

Vertex Sparsification in Trees

Gramoz Goranci¹ and Harald Räcke²

¹ Faculty of Computer Science, University of Vienna, Vienna, Austria

`gramoz.goranci@univie.ac.at`

² Institut für Informatik, Technische Universität München, Garching, Germany

`raecke@in.tum.de`

Abstract. Given an unweighted tree $T = (V, E)$ with terminals $K \subset V$, we show how to obtain a 2-quality vertex flow and cut sparsifier H with $V_H = K$. We prove that our result is essentially tight by providing a $2 - o(1)$ lower-bound on the quality of any cut sparsifier for stars.

In addition we give improved results for quasi-bipartite graphs. First, we show how to obtain a 2-quality flow sparsifier with $V_H = K$ for such graphs. We then consider the other extreme and construct exact sparsifiers of size $O(2^k)$, when the input graph is unweighted.

Keywords: Graph sparsification · Vertex flow sparsifiers · Trees

1 Introduction

Graph sparsification is a technique to deal with large input graphs by “compressing” them into smaller graphs while preserving important characteristics, like cut values, graph spectrum etc. Its algorithmic value is apparent, since these smaller representations can be computed in a preprocessing step of an algorithm, thereby greatly improving performance.

Cut sparsifiers [4] and spectral sparsifiers [19] aim at reducing the number of edges of the graph while approximately preserving cut values and graph spectrum, respectively. These techniques are used in a variety of fast approximation algorithms, and are instrumental in the development of nearly linear time algorithms.

In vertex sparsification [6, 9, 10, 12, 14, 16, 18], apart from reducing the number of edges, the goal is also to reduce the number of vertices of a graph. In such setting, one is given a large graph $G = (V, E, c)$, together with a relatively small subset of terminals $K \subseteq V$. The goal is to shrink the graph while preserving properties involving the terminals. For example, in *Cut Sparsification* one wants to construct a graph $H = (V_H, E_H, c_H)$ (with $K \subseteq V_H$) such that H preserves mincuts between terminals up to some approximation factor q (the *quality*).

Hagerup et al. [9] introduced this concept under the term *Mimicking Networks*, and focused on constructing a (small) graph H that maintains mincuts exactly. They showed that one can obtain H with $O(2^{2^k})$ vertices, where $k = |K|$. Krauthgamer and Rika [13] and Khan and Raghavendra [11] independently proved that $2^{\Omega(k)}$ vertices are required for some graphs if we want to preserve mincuts exactly.

Moitra [16] analyzed the setting where the graph H is as small as possible, namely $V_H = K$. Under this condition, he obtained a quality $O(\log k / \log \log k)$ cut sparsifier. A lower bound of $\Omega(\sqrt{\log k / \log \log k})$ was presented by Makarychev and Makarychev [15]. A strictly stronger notion than a cut sparsifier, is a *flow sparsifier* that aims at (approximately) preserving all multicommodity flows between terminals. The upper bound of [16] also holds for this version, but the lower bound is slightly stronger: $\Omega(\sqrt{\log k / \log \log k})$.

Due to the lower bounds on the quality of sparsifiers with $V_H = K$, the recent focus has been on obtaining better guarantees with slightly larger sparsifiers. Chuzhoy [7] obtained a constant quality flow sparsifier of size $C^{O(\log \log C)}$, where C is the total weight of the edges incident to terminal nodes. Andoni et al. [3] obtained quality of $(1 + \varepsilon)$ and size $O(\text{poly}(k/\varepsilon))$ for *quasi-bipartite* graphs, i.e., graphs where the terminals form an independent set. This is interesting since these graphs serve as a lower bound example for Mimicking Networks, i.e., in order to obtain an exact sparsifier one needs size at least $2^{\Omega(k)}$.

In this paper we study flow and cut sparsifiers for trees. Since, for tree networks it is immediate to obtain a sparsifier of size $O(k)$ and quality 1, we consider the problem of designing flow and cut sparsifiers with $V_H = K$ as in the original definition of Moitra. In Sect. 2 we show how to design such a flow sparsifier for unweighted trees with quality 2. In Sect. 3 we prove that this result is essentially tight by establishing a lower bound. Concretely, we prove that even for unweighted stars it is not possible to obtain cut sparsifiers with quality $2 - o(1)$.

As a further application of our techniques, we apply them to quasi-bipartite graphs. We first obtain a 2-quality flow sparsifier with $V_H = K$ for such graphs. In addition we explore the other extreme and construct exact sparsifiers of size $O(2^k)$, if the input graph is unweighted. This shows that even though quasi-bipartite graphs serve as lower bound instances for Mimicking Networks they are not able to close the currently large gap between the upper bound of $O(2^k)$ and the lower bound of $2^{\Omega(k)}$ on the size of Mimicking Networks.

Finally we obtain hardness results for the problem of deciding whether a graph H is a sparsifier for a given unweighted tree T . We prove that this problem is co- \mathcal{NP} -hard for cut sparsifiers, based on Chekuri et al. [5]. For flow sparsifiers we show that for a single-source version, where the sparsifier has to preserve flows in which all demands share a common source, the problem is co- \mathcal{NP} -hard. Due to space limitations the hardness results have been deferred to the full version.

1.1 Preliminaries

Let $G = (V, E, c)$ be an undirected graph with terminal set $K \subset V$ of cardinality k , where $c : E \rightarrow \mathbb{R}^+$ assigns a non-negative capacity to each edge. We present two different ways to sparsify the number of vertices in G .

Let $U \subset V$ and $S \subset K$. We say that a cut $(U, V \setminus U)$ is S -separating if it separates the terminal subset S from its complement $K \setminus S$, i.e., $U \cap K$ is either S or $K \setminus S$. The cutset $\delta(U)$ of a cut $(U, V \setminus U)$ represents the edges that have one endpoint in U and the other one in $V \setminus U$. The cost $\text{cap}_G(\delta(U))$ of a cut $(U, V \setminus U)$ is the sum over all capacities of the edges belonging to the cutset.

We let $\text{mincut}_G(S, K \setminus S)$ denote the S -separating cut of the minimum cost in G . A graph $H = (V_H, E_H, c_H)$, $K \subset V_H$ is a *vertex cut sparsifier* of G with *quality* $q \geq 1$ if: $\forall S \subset K$, $\text{mincut}_G(S, K \setminus S) \leq \text{mincut}_H(S, K \setminus S) \leq q \cdot \text{mincut}_G(S, K \setminus S)$.

We say that a (*multi-commodity*) flow f is a *routing* of the demand function d , if for every terminal pair (x, x') it sends $d(x, x')$ units of flow from x to x' . The *congestion* of an edge $e \in E$ incurred by the flow f is defined as the ratio of the total flow sent along the edge to the capacity of that edge, i.e., $f(e)/c(e)$. The *congestion of the flow* f for routing demand d is the maximum congestion over all edges in G . We let $\text{cong}_G(d)$ denote the minimum congestion over all flows. A graph $H = (V_H, E_H, c_H)$, $K \subset V_H$ is a *vertex flow sparsifier* of G with *quality* $q \geq 1$ if for every demand function d , $\text{cong}_H(d) \leq \text{cong}_G(d) \leq q \cdot \text{cong}_H(d)$.

We use the following tools about sparsifiers throughout the paper.

Lemma 1 [14]. *If $H = (V_H, E_H, c_H)$, $V_H = K$ is a vertex flow sparsifier of G , then the quality of H is $q = \text{cong}_G(d_H)$, where $d_H(x, x') := c_H(x, x')$ for all terminal pairs (x, x') .*

Let G_1 and G_2 be graphs on *disjoint* set of vertices with terminals $K_1 = \{s_1, \dots, s_k\}$ and $K_2 = \{t_1, \dots, t_m\}$, respectively. In addition, let $\phi(s_i) = t_i$, for all $i = 1, \dots, \ell$, be a one-to-one correspondence between some subset of K_1 and K_2 . The ϕ -merge (or 2-sum) of G_1 and G_2 is the graph G with terminal set $K = K_1 \cup \{t_{\ell+1}, \dots, t_m\}$ formed by identifying the terminals s_i and t_i for all $i = 1, \dots, \ell$. This operation is denoted by $G := G_1 \oplus_\phi G_2$.

Lemma 2 ([3], Merging). *Let $G = G_1 \oplus_\phi G_2$. Suppose G'_1 and G'_2 are flow sparsifiers of quality q_1 and q_2 for G_1 and G_2 , respectively. Then $G' = G'_1 \oplus_\phi G'_2$ is a flow sparsifier of quality $\max\{q_1, q_2\}$ for G .*

Lemma 3 (Convex Combination of Sparsifiers). *Let $H_i = (V^*, E_i, c_i)$, $i = 1, \dots, m$ with $K \subset V^*$ be vertex flow sparsifiers of G . In addition, let $\alpha_1, \alpha_2, \dots, \alpha_m$ be convex multipliers corresponding to H_i 's such that $\sum_i \alpha_i = 1$. Then the graph $H' = \sum_i \alpha_i \cdot H_i$ is a vertex flow sparsifier for G .*

2 Improved Vertex Flow Sparsifiers for Trees

In this section we show that given an unweighted tree $T = (V, E)$, $K \subset V$, we can construct a flow sparsifier H only on the terminals, i.e., $V(H) = K$, with quality at most 4. We then further improve the quality to 2. The graph H has the nice property of being a convex combination of trees.

We obtain the quality of 4 by combining the notion of probabilistic mappings due to Andersen and Feige [2] and a duality argument due to Räcke [17]. Our result then immediately follows using as a black-box an implicit result of Gupta [8]. We note that a direct application of the Transfer Theorem due to Andersen and Feige [2] does not apply, since their interchangeability argument relies on arbitrary capacities and lengths.

Let $w : E \rightarrow \mathbb{R}_{\geq 0}$ be a function which assigns non-negative values to edges which we refer to as *lengths*. Given a tree $T = (V, E, w)$ we use $d_w : V \times V \rightarrow \mathbb{R}_{\geq 0}$

to denote the shortest path distance induced by the edge length w . A *0-extension* of a tree $T = (V, E)$, $K \subset V$ is a retraction $f : V \rightarrow K$ with $f(x) = x$, for all $x \in K$, along with another graph $H = (K, E_H)$ such that $E_H = \{(f(u), f(v)) : (u, v) \in E\}$. The graph H is referred to as a *connected 0-extension* if in addition we require that $f^{-1}(x)$ induces a connected component in T .

Given a graph $G = (V, E)$, we let \mathcal{P} be a collection of multisets of E , which will be usually referred to as paths. A mapping $M : E \rightarrow \mathcal{P}$ maps every edge e to a path $P \in \mathcal{P}$. This mapping can be alternatively represented as a non-negative square matrix M of dimension $|E| \times |E|$, where $M(e', e)$ is the number of times edge e lies on the path $M(e')$. Let \mathcal{M} denote the collection of mappings M . If we associate to each mapping $M \in \mathcal{M}$ a convex multiplier λ_M , the resulting mapping is referred to as a *probabilistic mapping*.

Connected 0-Extension Embedding on Trees. Suppose we are given a tree $T = (V, E)$, $K \subset V$ and a connected 0-extension (H, f) , where $H = (K, E_H)$ and f is a retraction. Given an edge $(u, v) \in E$ from T , we can use the retraction f to find the edge $(f(u), f(v))$ in H (if u and v belong to different components). Since this edge is not an edge of the original tree T , we need a way to map it back to T in order to be consistent with our definition of mappings. The natural thing to do is to take the unique shortest path between $f(u)$ and $f(v)$ in T . Denote by $S_{u,v}^T$ all the edges in the shortest path between u and v in T . Then, we let $M_{H,f}(u, v) = S_{f(u),f(v)}^T$ be the mapping $M_{H,f} : E \rightarrow \mathcal{P}$ induced by (H, f) .

Let \mathcal{H} be the family of all connected 0-extensions for T , which are also trees. We then define the collection of mappings \mathcal{M} for T by $\{M_{H,f} : H \in \mathcal{H}\}$.

Capacity Mappings. Given a tree $T = (V, E, c)$, $c : E \rightarrow \mathbb{R}^+$ and a connected 0-extension (H, f) , the *load* of an edge $e \in E$ under (H, f) is $\text{load}_f(e) = \sum_{e'} M_{H,f}(e', e) \cdot c(e')$. The *expected load* of an edge $e \in E$ under a probabilistic mapping is $\sum_i \lambda_i \text{load}_{f_i}(e)$.

Distance Mappings. Given a tree $T = (V, E, w)$, $w : E \rightarrow \mathbb{R}^+$ and a connected 0-extension (H, f) , the *mapped length* of an edge $e' = (u', v') \in E$ under (H, f) is $d_w(f(u'), f(v')) = \sum_e M_{H,f}(e', e) \cdot w(e)$. The *expected mapped length* of an edge $e' = (u', v') \in E$ under a probabilistic mapping is $\sum_i \lambda_i d_w(f_i(u'), f_i(v'))$.

With the above definitions in mind, for some given tree $T = (V, E, c)$, we can find a flow sparsifiers that is a convex combination of connected 0-extensions using the following linear program, and its dual.

$$\begin{array}{ll}
 \min \alpha & \min \beta \\
 \text{s.t. } \forall e \sum_i \lambda_i \cdot \text{load}_{f_i}(e) \leq \alpha \cdot c(e) & \text{s.t. } \forall i \sum_e w(e) \cdot \text{load}_{f_i}(e) \geq \beta \quad (*) \\
 \sum_i \lambda_i \geq 1 & \sum_e w(e) \cdot c(e) \leq 1 \\
 \forall i \quad \lambda_i \geq 0. & \forall e \quad w(e) \geq 0.
 \end{array}$$

Next, we re-write the dual constraints of type (*) as follows:

$$\begin{aligned}
 \sum_e w(e) \text{load}_{f_i}(e) &= \sum_e w(e) \sum_{e'} M_{H,f_i}(e', e) \cdot c(e') \\
 &= \sum_{e'} c(e') (\sum_e M_{H,f_i}(e', e) \cdot w(e)) = \sum_{e'=(u',v')} c(e') \cdot d_w(f_i(u'), f_i(v')).
 \end{aligned}$$

Using this re-formulation and a few observations, the dual is equivalent to:

$$\max_{w \geq 0} \min_i \sum_{e=(u,v)} c(e) \cdot d_w(f_i(u), f_i(v)) / \sum_e w(e) \cdot c(e). \quad (1)$$

For the unweighted case $c(e) = 1$, we can make use of the following lemma:

Lemma 4 [8, Lemma 5.1]. *Given a tree $T = (V, E, w)$, $K \subset V$, we can find a connected 0-extension f such that $\sum_{e=(u,v)} d_w(f(u), f(v)) \leq 4 \cdot \sum_e w_e$.*

The above lemma tells us that optimal value of (1) is bounded by 4. This implies that the optimal value of the dual is bounded by 4, and by strong duality, the optimal value of the primal is also bounded by 4. The latter implies that T admits a 4-quality vertex sparsifier of size k .

2.1 Obtaining Quality 2

Next we show how to bring down the quality of flow sparsifiers on trees to 2. We give a direct algorithm that constructs a flow sparsifier and unlike in the previous subsection, it does not rely on the interchangeability between distances and capacities. We first consider trees where terminals are the only leaf nodes, i.e., $L(T) = K$. Later we show how to extend the result to arbitrary trees.

To convey some intuition, we start by presenting the deterministic version of our algorithm. We maintain at any point of time a partial mapping f -setting $f(v) = \perp$, when $f(v)$ is still undefined, but producing a valid connected 0-extension when the algorithm terminates. Note that $f(x) = x$, for all $x \in K$. Without loss of generality, we may assume that the tree is rooted at some non-terminal vertex and the child-parent relationships are defined. The algorithm works as follows: it repeatedly picks a non-terminal v farthest from the root and maps it to one of its children c , i.e., $f(v) = f(c)$ ¹ (we refer to such procedure as Algorithm 1. This process results in a flow sparsifier that is a connected 0-extension.

Unfortunately, the quality of the sparsifier produced by the above algorithm can be very poor. To see this, consider an unweighted star graph $S_{1,k}$, where leaves are the terminal vertices and the center is the non-terminal vertex v . Any connected 0-extension of $S_{1,k}$ is a new star graph $S_{1,k-1}$ lying on the terminals, where the center is the terminal x with $f(v) = x$. Now, consider a demand function d that sends a unit flow among all edges in $S_{1,k-1}$. Clearly, d can be feasibly routed in $S_{1,k-1}$. But routing d in $S_{1,k}$ gives a load of at least $k - 1$ along the edge (x, v) , and thus the quality of $S_{1,k-1}$ is at least $k - 1$ (Lemma 1).

One way to improve upon the quality is to map the non-terminal v uniformly at random to one of the terminals. We can equivalently view this as taking convex combination over all possible connected 0-extensions of $S_{1,k}$. By Lemma 3 we know that such a convex combination gives us another flow sparsifier for $S_{1,k}$, and it can be checked that the quality of such a sparsifier improves to 2. Surprisingly, we show that applying this trivial random-mapping of non-terminals in

¹ Alternatively, one can view this step as contracting an arbitrary child-edge of v .

trees with terminals as leaves leads to a flow sparsifier H which is a random connected 0-extension and achieves similar guarantees. We refer to such procedure as Algorithm 2.

To compute the quality of H as a flow sparsifier for T , we need to bound the congestion of every edge of T incurred by the embedding of H into T . This embedding routes the capacity of every terminal edge (x, x') in H along the (unique) shortest paths between leaves x and x' in T . First, we crucially observe that without loss of generality, it suffices to bound the load of the edges incident to the terminals, i.e., edges incident to leaf vertices. To see this, let (u, v) be an edge among non-terminals in T , with v being the parent of u . Now, when embedding H into T , we know that the demands among all terminal pairs that lie in the subtree $T(u)$ rooted at u *cannot* incur any load on the edge (u, v) , as these terminal shortest paths do not use this edge. Thus, we can safely replace the subtree $T(u)$ with some dummy terminal and perform the analysis as before.

First, we study edge loads under deterministic connected 0-extensions. Let $e = (x, v)$ be the edge incident to $x \in K$, m_x denote the level of x in T and $\{x, v_{m_x-1}, \dots, v_0\}$ be the set of vertices belonging to the shortest path between x and the root $r = v_0$ in T . Given a connected 0-extension f_i output by Algorithm 1, we say that x is *expanded* up to the ℓ -th level if $f_i(v_j) = x$, for all $j \in \{m_x, \dots, \ell\}$. This leads to the following lemma.

Lemma 5. *Let $e = (x, v)$ be the edge incident to $x \in K$, (H_i, f_i) be a connected 0-extension and recall that empty sum is defined as 0. If x is expanded up to the ℓ -th level, then the load of e under (H_i, f_i) is $\text{load}_{f_i}(e) \leq 1 + \sum_{j=\ell}^{m_x-1} (c_j - 1)$, $\ell \in \{m_x, \dots, 0\}$, where c_j denotes the number of children of non-terminal v_j in T .*

Let $I_\ell^x = \{(H_i, f_i)\}$ be the set of connected 0-extensions output by Algorithm 1 where x is expanded up to the ℓ -th level. We observe that the edge e has the same load regardless of which element of I_ℓ^x we choose. Thus, for any $(H_i, f_i) \in I_\ell^x$, we can write $\text{load}_\ell(e) = \text{load}_{f_i}(e)$.

Now, we study the expected edge loads under the random connected 0-extension output by Algorithm 2. Let N be the number of all different connected 0-extensions that can be output by Algorithm 1. If by Z_ℓ^x we denote the event that x is expanded up to the ℓ -th level, then it follows that the expected load $\mathbb{E}[\text{load}_f(e)]$ of $e = (x, v)$ under (H, f) is

$$\sum_{i=1}^N \text{load}_{f_i}(e) / N = \sum_{\ell=0}^{m_x} \frac{\# \text{ of } f_i \text{'s s.t. } Z_\ell^x}{N} \cdot \text{load}_\ell(e) = \sum_{\ell=0}^{m_x} \mathbb{P}[Z_\ell^x] \cdot \text{load}_\ell(e). \quad (2)$$

Since in Algorithm 2 all non-terminals are mapped independently of each other, we obtain $\mathbb{P}[Z_\ell^x] = (1 - 1/c_{\ell-1}) \prod_{j=\ell}^{m_x-1} 1/c_j$, $\ell \in \{m_x, \dots, 1\}$ (recall that the empty product is defined as 1). Further, observe that $\mathbb{P}[Z_0^x] = 1 / \prod_{j=0}^{m_x-1} c_j$. Plugging the probabilities and Lemma 5 in (2), we get that $\mathbb{E}[\text{load}_f(e)]$ is

$$\frac{1}{\prod_{j=0}^{m_x-1} c_j} \left(1 + \sum_{j=0}^{m_x-1} (c_j - 1) \right) + \sum_{\ell=1}^{m_x} (1 - 1/c_{\ell-1}) \prod_{j=\ell}^{m_x-1} \frac{1}{c_j} \left(1 + \sum_{j=\ell}^{m_x-1} (c_j - 1) \right).$$

Next, we rewrite the above as A/B , where $B = \prod_{j=0}^{m_x-1} c_j$ and A is given by

$$1 + \sum_{j=0}^{m_x-1} (c_j - 1) + \sum_{\ell=1}^{m_x-1} (c_{\ell-1} - 1) \prod_{j=0}^{\ell-2} c_j \left(1 + \sum_{j=\ell}^{m_x-1} (c_j - 1) \right) + (c_{m_x-1} - 1) \prod_{j=0}^{m_x-2} c_j.$$

The following lemma simplifies the middle expression of A .

Lemma 6. *For any positive integers $\{c_0, \dots, c_{m_x-1}\}$ and $m_x \geq 3$,*

$$\sum_{\ell=1}^{m_x-1} (c_{\ell-1} - 1) \prod_{j=0}^{\ell-2} c_j \left(1 + \sum_{j=\ell}^{m_x-1} (c_j - 1) \right) = (c_{m_x-1} + 1) \prod_{\ell=0}^{m_x-2} c_\ell - \sum_{\ell=0}^{m_x-1} (c_\ell - 1) - 2.$$

Proof. Let $P(m_x - 1)$ be the left-hand side expression in the statement of the lemma. We proceed by induction on m_x . For the base case $m_x = 3$, it is easy to argue that the claim is valid. If we assume that the lemma holds true for $m_x - 1$, then we get that:

$$\begin{aligned} P(m_x) &= \sum_{\ell=1}^{m_x-1} (c_{\ell-1} - 1) \prod_{j=0}^{\ell-2} c_j \left(1 + \sum_{j=\ell}^{m_x-1} (c_j - 1) + (c_{m_x} - 1) \right) \\ &\quad + (c_{m_x-1} - 1) \prod_{j=0}^{m_x-2} c_j \left((c_{m_x} - 1) + 1 \right) \\ &= \sum_{\ell=1}^{m_x-1} (c_{\ell-1} - 1) \prod_{j=0}^{\ell-2} c_j \left(1 + \sum_{j=\ell}^{m_x-1} (c_j - 1) \right) \\ &\quad + (c_{m_x} - 1) \sum_{\ell=1}^{m_x} (c_{\ell-1} - 1) \prod_{j=0}^{\ell-2} c_j + (c_{m_x-1} - 1) \prod_{j=0}^{m_x-2} c_j. \end{aligned} \tag{3}$$

Note that the following expression is a simple telescoping series:

$$\sum_{\ell=1}^{m_x} (c_{\ell-1} - 1) \prod_{j=0}^{\ell-2} c_j = \prod_{\ell=0}^{m_x-1} c_\ell - 1. \tag{4}$$

Plugging this into Eq. (3) and using induction hypothesis gives:

$$\begin{aligned} P(m) &= (c_{m_x-1} + 1) \prod_{\ell=0}^{m_x-2} c_\ell - \sum_{\ell=0}^{m_x-1} (c_\ell - 1) - 2 + (c_{m_x} - 1) \left(\prod_{\ell=0}^{m_x-1} c_\ell - 1 \right) \\ &\quad + (c_{m_x-1} - 1) \prod_{j=0}^{m_x-2} c_j = (c_{m_x} + 1) \prod_{\ell=0}^{m_x-1} c_\ell - \sum_{\ell=0}^{m_x} (c_\ell - 1) - 2. \end{aligned}$$

This completes the induction step, and hence the proof of the lemma. \square

Now, plugging the above lemma in A we get that $A = 2B - 1$. Thus, $\mathbb{E}[\text{load}_f(e)] = (2B - 1)/B \leq 2$. Since we consider only unweighted trees, it follows that the expected congestion of every edge is also bounded by 2. Taking the maximum over all edge congestions yields the following:

Lemma 7. *Given a tree $T = (V, E)$, $K \subset V$, $L(T) = K$, there is a 2-quality flow sparsifier H , which is a convex combination over connected 0-extensions.*

Derandomization. Next we show that Algorithm 2 can be easily derandomized. We obtain a deterministic algorithm that runs $O(n+k^2\alpha(2k))$ time and gives the same guarantees as in Lemma 7, where $\alpha(\cdot)$ is the inverse Ackermann function.

We first give an $O(n)$ time preprocessing step. For a tree $T = (V, E)$, $K \subset V$, $L(T) = K$, we repeatedly contract edges incident to non-terminals of degree 2 in T . When all such non-terminals are deleted from T , our new tree can have at most $2k$ vertices. Note that this tree exactly preserves all flows among terminals.

Now, we crucially observe that in the flow sparsifier H output by Algorithm 2, the capacity between any two terminals x and x' is exactly the probability that x and x' are connected under the random mapping f . We next show that this probability can be computed efficiently.

Let (x, x') be any terminal pair, $\text{lca}(x, x')$ denote their lowest common ancestor in T and r denote the level of $\text{lca}(x, x')$ in T . Moreover, let $V_r^x = \{x, v_{m_x-1}, \dots, v_r\}$, $v_r = \text{lca}(x, x')$, be the set of vertices belonging to the shortest path between x and the $\text{lca}(x, x')$. Similarly, define $V_r^{x'} = \{x', v'_{m_{x'}-1}, \dots, v_r\}$. Since in Algorithm 2 all non-terminals are mapped independently of each other, we obtain

$$\begin{aligned} \mathbb{P}[(f(x), f(x')) \in E_H] &= 2 \cdot \mathbb{P}[f(v_r) = x] \cdot \mathbb{P}[f(v) = x, \forall v \in V_{r-1}^x] \\ &\cdot \mathbb{P}[f(v') = x', \forall v' \in V_{r-1}^{x'}] = \frac{2}{c_r} \cdot \prod_{j=r}^{m_x-1} \frac{1}{c_j} \prod_{j=r}^{m_{x'}-1} \frac{1}{c'_j}. \end{aligned} \quad (5)$$

where c_j, c'_j are the number of children of the non-terminal v_j, v'_j , respectively.

The above expression suggest that one should build an efficient data-structure for T that answers queries of the form “What is the product of the elements associated with vertices along the path from x to x' in T ?”. This problem is known as *The Tree Product Query* problem. For an arbitrary tree with n vertices, Alon and Schieber [1] show that in order to answer each Tree Product query in at most $O(\alpha(n))$ steps, an $O(n)$ preprocessing time is sufficient.

Now we are ready to give our deterministic procedure. We first apply our initial preprocessing step in $O(n)$ time. Since the resulting tree has at most $2k$ vertices, it takes $O(k)$ time to preprocess the tree such that every internal vertex knows the number of its children. Next, using $O(k)$ preprocessing, we build a data-structure for the Tree Product Query problem. Now, for every terminal pair (x, x') we can compute in $O(\alpha(2k))$ time the capacity of (x, x') in H from the Tree product query between x and x' and Eq. (5). Since there are at most $O(k^2)$ terminal pairs, we get a running time of $O(n+k^2\alpha(2k))$. The correctness is immediate from the above observations.

Extension to Arbitrary Trees. The above algorithm can be extended to arbitrary trees (deferred to the full version). This leads to the following theorem:

Theorem 1. *Given an unweighted tree $T = (V, E)$, $K \subset V$, there exists a 2-quality flow sparsifier H . Moreover, H can be viewed as a convex combination over connected 0-extensions of T .*

3 Lower Bound

In this section we present a $2 - o(1)$ lower bound on the quality of any cut sparsifier for a star graph. Since previous lower bounds relied on non-planar graph instances, this is the first non-trivial lower bound for arbitrary cut sparsifiers on planar graphs. The result extends to the stronger notion of flow sparsifiers.

The main idea behind our approach is to exploit the symmetries of the star graph. We observe that these symmetries induce other symmetries on the cut structure of the graph. This simplifies the structure of an optimal cut-sparsifier.

Let $G = (K \cup \{v\}, E)$, be an unweighted star with k terminals. Let π' be any permutation of K . We extend π' to a permutation π of $K \cup \{v\}$ by setting $\pi(x) = \pi'(x), \forall x \in K$ and $\pi(v) = v$. Now, for any $U \subset K \cup \{v\}$ and any such a permutation π , we use the symmetry $\text{cap}_G(\delta(U)) = \text{cap}_G(\delta(\pi(U)))$. The latter implies that for any $S \subset K$, $\text{mincut}_G(S, K \setminus S) = \text{mincut}_G(\pi(S), K \setminus \pi(S))$.

For a cut sparsifier H of quality q for G , we show that $\pi(H)$, i.e., the graph obtained by renaming all vertices of H according to permutation π , is also a cut sparsifier of quality q for G . Indeed, for any $S \in K$, $\text{cap}_{\pi(H)}(\delta(S)) = \text{cap}_H(\delta(\pi^{-1}(S))) \geq \text{mincut}_G(\pi^{-1}(S), K \setminus \pi^{-1}(S)) = \text{mincut}_G(S, K \setminus S)$. Symmetrically, one can show that $\text{cap}_{\pi(H)}(\delta(S)) \leq q \cdot \text{mincut}_G(S, K \setminus S)$.

Lemma 8. *A convex combination of any two cut sparsifiers with the same quality gives a new cut sparsifier with the same or better quality.*

Lemma 9. *For the star graph G defined as above, there exists an optimum cut sparsifier H , which is a complete graph with uniform edges-weights.*

Proof. First, we observe by Lemma 8 that if we have two cut sparsifiers with the same quality, taking their convex combination gives a new cut sparsifier with the same or better quality. Suppose we are given some optimum cut sparsifier H' . We can generate $k!$ different cut sparsifiers by considering all possible permutations π as defined above. By the above arguments, for each π , we know that $\pi(H')$ is also an optimum cut sparsifier. Taking the convex combination over $k!$ such sparsifiers, we obtain a complete graph H with uniform edge-weights. \square

Lemma 10. *If H is uniform weighted complete graph that is an optimum cut sparsifier for the star graph G and k even, the edge weight must be at least $2/k$.*

Proof. By definition, H must dominate the terminal cut that has $k/2$ vertices on one side. The minimum value of such a cut in G is $k/2$. The number of edges that cross such a cut in H is $k^2/4$. Since H has uniform edge-weights, this gives that the edge weight must be at least $2/k$. \square

Theorem 2. *Let $G = (K \cup \{v\}, E)$ be an unweighted star with k terminals. Then, there is no cut sparsifier H that achieves quality better than $2 - o(1)$.*

Proof. By the above lemmas, we can assume without loss of generality that H is a complete graph with uniform edge-weights, where this edge weight is at least $2/k$. Hence, a cut that has a singleton terminal vertex on one side has capacity $2(k-1)/k = 2(1-1/k)$ in H but it has minimum cut value 1 in G . The latter implies that the quality of H must be at least $2(1-1/k)$. \square

4 Improved Results for Quasi-Bipartite Graphs

In this section, we present two new tradeoffs for flow sparsifiers in quasi-bipartite graphs. For this family of graphs, Andoni et al. [3] show how to obtain flow sparsifier with very good quality and moderate size. Specifically, they obtain an $(1 + \varepsilon)$ -quality flow sparsifier of size $\tilde{O}(k^7/\varepsilon^3)$. In the original definition of flow sparsifiers, Leighton and Moitra [14] studied the version where sparsifiers lie only on the terminals, i.e., $V_H = K$. For this restricted setting, we obtain a flow sparsifier of quality 2.

Exact Cut Sparsifier (a.k.a Mimicking Networks) were introduced by Hagerup et al. [9]. In their work they show that general graphs admit exact cut sparsifiers of size doubly exponential in k . As a second result, we show that unit weighted quasi-bipartite graphs admit an exact flow sparsifier of size 2^k .

A graph G with terminals K is *quasi-bipartite* if the non-terminals form an independent set. Throughout this section we assume w.l.o.g. that we are given a bipartite graph with terminals lying on one side and non-terminals in the other (this can be achieved by subdividing terminal-terminal edges).

A 2-Quality Flow Sparsifier of Size k . Assume we are given an unweighted bipartite graph G with terminals K . The crucial observation is that we can view G as taking union over stars, where each non-terminal is the center connected to some subset of terminals. Lemma 2 allows us to study these stars independently. Then, for every such star, we apply Lemma 7 to obtain a flow sparsifier only on the terminals belonging to that star. Finally, we merge the resulting sparsifiers and construct a sparsifier H with $V(H) = K$ by another application of Lemma 7. Since the quality of every star in isolation is 2 or better, H is also a 2-quality flow sparsifier.

We note that Lemma 7 only works for unweighted trees. There is an easy extension that gives a similar lemma for weighted stars.

Lemma 11. *Let $G = (K \cup \{u\}, E, c)$ be a weighted star with k terminals. Then G admits a 2-quality flow sparsifier H of size k .*

Applying the decomposition and merging lemma similarly to the unweighted case leads to the following theorem:

Theorem 3. *Let $G = (V, E, c)$ with $K \subset V$ be a weighted quasi-bipartite graph. Then G admits a 2-quality flow sparsifier H of size k .*

An Exact Flow Sparsifier of Size 2^k . In what follows it will be convenient to work with an equivalent definition for Flow Sparsifiers. Let $\lambda_G(d)$ denote the maximum fraction of *concurrent flow* when routing demand d among terminals in graph G . Then $H = (V_H, E_H, c_H)$ with $K \subset V_H$ is a *flow sparsifier* of G with quality $q \geq 1$ if for all demand functions d , $\lambda_G(d) \leq \lambda_H(d) \leq q \cdot \lambda_G(d)$.

The high level idea of our approach is to create “types” for non-terminals and then merge all non-terminals of the same type into a single non-terminal (i.e., add infinity capacity among all non-terminals of the same type). The main difficulty is to define the right types and show that the merging does not affect the

multi-commodity flow structure among the terminals. A similar approach was developed by Andoni et al. [3], but their guarantees applies only to approximate flow sparsifier.

We start by defining types. We say that two non-terminals u, v are of the same type if they are incident to the same subset of terminals. Non-terminals of the same type form groups. Note that a non-terminal belongs to a unique group. The *size* of the group is the number of non-terminals belonging to that group. Since the set of non-terminals is an independent set, by Lemma 2, we can construct sparsifiers for each group independently. Our final sparsifier is obtained by merging the sparsifiers over all groups. By another application of Lemma 2, if the sparsifiers of the groups are exact flow sparsifiers, then the final sparsifier is also an exact flow sparsifier for the original graph.

Next, if we replace each group by a single non-terminal, then the size guarantee of the final sparsifier follows from the fact that there are at most 2^k different subsets of terminals. Below we formalize the merging operation within groups.

Let $G_i = (K' \cup \{v_1, \dots, v_{n_i}\}, E_i, c)$ be a group of size $n_i \geq 2$, where $E_i = \{\{v_j, x\} : j \in \{1, \dots, n_i\}, x \in K'\}$, $K' \subseteq K$ and $c(e) = 1, e \in E_i$. We get:

Lemma 12. *Let G_i with $K' \subset V(G_i)$ be a group of size $n_i \geq 2$ defined as above. Then G_i can be replaced by a star $H_i = (K' \cup \{v_1\}, E_{H_i}, c_{H_i})$ with edge weights $c_{H_i}(e) = n_i$, for all $e \in E_{H_i}$, and which preserves exactly all multicommodity flows between terminals from K' .*

Taking the union over all sparsifiers H_i leads to the following theorem:

Theorem 4. *Let $G = (V, E)$ with $K \subset V$ be a unit weighted quasi-bipartite graph. Then G admits an exact flow sparsifier H of size at most 2^k .*

Proof (Lemma 12). First, observe that we can think of H_i as adding infinity capacity edges between non-terminals in G_i . Then merging into a single non-terminal is done by simply adding edge weights incident to the same terminal. More precisely, let $E_{H_i} = \{(v_r, v_s) : r, s = 1, \dots, n_i, r \neq s\}$. Then, we can assume that $H_i = (K' \cup \{v_1, \dots, v_{n_i}\}, E_i \cup E_{H_i}, c_{H_i})$ where $c_{H_i}(e) = c(e)$ if $e \in E_i$ and $c_{H_i}(e) = \infty$ if $e \in E_{H_i}$.

Since we can route every feasible demand from G_i in H_i even without using the infinity-capacity edges, it is immediate that for any demand function d , $\lambda_{H_i}(d) \geq \lambda_{G_i}(d)$. Thus, we only need to show that $\lambda_{H_i}(d) \leq \lambda_{G_i}(d)$. To achieve this, we will use the dual to the maximum concurrent flow problem (i.e., the Fractional Sparsest Cut Problem). The dual problem is the following²:

$$\begin{aligned}
 \min \quad & \sum_{j=1}^{n_i} \sum_{x \in K'} \ell_{v_j x} \\
 \text{s. t.} \quad & \ell_{sv_j} + \ell_{v_j t} \geq \delta_{st} \quad \forall \{s, t\} \in \binom{K'}{2}, \forall j \in \{1, \dots, n_i\} \\
 & \sum_{\{s, t\} \in \binom{K'}{2}} d_{st} \delta_{st} \geq 1 \\
 & \ell_e \geq 0, \quad \delta_{st} \geq 0.
 \end{aligned} \tag{6}$$

² Note that the dual requires that δ_{st} is at most the length of the shortest s - t path. In our scenario this is always a 2-hop path. Hence, the above formulation is correct.

Let d be an arbitrary demand function. Moreover, let $\{\ell_e, \delta_{st}\}$ be an optimal solution of value $\lambda_{G_i}(d)$ for the LP in Eq. (6), where δ_{st} is the shortest-path distance induced by the length assignment ℓ . We first modify this solution and get a new feasible solution with the same cost and a certain structure that we will later exploit.

The modification works as follows. For every terminal we create a set of edges incident to that terminal. Then, within each set, we replace the length of each edge by the total *average* length of the group. Specifically, for every $x \in K'$, let $E_x = \{(v_j, x) : j = 1, \dots, n_i\}$ be the set of edges incident to x .

The new edge lengths are defined as follows: $\tilde{\ell}_{v_j x} = \sum_{e \in E_x} \ell_e / n_i, \forall x \in K', \forall j = 1, \dots, n_i$. Let $\tilde{\delta}_{st}$ be the new shortest-path distance induced by the length assignment $\tilde{\ell}$. In order for $\{\tilde{\ell}_e, \tilde{\delta}_{st}\}$ to be feasible, we need to show that $\tilde{\delta}$ dominates δ , i.e., $\tilde{\delta}_{st} \geq \delta_{st}$, for every pair $s, t \in K'$. Indeed, since edge lengths within groups are the same, we get that for every pair $s, t \in K'$:

$$\begin{aligned} \tilde{\delta}_{st} &= \tilde{\ell}_{sv_1} + \tilde{\ell}_{v_1 t} = \frac{1}{n_i} \sum_{e \in E_s} \ell_e + \frac{1}{n_i} \sum_{e \in E_t} \ell_e = \frac{1}{n_i} \sum_{j=1}^{n_i} (\ell_{sv_j} + \ell_{v_j t}) \\ &\geq \min_{j \in \{1, \dots, n_i\}} \{\ell_{sv_j} + \ell_{v_j t}\} \geq \delta_{st}. \end{aligned}$$

Additionally, observe that the new solution has the same optimal value, namely $\lambda_{G'_i}^*(d) = \sum_{j=1}^{n_i} \sum_{x \in K'} \ell_{v_j x} = \sum_{j=1}^{n_i} \sum_{x \in K'} \tilde{\ell}_{v_j x}$. Hence, we can assume without loss of generality that an optimal solution satisfies: $\tilde{\ell}_{v_1 x} = \dots = \tilde{\ell}_{v_{n_i} x}, \forall x \in K'$. Now, we add edges (v_i, v_j) to G_i and set $\tilde{\ell}_{v_i v_j} = 0$, for all $i, j = 1, \dots, n_i$. Note that shortest-path distances $\tilde{\delta}_{st}$ do not change by this modification. Therefore, by adding these zero edge lengths between the non-terminals, we still get an optimum solution $\{\tilde{\ell}_e, \tilde{\delta}_{st}\}$ for the LP in (6).

Finally, let us define the dual problem for the star H_i :

$$\begin{aligned} \min \quad & \sum_{j=1}^{n_i} \sum_{x \in K'} \ell_{v_j x} \\ \text{s. t.} \quad & \sum_{e \in P_{st}} \ell_e \geq \delta_{st} \quad \forall \{s, t\} \in \binom{K'}{2}, \forall s\text{-}t \text{ paths on } E \cup E_{H_i} \\ & \sum_{\{s, t\} \in \binom{K'}{2}} d_{st} \delta_{st} \geq 1 \\ & \ell_e \geq 0, \quad \delta_{st} \geq 0, \quad \forall e \in E_{H_i} \quad \ell_e = 0. \end{aligned} \tag{7}$$

It follows from above that $\{\tilde{\ell}_e, \tilde{\delta}_{st}\}$ is a feasible solution for the LP in (7). Hence, $\lambda_{H_i}(d) \leq \lambda_{G_i}(d)$, what we were after. \square

References

1. Alon, N., Schieber, B.: Optimal preprocessing for answering on-line product queries. Technical report, Tel Aviv University (1987)
2. Andersen, R., Feige, U.: Interchanging distance and capacity in probabilistic mappings. CoRR, [arXiv:abs/0907.3631](https://arxiv.org/abs/0907.3631) (2009)
3. Andoni, A., Gupta, A., Krauthgamer, R.: Towards $(1 + \varepsilon)$ -approximate flow sparsifiers. In: Proceedings of the 25th SODA, pp. 279–293 (2014)

4. Benczúr, A.A., Karger, D.R.: Approximating s - t minimum cuts in $\tilde{O}(n^2)$ time. In: Proceedings of the 28th STOC, pp. 47–55 (1996)
5. Chekuri, C., Shepherd, F.B., Oriolo, G., Scutellà, M.G.: Hardness of robust network design. *Networks* **50**(1), 50–54 (2007)
6. Cheung, Y.K., Goranci, G., Henzinger, M.: Graph minors for preserving terminal distances approximately - lower and upper bounds. In: Proceedings of the 43rd ICALP, pp. 131:1–131:14 (2016)
7. Chuzhoy, J.: On vertex sparsifiers with steiner nodes. In: Proceedings of the 44th STOC, pp. 673–688 (2012)
8. Gupta, A.: Steiner points in tree metrics don't (really) help. In: Proceedings of the 12th SODA, pp. 220–227 (2001)
9. Hagerup, T., Katajainen, J., Nishimura, N., Ragde, P.: Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.* **57**(3), 366–375 (1998)
10. Kamma, L., Krauthgamer, R., Nguyen, H.L.: Cutting corners cheaply, or how to remove steiner points. *SIAM J. Comput.* **44**(4), 975–995 (2015)
11. Khan, A., Raghavendra, P.: On mimicking networks representing minimum terminal cuts. *Inf. Process. Lett.* **114**(7), 365–371 (2014)
12. Krauthgamer, R., Nguyen, H.L., Zondiner, T.: Preserving terminal distances using minors. *SIAM J. Discrete Math.* **28**(1), 127–141 (2014)
13. Krauthgamer, R., Rika, I.: Mimicking networks and succinct representations of terminal cuts. In: Proceedings of the 24th SODA, pp. 1789–1799 (2013)
14. Leighton, F.T., Moitra, A.: Extensions and limits to vertex sparsification. In: Proceedings of the 42nd STOC, pp. 47–56 (2010)
15. Makarychev, K., Makarychev, Y.: Metric extension operators, vertex sparsifiers and lipschitz extendability. In: Proceedings of the 51th FOCS, pp. 255–264 (2010)
16. Moitra, A.: Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In: Proceedings of the 50th FOCS (2009)
17. Räcke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: Proceedings of the 40th STOC, pp. 255–264 (2008)
18. Räcke, H., Shah, C., Täubig, H.: Computing cut-based hierarchical decompositions in almost linear time. In: Proceedings of the 25th SODA, pp. 227–238 (2014)
19. Spielman, D.A., Teng, S.: Spectral sparsification of graphs. *SIAM J. Comput.* **40**(4), 981–1025 (2011)