{T}$ is a good router.*

Proof: Recall that R is 1/3-well-linked. So we only need to prove that if G' contains a type-1 or a type-2 witness, then every pair of terminals can simultaneously send $1/k$ flow units to each other with congestion at most η^*. We need the following two simple claims, whose proofs appear in the Appendix.

Claim 3 *Let G' be a legal contracted graph, $S' \subseteq V(G') \setminus \mathcal{T}$, and $E' \subseteq \mathrm{out}_{G'}(S')$, such that S' is α-well-linked for E', for any $\alpha < 1$. Let $S \subseteq V(G) \setminus \mathcal{T}$ be the set of vertices obtained from S', after we replace every super-node $v_C \in S'$ with the set C of vertices. Then S is $\alpha/3$-well-linked for E' in graph G.*

Claim 4 *Let G' be a legal contracted graph for G, $S' \subseteq V(G') \setminus \mathcal{T}$ any subset of non-terminal vertices in G', and $E' \subseteq \mathrm{out}_{G'}(S')$ any subset of edges, and assume further that we are given a subset $\mathcal{T}' \subseteq \mathcal{T}$ of terminals with $|\mathcal{T}'| = |E'|$, such that there is a collection $\mathcal{P}' : \mathcal{T}' \overset{1:1}{\rightsquigarrow}_\eta E'$ of paths in G'. Let $S \subseteq V(G) \setminus \mathcal{T}$ be the set of vertices obtained from S' after we replace every super-node v_C by the set C of vertices, and consider the same subset $E' \subseteq \mathrm{out}_G(S)$ of edges. Then there is a set $\mathcal{P} : \mathcal{T}' \overset{1:1}{\rightsquigarrow}_{3\eta} E'$ of paths in graph G.*

Type-1 Witnesses Assume first that graph G' contains a type-1 witness $\mathcal{F} = \{S'_1, \ldots, S'_r\}$. Fix some $1 \le j \le r$, and consider the subset S'_j of vertices. Let S_j be the corresponding subset of vertices of the original graph G, after we un-contract each super-node $v_C \in S_j$, replacing it with the corresponding set C of vertices. Let $\mathcal{T}_j \subseteq \mathcal{T}$ be the subset of $\lceil k/2 \rceil$ terminals that serve as endpoints of the paths in $\mathcal{P}'_j$, and let $E_j \subseteq \mathrm{out}_{G'}(S'_j)$ be the subset of $\lceil k/2 \rceil$ edges where these paths terminate. From Claim 3, set S_j is $\alpha_W(k^*)/3$-well-linked. Therefore, every pair $(e, e') \in E_j$ of edges can simultaneously send to each other at least $\frac{1}{\lceil k/2 \rceil} \cdot \frac{\alpha_W(k^*)}{3\beta_{\mathrm{FCG}}(k^*)} \ge \frac{8}{kr}$ flow units with no congestion in $G[S_j]$. (We have used Equation 1). Denote this flow by F_j. From Claim 4, there is a set $\mathcal{P}_j : \mathcal{T}_j \overset{1:1}{\leadsto}_3 E_j$ of paths in graph G. Let $\mathcal{T}'_j = \mathcal{T} \setminus \mathcal{T}_j$. Then $|\mathcal{T}'_j| \le |\mathcal{T}_j| \le k/2$. Since graph G is $1/3$-well-linked for $\mathcal{T}$, there is a set $\mathcal{P}^*_j : \mathcal{T}'_j \overset{1:1}{\leadsto}_3 \mathcal{T}_j$ of paths in graph G. We now define a flow F^*_j, as follows: each terminal $t \in \mathcal{T}'_j$ sends $1/r$ flow units to some terminal in $\mathcal{T}_j$, along the path in $\mathcal{P}^*_j$ that originates at t. Next, each terminal $t' \in \mathcal{T}_j$ sends $\frac{2-1/k}{r}$ flow units to some edge in E_j, using the path in $\mathcal{P}_j$ that originates at t'. Each edge in E_j now receives $\frac{2-1/k}{r}$ flow units, and uses the flow F_j to spread this flow evenly among the edges of E_j. This defines the flow F^*_j, where every pair (t, t') of terminals sends $\frac{1}{kr}$ flow units to each other. The congestion of the flow F^*_j is computed as follows: the congestion due to flow on paths in $\mathcal{P}^*_j$ is at most $3/r$; the congestion due to flow on paths in $\mathcal{P}_j$ is at most $6/r$, and the congestion due to the flow F_j is at most 1. Notice that flow F_j is entirely contained inside $G[S_j]$.

The final flow F^* is simply the union of flows F_j for $1 \le j \le r$. Clearly, in F^*, every pair of terminals sends $1/k$ flow units to each other. It is easy to see that the flow congestion is bounded by 10.

Type-2 Witnesses Assume now that we are given a type-2 witness $\tilde{A}$, and let $A \subseteq V(G) \setminus \mathcal{T}$ be the subset of vertices obtained from $\tilde{A}$, after we replace each super-node v_C with the set C of vertices. From Claim 3, set A is $\frac{1}{3}\alpha_W(r \lceil k/4 \rceil)$-well-linked for the subset $\tilde{E} \subseteq \mathrm{out}_G(A)$ of edges. Therefore, every pair $(e, e') \in \tilde{E}$ of edges can send $\frac{1}{r \cdot \lceil k/4 \rceil} \cdot \frac{\alpha_W(r \cdot \lceil k/4 \rceil)}{3\beta_{\mathrm{FCG}}(r \cdot \lceil k/4 \rceil)} > \frac{16}{kr^2}$ flow units to each other with no congestion in graph G. (We have used Equation 1 and the fact that $k^* > r \cdot \lceil k/4 \rceil$). Let F denote this flow. Recall that for each $1 \le j \le r$, we have a collection $\mathcal{P}'_j : \mathcal{T}^* \overset{1:1}{\leadsto}_2 E_j$ of paths in graph G'. From Claim 4, there is a set $\mathcal{P}_j : \mathcal{T}^* \overset{1:1}{\leadsto}_6 E_j$ of paths in graph G. Finally, partition $\mathcal{T} \setminus \mathcal{T}^*$ into three subsets, $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ of size at most $\lceil k/4 \rceil$ each. Since graph G is $1/3$-well-linked, for each $1 \le i \le 3$, there is a set $\mathcal{Q}_i : \mathcal{T}_i \overset{1:1}{\leadsto}_3 \mathcal{T}^*$ of paths in G. Let $\mathcal{Q}$ denote the following set of paths: start with $\mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \mathcal{Q}_3$, and add, for each terminal $t \in \mathcal{T}^*$, an empty path Q_t connecting t to itself. Then set $\mathcal{Q}$ contains, for each terminal $t \in \mathcal{T}$, a path Q_t, connecting t to some terminal $t' \in \mathcal{T}^*$, such that for each terminal $t' \in \mathcal{T}^*$, there are exactly four terminals in $\mathcal{T}$ whose path Q_t terminates at t'. Notice that the paths in $\mathcal{Q}$ cause congestion at most 9 in G. We are now ready to define our final flow F^*. First, every terminal in $\mathcal{T}$ sends one flow unit to some terminal in $\mathcal{T}^*$, along the path $Q_t \in \mathcal{Q}$. Next, for each $1 \le j \le r$, each terminal $t \in \mathcal{T}^*$, sends $\frac{4}{r}$ flow units along the path in $\mathcal{P}_j$ that originates at t. Notice that each edge in $\tilde{E}$ now receives $\frac{4}{r}$ flow units. Finally, we use the flow F, to spread the flow that every edge receives evenly among the edges in $\tilde{E}$, so every pair of edges in $\tilde{E}$ needs to send $\frac{4}{r} \cdot \frac{1}{r \cdot \lceil k/4 \rceil} \le \frac{16}{kr^2}$ flow units to each other. This finishes the definition of the flow F^*. Clearly, every pair of terminals sends $1/k$ flow units to each other. We now analyze the congestion due to this flow. The paths in $\mathcal{Q}$ cause congestion 9, and the paths in $\mathcal{P}_1, \ldots, \mathcal{P}_r$ cause congestion at most 24 altogether (each set $\mathcal{P}_j$ of paths originally caused congestion

14

6, and we send $4/r$ flow units along each path in $\mathcal{P}_j$). Finally, flow F causes congestion at most 1. Altogether, flow F^* causes congestion at most 34. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

The next theorem provides an algorithm that, given any legal contracted graph G', either finds a contractible subset of vertices in G', or finds a witness of type 1 or 2 in G'.

Theorem 8 *Let G' be any legal contracted graph, and assume that $|V(G') \setminus \mathcal{T}| > F(k)$. Then there is an efficient algorithm that finds either a contractible subset S' of vertices, or a type-1 witness $\mathcal{F}$, or a type-2 witness $\tilde{A}$ in graph G'.*

Proof: Since we only work with graph G' in this proof, we omit the sub-script G' in our notation, and use $\text{out}(S)$ to denote $\text{out}_{G'}(S)$. Let $S \subseteq V(G') \setminus \mathcal{T}$ be any subset of non-terminal vertices. We say that a partition (X, Y) of S is *balanced*, iff $|X|, |Y| \geq |S|/4$. We start with the following lemma.

Lemma 1 *Let $S \subseteq V(G') \setminus \mathcal{T}$ be any subset of non-terminal vertices with $|S| > 2^9 \cdot F(k/2)$. Then there is an efficient algorithms that either finds a type-2 witness $\tilde{A}$, or a contractible set S' of vertices in G', or a balanced partition (X, Y) of S with $|E(X, Y)| \leq rk$.*

Proof: Let (X, Y) be any balanced partition of S, and assume w.l.o.g. that $|X| \geq |Y|$. If $|E(X, Y)| \leq rk$, then we stop and output the partition (X, Y). Otherwise, we perform a number of iterations. In each iteration, we are given as input a balanced partition (X, Y) of S with $|X| \geq |Y|$ and $|E(X, Y)| > rk$, and we try to establish whether X is a type-2 witness. If this is not the case, then we will either find a contractible subset S' of vertices in G', or we will produce a new balanced partition (X', Y') of S, with $|E(X', Y')| < |E(X, Y)|$. Therefore, after at most $|E(G')|$ steps, we are guaranteed to find a type-2 witness $\tilde{A}$, or a contractible set S' of vertices, or a balanced partition (X, Y) of S with $|E(X, Y)| \leq rk$.

We now proceed to describe each iteration. Suppose we are given a balanced partition (X, Y) of S with $|X| \geq |Y|$ and $|E(X, Y)| > rk$. Throughout the iteration execution, we denote $\Gamma = E(X, Y)$. An iteration consists of three steps. In the first step, we try to find a collection $\mathcal{P}_1$ of $\lceil k/4 \rceil$ edge-disjoint paths in graph G' connecting $\lceil k/4 \rceil$ distinct terminals in $\mathcal{T}$ to a subset E_1 of $\lceil k/4 \rceil$ edges in Γ. In the second step, we identify additional $(r - 1)$ subsets $E_2, \ldots, E_r$ of edges of Γ of size $\lceil k/4 \rceil$ each, and try to find, for each $1 \leq j \leq r$, a collection $\mathcal{P}_j$ of paths connecting terminals in $\mathcal{T}$ to the edges in E_j with congestion at most 2. Finally, in the third step, we set $\tilde{E} = \bigcup_{j=1}^{r} E_j$, and we try to establish whether X is $\alpha_W(r \cdot \lceil k/4 \rceil)$-well-linked for $\tilde{E}$. If all three steps succeed, then we output X as a type-2 witness. If any of the three steps fails, then we will either find a contractible set S' of vertices in G', or a new balanced partition (X', Y') of S with $|E(X', Y')| < |E(X, Y)|$. In the latter case, we continue to the next iteration with the new partition (X', Y') replacing the partition (X, Y). We now turn to describe each of the three steps.

Step 1 In this step we try to find a set $\mathcal{P}_1$ of edge-disjoint paths in graph G' connecting $\lceil k/4 \rceil$ distinct terminals in $\mathcal{T}$ to the edges of Γ. In order to do so, we set up the following flow network N. We sub-divide each edge $e \in \Gamma$ by a vertex z_e, and set $\mathcal{T}' = \{z_e \mid e \in \Gamma\}$. We then contract the vertices of $\mathcal{T}$ into a source s, and the vertices of $\mathcal{T}'$ into a sink t. Assume first that there is an s-t flow of value at least $\lceil k/2 \rceil$ in the resulting network N. This flow defines a collection $\mathcal{P}'$ of $\lceil k/2 \rceil$ paths, where each path connects a distinct terminal in $\mathcal{T}$ to some edge in Γ (since each terminal in $\mathcal{T}$ has exactly one adjacent edge in G'). These paths are completely edge-disjoint, except that each

15

edge in Γ may serve as an endpoint of up to two such paths. We select a subset $\mathcal{P}_1 \subseteq \mathcal{P}'$ of $\lceil k/4 \rceil$ paths, such that each edge in Γ now participates in at most one path in $\mathcal{P}_1$, that is, the paths in $\mathcal{P}_1$ are edge-disjoint.

Assume now that the value of the maximum s-t flow in N is less than $\lceil k/2 \rceil$. We show that in this case, we can either find a contractible set S' of vertices, or a balanced partition (X', Y') of S with $|E(X', Y')| \leq \lceil k/2 \rceil < rk$. Since the value of the maximum s-t flow in N is less than $\lceil k/2 \rceil$, there is an s-t cut (A', B') with $s \in A'$, $t \in B'$, and $|E(A', B')| < \lceil k/2 \rceil$ in N. Let $A = A' \setminus \{s\}$ and $B = B' \setminus \{t\}$. Then (A, B) is a partition of $V(G') \setminus \mathcal{T}$. Denote $X_A = X \cap A$, $X_B = X \cap B$, $Y_A = Y \cap A$, and $Y_B = Y \cap B$. Let $\Gamma' \subseteq \Gamma$ be the subset of edges $e = (u, v)$ where either $u \in A$, $v \in B$, or both $u, v \in A$ (see Figure 2). Notice that every edge in Γ' contributes at least 1 to the cut $E_N(A', B')$, and since $E_{G'}(A, B) \subseteq E_N(A, B) \cup \Gamma'$, we get that $|E_{G'}(A, B)| < \lceil k/2 \rceil$.
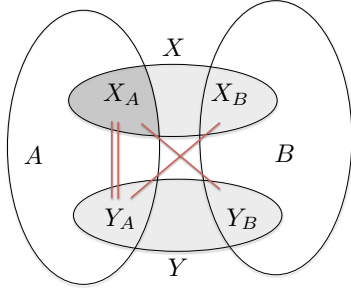

Figure 2: Illustration for Lemma 1. Edges in Γ' are shown in red.

Assume first that $|X_A| \geq |X_B|$. In this case, we define a new partition (X', Y') of S, where $X' = X_A$ and $Y' = Y \cup X_B$. It is immediate to see that (X', Y') is a balanced cut, since $|X'| \geq |X|/2 \geq |S|/4$. In order to bound $E(X', Y')$, observe that

$$E_{G'}(X', Y') = E_{G'}(X_A, X_B) \cup E_{G'}(X_A, Y) \subseteq E_{G'}(X_A, X_B) \cup \Gamma'.$$

Therefore, $|E_{G'}(X', Y')| \leq |E_{G'}(X_A, X_B)| + |\Gamma'| < \lceil k/2 \rceil < kr$.

From now on we assume that $|X_B| \geq |X_A|$, so $|X_B| \geq |S|/4 > 2^7 F(k/2)$. Let $\mathcal{C}_1$ be the set of all connected components of $G'[B]$. If for any component $C \in \mathcal{C}_1$, $|C| > 2^7 F(k/2)$, then we stop the algorithm, and output C as a contractible set. Indeed, $|C| > 2^7 F(k/2)$, while $|\operatorname{out}(C)| \leq |E_G'(A, B)| < \lceil k/2 \rceil$. We now assume that for all components $C \in \mathcal{C}_1$, $|C| \leq 2^7 F(k/2) < |S|/4$.

Let $\mathcal{C}_2$ be the set of all connected components of $G'[X_B \cup Y_B]$. Notice that each connected component $C \in \mathcal{C}_2$ must be contained in some connected component $C' \in \mathcal{C}_1$, so $|C| < |S|/4$ must hold. We construct a new partition (X', Y') of S, as follows. Start with $X' = \emptyset$, and add components $C \in \mathcal{C}_2$ to X' one-by-one, until $|X'| \geq |S|/4$ holds. Since the size of each such component is less than $|S|/4$, while $|X_B| \geq |S|/4$, in the end, $|S|/4 \leq |X'| \leq |S|/2$. Let $Y' = S \setminus X'$. Then (X', Y') is a balanced partition of S, and $E_{G'}(X', Y')| \subseteq E_{G'}(A, B)$, so $|E_{G'}(X', Y')| \leq |E_{G'}(A, B)| \leq \lceil k/2 \rceil < kr$.

Step 2 From now on, we assume that we have successfully found a set $\mathcal{P}_1$ of $\lceil k/4 \rceil$ edge-disjoint paths connecting a subset $\mathcal{T}^* \subseteq \mathcal{T}$ of $\lceil k/4 \rceil$ terminals to the edges in Γ. Let Γ_1 be the subset of $\lceil k/4 \rceil$ edges of Γ that serve as endpoints of these paths, so $\mathcal{P}_1 : \mathcal{T}^* \overset{1:1}{\rightsquigarrow}_1 \Gamma_1$.

We select arbitrary $(r-1)$ disjoint subsets $\Gamma_2, \ldots, \Gamma_r$ of $\Gamma \setminus \Gamma_1$, containing $\lceil k/4 \rceil$ edges of each. For each $2 \leq j \leq \Gamma_j$, we will try to find a collection $\mathcal{P}'_j$ of edge-disjoint paths, connecting the edges of Γ_1 to the edges of Γ_j. We will show that if such set of paths cannot be found, then we can find another balanced partition (X', Y') of S with $|E_{G'}(X', Y')| < |E_{G'}(X, Y)|$. For simplicity, we provide and analyze the procedure for $j = 2$, and the procedure is similar for all $2 \leq j \leq r$.

We set up the following flow network. Start with the graph $G'[X] \cup \Gamma_1 \cup \Gamma_2$. Let V_1 be the set of the endpoints of edges of Γ_1 that do not belong to X, $V_1 = \{v \mid (v, u) \in \Gamma_1, v \notin X\}$, and we define V_2 similarly for Γ_2. We then unify all vertices of V_1 into a source s, and all vertices of V_2 into a sink t. Let N' be the resulting network. Assume first that there is an s-t flow in N' of value $\lceil k/4 \rceil$. Then this flow defines a collection $\mathcal{P}'_2$ of $\lceil k/4 \rceil$ edge-disjoint paths, connecting the edges of Γ_1 to the edges of Γ_2. Concatenating the paths in $\mathcal{P}_1$ with the paths in $\mathcal{P}'_2$, we obtain a collection $\mathcal{P}_2 : \mathcal{T}^* \overset{1:1}{\leadsto}_2 \Gamma_2$ of paths in G'.

Assume now that such flow does not exist. Then there is an s-t cut (A, B) in N', with $s \in A$, $t \in B$, and $|E(A, B)| < \lceil k/4 \rceil$. We partition the edges of Γ_1 into two subsets: set T_A denotes the edges that do not belong to the cut $E_{N'}(A, B)$ (that is, for each edge $e = (s, v) \in T_A$, $v \in A$), and set T_B denotes edges that belong to the cut (for each edge $e = (s, v) \in T_B$, $v \in B$). Similarly, we partition the set Γ_2 of edges as follows: set T'_A contains all edges that belong to the cut $E_{N'}(A, B)$, and T'_B contains all edges that do not belong to the cut. The set $\Gamma \setminus (\Gamma_1 \cup \Gamma_2)$ of edges is also partitioned into two subsets: Υ_A denotes all edges $(u, v) \in \Gamma$ with $u \in Y$ and $v \in A$, and Υ_B denotes all edges $(u, v) \in \Gamma$ with $u \in Y$ and $v \in B$. Finally, let $E' = E_{G'}(A, B)$ (See Figure 3).
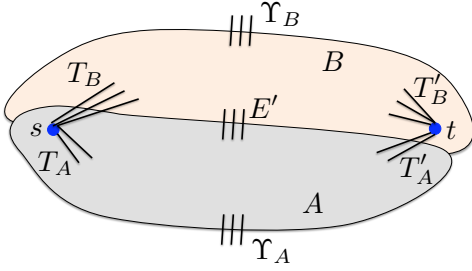

Figure 3: Illustration for Lemma 1

The set of edges that belong to the cut $E_{N'}(A, B)$ is $E' \cup T_B \cup T'_A$, and the value of this cut is less than $\lceil k/4 \rceil$. In particular, since $|T_A \cup T_B| = \lceil k/4 \rceil$ and $|T'_A \cup T'_B| = \lceil k/4 \rceil$, it follows that $|E'| < |T_A|$, and $|E'| < |T'_B|$. Assume first that $|A| \leq |B|$. We then define a new partition (X', Y') of S, where $X' = B$ and $Y' = Y \cup A$. It is easy to see that (X', Y') is a balanced cut. Notice that $E_{G'}(X, Y) = T_A \cup T'_A \cup T_B \cup T'_B \cup \Upsilon_A \cup \Upsilon_B$, while $E(X', Y') = T_B \cup T'_B \cup \Upsilon_B \cup E'$. In order to show that $|E(X', Y')| < |E(X, Y)|$, it is enough to prove that $|E'| < |T_A| + |T'_A|$, which follows from the fact that $|E'| < |T_A|$.

Otherwise, if $|A| > |B|$, we define a new partition (X', Y') of S where $X' = A$ and $Y' = Y \cup B$. Again, it is easy to see that (X', Y') is a balanced cut. Notice that $E(X, Y) = T_A \cup T'_A \cup T_B \cup T'_B \cup \Upsilon_A \cup \Upsilon_B$, while $E(X', Y') = T_A \cup T'_A \cup \Upsilon_A \cup E'$. In order to show that $|E(X', Y')| < |E(X, Y)|$, it is enough to prove that $|E'| < |T_B| + |T'_B|$, which follows from the fact that $|E'| < |T_A|$.

We say that steps 1 and 2 are successful iff we have found r disjoint subsets $\Gamma_1, \ldots, \Gamma_r$ of Γ containing $\lceil k/4 \rceil$ edges each, and for each $1 \leq j \leq r$, we have found a set $\mathcal{P}_j$ of $\lceil k/4 \rceil$ edge-disjoint paths, $\mathcal{P}_j : \mathcal{T}^* \overset{1:1}{\leadsto}_2 \Gamma_j$. We assume from now on that steps 1 and 2 have been successful. We now

proceed to describe step 3.

Step 3 Let $\Gamma' = \bigcup_{j=1}^r \Gamma_j$. In this step, we try to verify that X is $\alpha_W(r \cdot \lceil k/4 \rceil)$-well-linked for Γ'. If this is not the case, then we return a balanced partition (X', Y') of S with $|E_{G'}(X', Y')| < |E_{G'}(X, Y)|$. We set up an instance of the sparsest cut problem, as follows. Start with the graph G' and sub-divide every edge $e \in \Gamma'$ by a vertex v_e. Let $\mathcal{T}' = \{v_e \mid e \in \Gamma'\}$, and let G'' be the sub-graph of the resulting graph induced by $X \cup \mathcal{T}'$. We run algorithm $\mathcal{A}_{\text{ARV}}$ on the instance $(G'', \mathcal{T}')$ of the sparsest cut problem. Let A, B be the resulting partition of X, and assume w.l.o.g. that $|A| \leq |B|$. Denote $T_A = \text{out}_{G'}(A) \cap \Gamma'$ and $T_B = \text{out}_{G'}(B) \cap \Gamma'$. Assume first that $|E_{G'}(A, B)| < \min\{|T_A|, |T_B|\}$. We then define a new partition (X', Y') of S, where $X' = B$ and $Y' = A \cup Y$. It is easy to see that (X', Y') is a balanced partition, since $|B| \geq |A|$. Moreover, $|E_{G'}(X', Y')| \leq |E_{G'}(X, Y)| - |T_A| + |E_{G'}(A, B)| < |E_{G'}(X, Y)|$ as required.

Assume now that $|E_{G'}(A, B)| \geq \min\{|T_A|, |T_B|\}$. Then we are guaranteed that X is $(1/\alpha_{\text{ARV}}(r \cdot \lceil k/4 \rceil)) \geq \alpha_W(r \cdot \lceil k/4 \rceil)$-well-linked for Γ'. We then declare that X is a type-2 witness and terminate the algorithm. Indeed, we have established that X is $\alpha_W(r \cdot \lceil k/4 \rceil)$-well-linked for Γ', and we have found, for each $1 \leq j \leq r$, a collection $\mathcal{P}_j : \mathcal{T}^* \overset{1:1}{\leadsto}_2 \Gamma_j$ of paths in G'. □

We are now ready to complete the proof of Theorem 8. The algorithm consists of two phases. In the first phase we have $\lceil \log r \rceil$ iterations. In each iteration i, we start with a family $\mathcal{S}_i$ of 2^{i-1} disjoint subsets of vertices of $V(G') \setminus \mathcal{T}$, where for each $S \in \mathcal{S}_i$ $|S| > 2^9 F(k/2)$, and produce a family $\mathcal{S}_{i+1}$ of 2^i subsets, that become an input to the next iteration. In the input to the first iteration, $\mathcal{S}_1 = \{V(G') \setminus \mathcal{T}\}$. Iteration i is executed as follows. Consider some set $S \in \mathcal{S}_i$. We apply the algorithm from Lemma 1 to set S. If the output is a type-2 witness $\tilde{A}$, or a contractible set S' of vertices, we stop the algorithm and output this set. Otherwise, we obtain a balanced partition (X, Y) of S, with $|E(X, Y)| \leq rk$. In this case, we add X and Y to $\mathcal{S}_{i+1}$. Notice that $|\text{out}(X)|, |\text{out}(Y)| \leq |\text{out}(S)| + rk$. We let $\mathcal{S}_{i+1}$ be the set obtained after we process all sets $S \in \mathcal{S}_i$. Observe that since we find balanced cuts in each iteration, for each $1 \leq i \leq \lceil \log r \rceil + 1$, for each $S \in \mathcal{S}_i$, $|S| \geq \frac{|V(G') \setminus \mathcal{T}|}{4^{i-1}} \geq \frac{F(k)}{4^{\lceil \log r \rceil}} \geq \frac{F(k)}{4r^2} > 2^9 \cdot F(k/2)$, and so we can indeed apply Lemma 1 to all sets in $\mathcal{S}_i$.

Consider now the output of the last iteration $\mathcal{S}_{\lceil \log r \rceil}$, and let $S_1, \ldots, S_r$ be any r sets in $\mathcal{S}_{\lceil \log r \rceil + 1}$. Fix some $j : 1 \leq j \leq r$, and consider the set S_j. From the above discussion, $S_j \subseteq V(G') \setminus \mathcal{T}$. Moreover, since $|\text{out}(V(G') \setminus \mathcal{T})| = k$, and in each iteration, if we start with a set S and produce a partition (X, Y) of S, then $|\text{out}(X)|, |\text{out}(Y)| \leq |\text{out}(S)| + rk$, we get that $|\text{out}(S_j)| \leq k + rk \cdot \lceil \log r \rceil \leq 2kr \log r = k^*$. Moreover, as observed above, $|S_j| \geq \frac{F(k)}{16r^2} \geq 2^{12}r \log r \cdot F(k/2)$.

Let $\mathcal{W}_j$ be the weak well-linked decomposition of S_j, given by Theorem 3. Notice that from the definition of well-linkedness, for every cluster $C \in \mathcal{W}_j$, $G'[C]$ is connected. If any set $R' \in \mathcal{W}_j$, with $|\text{out}(R')| \leq \lceil k/2 \rceil$ is contractible, then we simply output R' as a contractible set. From now on assume that all sets in $\mathcal{W}_j$ are non-contractible. Notice that for each $R' \in \mathcal{W}_j$, set R' is $\alpha_W(k^*)$-well-linked. Let $S'_j \in \mathcal{W}_j$ be the set of maximum cardinality. We need the following claim.

Claim 5 $|S'_j| > 2^7 \cdot F(k/2)$.

Proof: Recall that from Theorem 3, $\sum_{R' \in \mathcal{W}_j} |\text{out}(R')| \leq 1.2 |\text{out}(S_j)| \leq 2.4kr \log r$. We partition the set $\mathcal{W}_j$ into two subsets: $\mathcal{W}^1$ contains all sets $R' \in \mathcal{W}_j$ with $|\text{out}(R')| \geq k/2$, and $\mathcal{W}^2$ contains all remaining sets. Further, we partition the set $\mathcal{W}^2$ into subsets $\mathcal{R}_i$, for $2 \leq i \leq \log k + 1$, as follows: $\mathcal{R}_i$ contains all sets R' with $k/2^i < |\text{out}(R')| \leq k/2^{i-1}$.

Fix some $2 \le i \le \log k + 1$. Since $\sum_{R' \in \mathcal{W}_j} |\operatorname{out}(R')| \le 2.4kr \log r$, we get that $|\mathcal{R}_i| \le 3r \log r \cdot 2^i$, and since each set $R' \in \mathcal{R}_i$ is non-contractible, $|R'| \le 2^7 \cdot F(k/2^{i-1})$ must hold. We therefore obtain the following bound:

$$\sum_{R' \in \mathcal{W}^2} |R'| \le \sum_{i=2}^{\log k+1} 3 \cdot 2^i \cdot r \log r \cdot 2^7 \cdot F(k/2^{i-1})$$

$$= 3 \cdot 2^7 \cdot r \log r \sum_{i=2}^{\log k+1} 2^i F(k/2^{i-1})$$

Denote $T(i) = 2^i F(k/2^{i-1})$. By the recursive definition of $F(k')$, $T(i) < \frac{2^i}{8} F(k/2^{i-2}) = T(i-1)/4$. Therefore, the values $T(i)$ form a geometric series, and $\sum_{i=2}^{\log k+1} T(i) < 4T(2)/3$. We conclude that $\sum_{R' \in \mathcal{W}^2} |R'| < 2^{11} r \log r \cdot F(k/2) \le |S_j|/2$, and so $\sum_{R' \in \mathcal{W}^1} |R'| > |S_j|/2 \ge 2^{11} r \log r \cdot F(k/2)$.

Finally, observe that set $\mathcal{W}^1$ may contain at most $5r \log r$ sub-sets, since for each subset $R' \in \mathcal{W}^1$, $|\operatorname{out}(R')| > k/2$, while $\sum_{R' \in \mathcal{W}_j} |\operatorname{out}(R')| \le 2.4rk \log r$. Therefore, at least one subset in $\mathcal{W}^1$ contains more than $\frac{2^{11} r \log r \cdot F(k/2)}{5r \log r} > 2^7 \cdot F(k/2)$ vertices. □

Next, we try to route the terminals in $\mathcal{T}$ to the edges in $\operatorname{out}(S'_j)$, as follows. We build a flow network, starting from the graph G', contracting all terminals in $\mathcal{T}$ into a source s, and all vertices in S'_j into a sink t. We try to find an s-t flow in this network of value at least $\lceil k/2 \rceil$. Assume first that we are unable to find such flow. Then we can find a minimum s-t cut (A, B), with $s \in A$, $t \in B$, and the cut value is less than $\lceil k/2 \rceil$ in this network. Let $B' = (B \setminus \{t\}) \cup S'_j$. Then $B' \subseteq V(G') \setminus \mathcal{T}'$, and it is a contractible set, since $|\operatorname{out}_{G'}(B')| \le \lceil k/2 \rceil$, while $S'_j \subseteq B'$, so $|B'| > 2^7 \cdot F(k/2)$. Moreover, $G'[B']$ is a connected graph, since $G'[S_j]$ is connected.

Assume now that we have managed to find a flow of value at least $\lceil k/2 \rceil$ in this network. Then this defines a collection $\mathcal{P}'_j$ of $\lceil k/2 \rceil$ edge-disjoint paths, connecting distinct terminals in $\mathcal{T}$ to distinct edges of $\operatorname{out}(S'_j)$.

Overall, our algorithm may terminate early, in which case it is guaranteed to produce either a type-2 witness $\tilde{A}$, or a contractible set. Otherwise, the algorithm finds a family $\mathcal{F} = \{S'_1, \ldots, S'_r\}$ of vertex subsets that are $\alpha_W(k^*)$-well-linked, with the collections $\mathcal{P}'_1, \ldots, \mathcal{P}'_r$ of paths as required, thus giving a type-1 witness. □

We are now ready to complete the proof of Theorem 6. If the set $R = V(G) \setminus \mathcal{T}$ is a good router, then we let $\mathcal{C} = \{R\}$, and the sparsifier H is the corresponding contracted graph (a star graph, where the star center is v_R, and the leaves are the terminals). Assume now that R is not a good router. We then start with $G' = G$, and repeatedly apply Theorem 8 to G'. From Theorem 7, graph G' cannot contain type-1 or type-2 witnesses, so the output of the theorem will always be a contractible set S of vertices in G'. We then apply Procedure $\mathsf{Contract}(G', S)$ to obtain a new contracted graph G'', with $|V(G'')| < |V(G')|$, and continue. We are guaranteed to obtain, after at most $|V(G)|$ iterations, a legal contracted graph G' with $|V(G') \setminus \mathcal{T}| \le F(k)$, which we output as the final sparsifier. From Claim 1, G' is indeed a quality-$(2\eta^*)$-sparsifier.

In order to bound the running time of the algorithm, we prove that for any n-vertex graph with a set $\mathcal{T}$ of k terminals, the running time of the algorithm is $T(n, k) = n^{O(\log k)} \cdot 2^k$. The proof is by induction on the values of k. For $k \le 4$, the running time of the algorithm is $\operatorname{poly}(n)$. Assume that the claim holds for all values $k' < k$, and we now prove it for k. Recall that our algorithm performs at most n iterations. Each iteration involves a call to procedure $\mathsf{Contract}(G', S)$, and

takes an additional time of $\text{poly}(n)$. Let $k' = |\text{out}(S)|$, and recall that $k' \leq \lceil k/2 \rceil$. Procedure $\mathsf{Contract}(G', S)$ computes a strong well-linked decomposition $\mathcal{S}$ of the set S, and then computes a sparsifier for each set $Z \in \mathcal{S}$ recursively. For each set $Z \in \mathcal{S}$, let $k_Z = \text{out}_G(Z)$, and let $n_Z = |Z|$. Then $k_Z \leq k' \leq \lceil k/2 \rceil$, and $\sum_{Z \in \mathcal{S}} n_Z \leq n$. Therefore, by the induction hypothesis, the running time of the recursive procedure for each set $Z \in \mathcal{S}$ is at most $T(n_Z, k_Z) \leq T(n_Z, \lceil k/2 \rceil)$, and the total running time of procedure $\mathsf{Contract}(G', S)$ is at most $2^k \text{poly}(n) + \sum_{Z \in \mathcal{S}} T(n_Z, \lceil k/2 \rceil) \leq 2^k \text{poly}(n) + T(n, \lceil k/2 \rceil)$. Overall, the running time of the algorithm is then bounded by $n \cdot (\text{poly}(n) + 2^k \text{poly}(n) + T(n, \lceil k/2 \rceil)) \leq 2^k \text{poly}(n) + n \cdot (n^{O(\log \lceil k/2 \rceil)} \cdot 2^k) \leq n^{O(\log k)} \cdot 2^k = T(n, k)$.

References

[And10]   Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via raecke decompositions. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 277–286, Washington, DC, USA, 2010. IEEE Computer Society.

[AR98]   Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[ARV09]   Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[CKS04]   Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. The all-or-nothing multicommodity flow problem. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 156–165, New York, NY, USA, 2004. ACM.

[CKS05]   Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 183–192, New York, NY, USA, 2005. ACM.

[CLLM10]   Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 265–274, Washington, DC, USA, 2010. IEEE Computer Society.

[EGK$^+$10]   Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: new results from old techniques. In *Proceedings of the 13th international conference on Approximation, and 14 the International conference on Randomization, and combinatorial optimization: algorithms and techniques*, APPROX/RANDOM'10, pages 152–165, Berlin, Heidelberg, 2010. Springer-Verlag.

[FJS88]   T. Figiel, W. B. Johnson, and F. Schechtman. Factorizations of natural embeddings of l_p^n into L_r. I. *Studia Math.*, 89:79–103, 1988.

[GVY95]   N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)-cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1995.

[LLR94]   N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Proceedings of 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 577–591, 1994.

[LM10]   F. Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 47–56, New York, NY, USA, 2010. ACM.

[LR99]   F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787–832, 1999.

[MM10]   Konstantin Makarychev and Yury Makarychev. Metric extension operators, vertex sparsifiers and lipschitz extendability. In *FOCS*, pages 255–264. IEEE Computer Society, 2010.

[Moi09]   Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *FOCS*, pages 3–12. IEEE Computer Society, 2009.

[Räc02]   Harald Räcke. Minimizing congestion in general networks. In *In Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 43–52, 2002.

[RZ10]   Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

A   List of Parameters

$\alpha_{\mathrm{ARV}}(k)$	$O(\sqrt{\log k})$	Approximation factor of algorithm $\mathcal{A}_{\mathrm{ARV}}$ for sparsest cut
$\alpha_W(z)$	$\Omega\left(\frac{1}{\log^{3/2} z}\right)$	Well-linkedness parameter for the weak well-linked decomposition
$\beta_{\mathrm{FCG}}(k)$	$O(\log k)$	Flow-cut gap for undirected graphs
η^*	34	Parameter from the definition of good routers
r	$O(\log^3 k)$	Number of vertex subsets in a type-1 witness
k^*	$2kr \log r = k\,\mathrm{poly}\log k$	

B Proofs Omitted from Section 2

B.1 Proof of Theorem 3

We use the $\alpha_{\text{ARV}}(z)$-approximation algorithm $\mathcal{A}_{\text{ARV}}$ for the sparsest cut problem. We set $\alpha_W(z) = \frac{1}{2^7 \alpha_{\text{ARV}}(z) \log z} = \Omega\left(\frac{1}{\log^{3/2} z}\right)$.

Throughout the algorithm, we maintain a partition $\mathcal{W}$ of the input set S of vertices, where for each $R \in \mathcal{W}$, $|\operatorname{out}(R)| \leq |\operatorname{out}(S)|$. At the beginning, $\mathcal{W}$ consists of the subsets of S defined by the connected components of $G[S]$.

Let $R \in \mathcal{W}$ be any set in the current partition, and let $(G_R, \mathcal{T}_R')$ be the instance of the sparsest cut problem corresponding to R, as defined in Section 2. We say that a cut (A', B') in G_R is sparse, iff its sparsity is less than $1/(2^7 \log z)$. We apply the algorithm $\mathcal{A}_{\text{ARV}}$ to the instance $(G_R, \mathcal{T}_R')$ of sparsest cut. If the algorithm returns a cut (A', B'), that is a sparse cut, then let $A = A' \setminus \mathcal{T}_R'$, and $B = B' \setminus \mathcal{T}_R'$. We remove R from $\mathcal{W}$, and add A and B to it instead. Let $T_A = \operatorname{out}(R) \cap \operatorname{out}(A)$, and $T_B = \operatorname{out}(R) \cap \operatorname{out}(B)$, and assume w.l.o.g. that $|T_A| \leq |T_B|$. Then $|E(A, B)| < |T_A|/(2^7 \log z)$ must hold, and in particular, $|\operatorname{out}(A)| \leq |\operatorname{out}(B)| \leq |\operatorname{out}(R)| \leq |\operatorname{out}(S)|$. For accounting purposes, each edge in set T_A is charged $1/(2^7 \log z)$ for the edges in $E(A, B)$. Notice that the total charge to the edges in T_A is $|T_A|/(2^7 \log z) \geq |E(A, B)|$. Notice also that since $|T_A| \leq |\operatorname{out}(R)|/2$ and $|E(A, B)| \leq |T_A|/2^7$, $|\operatorname{out}(A)| \leq 0.51 |\operatorname{out}(R)|$.

The algorithm stops when for each set $R \in \mathcal{W}$, the procedure $\mathcal{A}_{\text{ARV}}$ returns a cut that is not sparse. We argue that this means that each set $R \in \mathcal{W}$ is $\alpha_W(z)$-well-linked. Assume otherwise, and let $R \in \mathcal{W}$ be a set that is not $\alpha_W(z)$-well-linked. Then, by the definition of well-linkedness, the corresponding instance of the sparsest cut problem must have a cut of sparsity less than $\alpha_W(z) = 1/(2^7 \alpha_{\text{ARV}}(z) \log z)$. The algorithm $\mathcal{A}_{\text{ARV}}$ should then have returned a cut whose sparsity is less than $\alpha_W(z) \cdot \alpha_{\text{ARV}}(z) = 1/(2^7 \log z)$, that is a sparse cut.

Finally, we need to bound $\sum_{R \in \mathcal{W}} |\operatorname{out}(R)|$. We use the charging scheme defined above. Consider some iteration where we partition the set R into two subsets A and B, with $|T_A| \leq |T_B|$. Recall that each edge in T_A is charged $1/(2^7 \log z)$ in this iteration, while $|\operatorname{out}(A)| \leq 0.51 |\operatorname{out}(R)|$ holds. Consider some edge $e = (u, v)$. Whenever e is charged via the vertex u, the size of the set $\operatorname{out}(R)$, where $u \in R \in \mathcal{W}$ goes down by a factor of at least 0.51. Therefore, e can be charged at most $2 \log z$ times via each of its endpoints. The total charge to e is then at most $4 \log z/(2^7 \log z) = 1/2^5$. This however only accounts for the *direct* charge. For example, some edge $e' \notin \operatorname{out}(S)$, that was first charged to the edges in $\operatorname{out}(S)$, can in turn be charged for some other edges. We call such charging *indirect*. If we sum up the indirect charge for every edge $e \in \operatorname{out}(S)$, we obtain a geometric series, and so the total direct and indirect amount charged to every edge $e \in \operatorname{out}(S)$ is at most $1/2^4 < 0.1$. Therefore, $\sum_{R \in \mathcal{W}} |\operatorname{out}(R)| \leq 1.2 |\operatorname{out}(S)|$ (we need to count each edge $e \in \left(\bigcup_{R \in \mathcal{W}} \operatorname{out}(R)\right) \setminus \operatorname{out}(S)$ twice: once for each its endpoint).

B.2 Proof of Theorem 4

The algorithm is very similar to the algorithm used in the proof of Theorem 3, but the analysis is different. We start by describing the algorithm.

Throughout the algorithm, we maintain a partition $\mathcal{S}$ of the input set S of vertices, where for each $R \in \mathcal{S}$, $|\operatorname{out}(R)| \leq |\operatorname{out}(S)|$. At the beginning, $\mathcal{S} = \{S\}$.

Let $R \in \mathcal{S}$ be any set in the current partition, and let $(G_R, \mathcal{T}_R')$ be the corresponding instance of the sparsest cut problem. We say that a cut (A', B') in G_R is sparse, iff its sparsity is less than

1/3. Notice that the set R is 1/3-well-linked iff there is no sparse cut in R

Our algorithm proceeds as follows. Whenever there is a set $R \in \mathcal{S}$, such that the sparsity of the sparsest cut (A', B') in the corresponding instance $(G_R, \mathcal{T}'_R)$ is less than 1/3, we set $A = A' \setminus \mathcal{T}'_R$ and $B = B' \setminus \mathcal{T}'_R$. We then remove R from $\mathcal{S}$, and add A and B to it instead. Since $|\operatorname{out}(R)| \leq |\operatorname{out}(S)| \leq z$, the sparsest cut problem instance can be solved in time $2^z \operatorname{poly}(n)$: we simply go over all bi-partitions $(\mathcal{T}_1, \mathcal{T}_2)$ of the set $\mathcal{T}'_R$ of terminals, and for each such bi-partition, compute the minimum cut separating the vertices in $\mathcal{T}_1$ from the vertices in $\mathcal{T}_2$. The algorithm terminates when for every set $R \in \mathcal{S}$, the value of the sparsest cut in the corresponding instance is at least 1/3. From the above discussion, the running time of the algorithm is $2^z \operatorname{poly}(n)$. It is also clear that when the algorithm terminates, for every set $R \in \mathcal{S}$, $|\operatorname{out}(R)| \leq |\operatorname{out}(S)|$, and R is 1/3-well-linked. It now only remains to analyze the sizes of the collections $\mathcal{S}_i$ of vertex subsets in the resulting partition, and to bound $\sum_{R \in \mathcal{S}} |\operatorname{out}(R)|$.

Let $\mathcal{C}$ be the set of all cuts that we produce throughout this algorithm (that is, $\mathcal{C}$ contains all sets R that belonged to $\mathcal{S}$ at any stage of the algorithm), and let T be the corresponding partitioning tree, whose vertices represent the sets in $\mathcal{C}$, and every inner vertex v_R has exactly two children v_A, v_B, where (A, B) is the partition that our procedure computed for the set R.

For the sake of the analysis of the algorithm, we define a new graph G', as follows. We start with the graph G, and the final partition $\mathcal{S}$ of S. Let $R, R' \in \mathcal{S}$ be any pair of distinct vertex subsets, and let $e = (u, v)$ be any edge with $u \in R, v \in R'$. We subdivide the edge e by adding two vertices u_e, v_e to it, so that now we have a path (u, u_e, v_e, v) instead of the edge e. Additionally, for every edge $e = (u, v)$ with $u \in S$, $v \notin S$, we subdivide edge e by adding a new vertex u_e to it.

All the newly added vertices are called *terminals* and the set of all such terminals is denoted by Γ. The resulting graph is denoted by G'. For each subset $X \subseteq S$ of vertices, we will still denote by $\operatorname{out}(X) = \operatorname{out}_G(X)$. Consider now some subset $R \in \mathcal{C}$ of vertices. We now define the subset $\Gamma(R) \subseteq \Gamma$ of terminals associated with R, as follows: $\Gamma(R) = \{u_e \mid e = (u, v) \in \operatorname{out}(R), u \in R\}$, so $|\Gamma(R)| = |\operatorname{out}(R)|$. Moreover, $\sum_{R \in \mathcal{S}} |\operatorname{out}(R)| = \sum_{R \in \mathcal{S}} |\Gamma(R)| = |\Gamma|$.

We define a charging scheme that will help us bound the number of terminals, and the number of sets in each collection $\mathcal{S}_i$.

We say that a set $R \in \mathcal{C}$ belongs to level i iff $\frac{z}{2^{i/2}} < |\operatorname{out}(R)| \leq \frac{z}{2^{(i-1)/2}}$. In particular, S is a level-1 set, and we have at most $2 \log z + 1$ levels. We also partition all terminals into levels, and within each level, we have two types of terminals: regular and special. Intuitively, special terminals at level i are the terminals that have been created by partitioning some sets from levels $1, \ldots, i-1$, while regular terminals are created by partitioning sets that belong to level i.

The terminals in set $\Gamma(S)$ are called special terminals, and they belong to level 1. Let R be some level-i set, and let A, B be its children, with $|\operatorname{out}(R) \cap \operatorname{out}(A)| \leq |\operatorname{out}(R) \cap \operatorname{out}(B)|$. Then $|E(A, B)| \leq |\operatorname{out}(R) \cap \operatorname{out}(A)|/3 = |\Gamma(R) \cap \Gamma(A)|/3$, and so $|\Gamma(A)| = |\Gamma(R) \cap \Gamma(A)| + |E(A, B)| \leq |\Gamma(R)|(\frac{1}{2} + \frac{1}{6}) < |\Gamma(R)|/\sqrt{2}$. Let i' and i'' be the levels to which sets A and B belong, respectively. Then $i' \geq i+1$ must hold. We call all terminals in set $\Gamma(A) \setminus \Gamma(R)$ *special terminals for level i'* (to indicate that they were created from partitioning a lower-level set). If $i'' \geq i+1$ holds as well, then we call all terminals in set $\Gamma(B) \setminus \Gamma(R)$ special terminals at level i''. Otherwise, they are *regular terminals*, that belong to level $i'' = i$.

Notice that both sets $\Gamma(A) \setminus \Gamma(R)$ and $\Gamma(B) \setminus \Gamma(R)$ of new terminals come from subdivisions of the edges in $E(A, B)$, so each such edge gives rise to one terminal in $\Gamma(A)$ and one in $\Gamma(B)$. We stress that for each i, a level-i terminal continues to be a level-i terminal throughout the algorithm, even if its corresponding subset of vertices in $\mathcal{S}$ becomes further subdivided and stops being a level-i

set. So for example, if R is a level-i set, the terminals in $\Gamma(R)$ may belong to levels $1, \ldots, i$. The next lemma bounds the number of terminals at each level, and it is central to the analysis of the algorithm.

Lemma 2 *For each $1 < i \leq 2 \log z + 1$, there are at most $2^{i-2}z$ special terminals, and at most $2^{i-1}z$ regular level-i terminals. For $i = 1$, there are z level-1 special terminals, and at most $z/2$ regular level-1 terminals.*

Proof: For each level i, let n_i be the number of special terminals, and n_i' the number of regular terminals. Let S_i be the total number of terminals at levels $1, \ldots, i$. We use the following two simple claims.

Claim 6 *For each $i > 1$, $n_i \leq S_{i-1}/3$.*

Proof: Recall that a level-i special terminal can only be created when partitioning some cluster $R \in \mathcal{C}$ that belongs to levels $1, \ldots, i-1$. Suppose that the partition of R is (A, B), and assume that $|\Gamma(A) \cap \Gamma(R)| \leq |\Gamma(B) \cap \Gamma(R)|$. Let $X \in \{A, B\}$ be the cluster that belongs to level-i (it is possible that both A and B belong to level i - the analysis for this case is similar and is carried out for each one of the clusters separately). Then the terminals in $\Gamma(X) \cap \Gamma(R)$ belong to levels $1, \ldots, i-1$, and the terminals in $\Gamma(X) \setminus \Gamma(R)$ become special terminals at level i. We charge the terminals in $\Gamma(X) \cap \Gamma(R)$ for the terminals in $\Gamma(X) \setminus \Gamma(R)$, where the charge to every terminal in $\Gamma(X) \cap \Gamma(R)$ is at most $1/3$. Moreover, since X now becomes a level-i cluster, the terminals in $\Gamma(X)$ will never be charged for special level-i terminals again. Therefore, the number of special level-i terminals is at most $S_{i-1}/3$. $\qquad\square$

Claim 7 *For every level i, $n_i' \leq (S_{i-1} + n_i)/2$.*

Proof: The proof uses a charging scheme, that is defined as follows. Let Γ' be the set of all terminals at levels $1, \ldots, i-1$, together with the special terminals at level i. We charge the regular level-i terminals to the terminals in Γ', and show that $n_i' \leq |\Gamma'|/2$.

Recall that regular level-i terminals are only created by partitioning level-i clusters $R \in \mathcal{C}$. Let R be any such level-i cluster, that we have partitioned into (A, B). Assume w.l.o.g. that $|T_A| \leq |T_B|$, and recall that A must belong to some level $i' > i$. This partition only creates level-i terminals if B belongs to level i. The number of the newly created level-i terminals, $|\Gamma(B) \setminus \Gamma(R)| = |E(A, B)| \leq |T_A|/3 = |\Gamma(A) \cap \Gamma(R)|/3$. We charge the terminals in $|\Gamma(A) \cap \Gamma(R)|$ for the newly created terminals in $\Gamma(B) \setminus \Gamma(R)$, where each terminal in $|\Gamma(A) \cap \Gamma(R)|$ is charged $1/3$. Observe that the terminals in $|\Gamma(A) \cap \Gamma(R)|$ must belong to levels $1, \ldots, i$, and moreover, since A does not belong to level i, these terminals will never be charged again for level-i terminals. But we may charge them indirectly - by charging the terminals in $\Gamma(B) \setminus \Gamma(R)$ for some new terminals. However, since a direct charge to every terminal is bounded by $1/3$, the total direct and indirect charge to any terminal in Γ' forms a geometrically decreasing sequence, and its sum is bounded by $1/2$. Therefore, $n_i' \leq |\Gamma'|/2$. $\quad\square$

The number of special level-1 terminals is z, and the number of regular level-1 terminals is at most $z/2$ by Claim 7, so $S_1 \leq 3z/2$. In general, we now have that for any $i > 1$, $n_i \leq S_{i-1}/3$, and $n_i' \leq (S_{i-1} + n_i)/2 \leq 2S_{i-1}/3$. Therefore, $S_i = S_{i-1} + n_i + n_i' \leq 2S_{i-1}$. We conclude that $S_i \leq 3 \cdot 2^{i-2}z$ for all i, and so $n_i \leq 2^{i-2}z$, and $n_i' \leq 2^{i-1}z$ for all $i > 1$. $\quad\square$

Consider now the final partition $\mathcal{S}$. We can now bound $\sum_{R \in \mathcal{S}} |\text{out}(R)| = |\Gamma| = |S_{2 \log z + 1}| \leq 3z \cdot 2^{2 \log z} = O(z^3)$.

Let $R \in \mathcal{S}$, and assume that R belongs to level i. Then all terminals of $\Gamma(R)$ must belong to levels $1, \ldots, i$. The total number of such terminals is bounded by $3z \cdot 2^{i-2}$, and set R uses at least $z/2^{i/2}$ of them. It follows that the number of level-i sets in $\mathcal{S}$ is bounded by $2^{3i/2}$.

Recall that $\mathcal{S}_i$ contains all sets $R \in \mathcal{S}$ with $z/2^i < |\text{out}(R)| \leq z/2^{i-1}$, and so $\mathcal{S}_i$ only contains sets $R \in \mathcal{S}$ that belong to levels $2i$ or $2i - 1$. From the above discussion $|\mathcal{S}_i| \leq 2^{3i+3}$.

C   Proof of Corollary 1

Assume first that all edge capacities are integral and bounded by C. For each terminal $t \in \mathcal{T}$, let C_t be the total capacity of all edges incident on t. We replace every edge $e \in E$ with c_e parallel edges of unit capacity. For each terminal $t \in \mathcal{T}$, we sub-divide each edge e incident on t with a vertex v_e, and we let S_t be the set of these new vertices. Let $\tilde{G}$ be the resulting graph, and let $G' = \tilde{G} \setminus \mathcal{T}$. We let $\mathcal{T}' = \bigcup_{t \in \mathcal{T}} S_t$ be the set of terminals for the new graph G'. Then $|\mathcal{T}'| = C$, and each vertex in $\mathcal{T}'$ has exactly one edge incident to it. We now apply Theorem 2 to $(G', \mathcal{T}')$ to obtain a sparsifier H'. In our final step, for each $t \in \mathcal{T}$, we unify all vertices in the set S_t in graph H' into a single vertex t. Let H be this final sparsifier. Clearly, $|V(H)| = C^{O(\log \log C)}$. We now show that H is a quality-$(2\eta^*)$ sparsifier for G.

Let D be any set of demands on $\mathcal{T}$, and let F be the routing of these demands in graph G with congestion $\eta = \eta(G, D)$. Then the routing F naturally defines a set D' of demands over the vertices of $\mathcal{T}'$. For each pair $(v_e, v_{e'}) \in \mathcal{T}'$ of vertices, the demand $D'(v_e, v_{e'})$ is the total flow on all flow-paths in F that start at e and terminate at e'. Flow F also gives a routing of this new set D' of demands in graph G' with congestion η. Since H' is a flow sparsifier for G', there is a routing F' of the demands in D' in graph H' with congestion at most η. This routing induces a routing of the set D of demands in graph H with congestion at most η.

The other direction is proved similarly: if D is any set of demands in $\mathcal{T}$, and F is the routing of these demands in H with congestion $\eta = \eta(H, D)$, then F defines a set D' of demands on the vertices of $\mathcal{T}'$ exactly as before. Flow F then induces a routing of the set D' of demands in graph H' with congestion at most η. Since H' is a quality-$(2\eta^*)$ sparsifier for G', there is a routing F' of the set D' of demands in graph G' with congestion at most $\eta \cdot 2\eta^*$. Flow F' then induces a routing of the set D of demands in graph G with congestion at most $\eta \cdot 2\eta^*$.

Finally, consider the general case, where the edge capacities $c_e \geq 1$ are no longer required to be integral, and may not be bounded by C. For each edge $e \in E$, if $c_e > C$, then we set $c_e = C$. It is easy to see that this transformation does not affect the values $\eta(G, D)$ for demand sets D, since all flow in the network must traverse one of the edges incident to the terminals. Next, we define new edge capacities, by setting $c'_e = \left\lceil \frac{2\eta^*}{\epsilon} c_e \right\rceil$. Let G' be the resulting graph. Notice that for each edge e, $\frac{2\eta^*}{\epsilon} c_e \leq c'_e \leq \frac{2\eta^*}{\epsilon} c_e + 1 \leq \frac{2\eta^* + \epsilon}{\epsilon} c_e$.

Consider some set D of demands defined over the set $\mathcal{T}$ of terminals. Let F be the routing of D in graph G', whose congestion is $\eta(G', D)$. Consider the same flow F in graph G. The congestion caused by F on each edge e of G is bounded by $\frac{F(e)}{c_e} \leq \frac{F(e)}{c'_e} \cdot \frac{2\eta^* + \epsilon}{\epsilon}$. Therefore, $\eta(G, D) \leq \eta(G', D) \cdot \frac{2\eta^* + \epsilon}{\epsilon}$ for all D.

Similarly, given any set D of demands, let F be the routing of D in graph G, whose congestion is $\eta(G, D)$. Consider the same flow F in graph G'. The congestion on each edge e in G' is bounded by $\frac{F(e)}{c'_e} \leq \frac{F(e)}{c_e} \cdot \frac{\epsilon}{2\eta^*}$. Therefore, $\eta(G', D) \leq \eta(G, D) \cdot \frac{\epsilon}{2\eta^*}$ for all D. We conclude that for all D, $\frac{\epsilon}{2\eta^* + \epsilon} \eta(G, D) \leq \eta(G', D) \leq \frac{\epsilon}{2\eta^*} \eta(G, D)$.

Let H' be a quality-$(2\eta^*)$ sparsifier for graph G', and let H be the graph obtained from H' by multiplying all edge capacities by the factor of $\frac{\epsilon}{2\eta^*}$. Then for any set D of demands:

$$\eta(H, D) = \frac{2\eta^*}{\epsilon}\eta(H', D) \le \frac{2\eta^*}{\epsilon}\eta(G', D) \le \eta(G, D);$$

and

$$\begin{aligned}
\eta(H, D) &= \frac{2\eta^*}{\epsilon}\eta(H', D) \\
&\ge \frac{1}{2\eta^*}\frac{2\eta^*}{\epsilon}\eta(G', D) \\
&\ge \frac{1}{2\eta^* + \epsilon}\eta(G, D).
\end{aligned}$$

We conclude that H is a quality-$(2\eta^* + \epsilon)$-sparsifier for G, and $|V(H)| = C^{O(\log \log C)}$.

D   Proof Omitted from Section 4.1

D.1   Proof of Claim 3

For simplicity, we build two new graphs H and H', as follows. Sub-divide every edge $e \in E'$ with a vertex v_e in both G and G', and let $\mathcal{T}' = \{v_e \mid e \in E'\}$. Let H be the sub-graph of the resulting graph obtained from G, induced by $S \cup \mathcal{T}'$, and let H' be the sub-graph of the corresponding graph obtained from G', induced by $S' \cup \mathcal{T}'$. Clearly, H' is a legal contracted graph for H, and from the definition of well-linkedness, H' is α-well-linked for $\mathcal{T}'$. We only need to prove that H is $\alpha/3$-well-linked for $\mathcal{T}'$. For any subset Z of vertices in either H or H', let $\mathcal{T}_Z = Z \cap \mathcal{T}'$.

Assume for contradiction that H' is not α-well-linked for $\mathcal{T}'$, and let (X, Y) be the violating partition of $V(H)$, that is, $|E_H(X, Y)| < \frac{\alpha}{3}\min\{|\mathcal{T}_X|, |\mathcal{T}_Y|\}$. We construct a partition (X', Y') of $V(H')$, such that $|E_{H'}(X', Y')| < \alpha \cdot \min\{|\mathcal{T}_{X'}|, |\mathcal{T}_{Y'}|\}$, contradicting the fact that H' is α-well-linked for $\mathcal{T}'$.

Let $\mathcal{C}_S \subseteq \mathcal{C}$ be the collection of all clusters C, with $v_C \in S'$. In order to construct the partition (X', Y') of $V(H')$, we start with the partition (X, Y) of $V(H)$, and process all clusters $C \in \mathcal{C}_S$ one-by-one. For each such cluster, we move all vertices of C either to X or to Y. Once we process all clusters in $\mathcal{C}_S$, we will obtain obtain a partition $(\tilde{X}, \tilde{Y})$ of $V(H)$, that will naturally define a partition (X', Y') of $V(H')$.

Consider some cluster $C \in \mathcal{C}_S$, and partition the edges in $\text{out}_H(C)$ into four subsets, E_X, E_Y, E_{XY}, E_{YX}, as follows. For each edge $e = (u, v) \in \text{out}_H(C)$, with $u \in C$, $v \notin C$, if both $u, v \in X$, then we add e to E_X; if both $u, v \in Y$, we add e to E_Y; if $u \in X$, $v \in Y$, then we add e to E_{XY}, and otherwise we add it to E_{YX}. If $|E_X| + |E_{XY}| \le |E_Y| + |E_{YX}|$, then we move all vertices of C to Y, and otherwise we move them to X. Let $E_C \subseteq E_H(X, Y)$ be the subset of the edges in the cut (X, Y), with both endpoints in C, that is, $E_C = E_H(X \cap C, Y \cap C)$.

Assume w.l.o.g. that $|E_X| + |E_{XY}| \le |E_Y| + |E_{YX}|$, and so we have moved the vertices of C to Y. The only new edges that have been added to the cut are the edges of E_X. Since set C is $1/3$-well-linked in H, $|E_X| \le 3|E_C|$. We charge the edges in E_C for the edges in E_X. The charge to every edge of E_C is at most 3, and since the edges of E_C have both endpoints inside the cluster C, we will never charge them again.

26

Once we process all super-nodes v_C in this fashion, we obtain a partition $(\tilde{X}, \tilde{Y})$ of $V(H)$, where for every cluster $C \in \mathcal{C}_S$, all vertices of C belong to either $\tilde{X}$ or $\tilde{Y}$. This cut naturally defines a partition (X', Y') of the vertices of $V(H')$, where $v_C \in X'$ iff $C \subseteq \tilde{X}$. From the above discussion, $|E_{H'}(X', Y')| = |E_H(\tilde{X}, \tilde{Y})| \leq 3|E_H(X, Y)|$. Moreover, since the terminals in $\mathcal{T}'$ do not belong to any cluster $C \in \mathcal{C}_S$, the partitions of the terminals of $\mathcal{T}'$ induced by the cuts (X, Y) in H and (X', Y') in H' are identical. Therefore, $|E_{H'}(X', Y')| < \alpha \min\{|\mathcal{T}_{X'}|, |\mathcal{T}_{Y'}|\}$, a contradiction.

D.2   Proof of Claim 4

We set up the following two flow networks. For the first flow network, we start with the graph G. We add a source s, and connect it with a directed edge to every vertex $v \in \mathcal{T}'$. For every edge $e \in E'$, we sub-divide e by adding a vertex z_e to it, and connect z_e to the sink t with a directed edge. We set the capacity of every edge in this network to be 3η, except for the edges leaving s or entering t, whose capacities are set to 1. Let N_1 denote the resulting flow network. In order to show the existence of the set $\mathcal{P}$ of paths, it is enough to show that the value of the maximum flow in network N_1 is $|\mathcal{T}'|$. For each edge $e \in E(N_1)$, we denote by $c(e)$ its capacity, and for each cut (X, Y) in the network, we denote by $c(X, Y)$ the total capacity of edges connecting the vertices of X to the vertices of Y.

The second network, N_2, is constructed similarly, except that we use the contracted graph G', instead of the graph G. Specifically, we start with graph G', add a source s and a sink t. Source s connects with a directed edge to every vertex $v \in \mathcal{T}'$. As before, we sub-divide every edge $e \in E'$ with a vertex z_e, and connect z_e to the sink t. The capacities of all edges in N_2 are η, except for the edges that leave the source s or enter the sink t, whose capacities are set to 1. Observe that the existence of the set $\mathcal{P}'$ of paths in graph G' guarantees that there is a flow of value $|\mathcal{T}'|$ in network N_2. For each edge $e \in E(N_2)$, we denote by $c'(e)$ its capacity in N_2, and for each cut (X, Y) in the network, we denote by $c'(X, Y)$ the total capacity of edges connecting the vertices of X to the vertices of Y.

It is now enough to prove that there is a flow of value $|\mathcal{T}'|$ in network N_1. Assume this is not the case. Then there is an s-t cut (X, Y) in network N_1, with $c(X, Y) < |\mathcal{T}'|$. We show that there is an s-t cut (X', Y') in network N_2, with $c'(X', Y') < |\mathcal{T}'|$, contradicting the existence of the set $\mathcal{P}$ of paths.

We consider the super-nodes $v_C \in V(G')$ one-by-one. For each such super-node v_C, we move all vertices of C either to X or to Y. The final cut, $(\tilde{X}, \tilde{Y})$ will naturally define an s-t cut (X', Y') in network N_2, and we will show that its capacity is less than $|\mathcal{T}'|$.

Consider some super-node $v_C \in V(G')$. Recall that we are guaranteed that $\mathcal{T}' \cap C = \emptyset$, so $\text{out}_{N_1}(C) = \text{out}_G(C)$. Let $E_C = E_G(C \cap X, C \cap Y)$. We partition the edges of $\text{out}_G(C)$ into four subsets, E_X containing edges with both endpoints in X, E_Y containing edges with both endpoints in Y, E_{XY} containing edges $e = (u, v)$ with $u \in X \cap C$, $v \in Y \setminus C$, and E_{YX} containing the remaining edges. If $|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|$, then we move all vertices of C to Y, and otherwise we move them to X.

Assume w.l.o.g. that $|E_X| + |E_{XY}| \leq |E_Y| + |E_{YX}|$, so we have moved the vertices of C to Y. Since cluster C is $1/3$-well-linked in graph G, $|E_C| \geq |E_X|/3$. We charge the edges of E_C for the edges of E_X, with the charge to every edge of E_C being at most 3. Observe that none of the edges in $E_C \cup E_X$ is incident on the source s or the sink t.

Let $(\tilde{X}, \tilde{Y})$ be the cut obtained after processing all super-nodes $v_C \in V(G')$. Let $E_1(X, Y) \subseteq E(X, Y)$ be the subset of edges incident on the source s or on the sink t in the original cut, and let

$E_1(\tilde{X}, \tilde{Y})$ be the subset of edges incident on the source s or on the sink t in the new cut. Since none of the clusters C we have considered contained vertices of $\mathcal{T}$, or vertices z_e for $e \in E'$, $|E_1(X, Y)| = |E_1(\tilde{X}, \tilde{Y})|$. Let $E_2(X, Y) = E(X, Y) \backslash E_1(X, Y)$, and similarly, let $E_2(\tilde{X}, \tilde{Y}) = E(\tilde{X}, \tilde{Y}) \backslash E_1(\tilde{X}, \tilde{Y})$. From the above discussion, $|E_2(\tilde{X}, \tilde{Y})| \leq 3|E_2(X, Y)|$. Recall that the capacities of all edges in $E_2(X, Y)$ and $E_2(\tilde{X}, \tilde{Y})$ are 3η, while the capacities of edges in $E_1(X, Y)$ are 1, and we have assumed that $c(X, Y) = |E_1(X, Y)| + 3\eta|E_2(X, Y)| < |\mathcal{T}'|$.

Cut $(\tilde{X}, \tilde{Y})$ naturally defines an s-t cut (X', Y') in network N_2. The capacity of this cut in network N_2 is $c'(X', Y') = |E_1(X, Y)| + \eta|E_2(\tilde{X}, \tilde{Y})| \leq |E_1(X, Y)| + 3\eta|E_2(X, Y)| < |\mathcal{T}'|$, a contradiction.