Towards $(1+\varepsilon)$-Approximate Flow Sparsifiers*

Alexandr Andoni† Microsoft Research
Anupam Gupta‡ CMU and MSR
Robert Krauthgamer§ Weizmann Institute

Abstract

A useful approach to "compress" a large network G is to represent it with a *flow-sparsifier*, i.e., a small network H that supports the same flows as G, up to a factor $q \geq 1$ called the quality of sparsifier. Specifically, we assume the network G contains a set of k terminals T, shared with the network H, i.e., $T \subseteq V(G) \cap V(H)$, and we want H to preserve all multicommodity flows that can be routed between the terminals T. The challenge is to construct H that is small.

These questions have received a lot of attention in recent years, leading to some known tradeoffs between the sparsifier's quality q and its size $|V(H)|$. Nevertheless, it remains an outstanding question whether every G admits a flow-sparsifier H with quality $q = 1+\varepsilon$, or even $q = O(1)$, and size $|V(H)| \leq f(k,\varepsilon)$ (in particular, independent of $|V(G)|$ and the edge capacities).

Making a first step in this direction, we present new constructions for several scenarios:

- Our main result is that for quasi-bipartite networks G, one can construct a $(1+\varepsilon)$-flow-sparsifier of size $\mathrm{poly}(k/\varepsilon)$. In contrast, exact ($q = 1$) sparsifiers for this family of networks are known to require size $2^{\Omega(k)}$.
- For networks G of bounded treewidth w, we construct a flow-sparsifier with quality $q = O(\log w / \log\log w)$ and size $O(w \cdot \mathrm{poly}(k))$.
- For general networks G, we construct a *sketch* $sk(G)$, that stores all the feasible multicommodity flows up to factor $q = 1+\varepsilon$, and its size (storage requirement) is $f(k,\varepsilon)$.

*A full version of this paper is submitted to arXiv [AGK13].

†Email: andoni@microsoft.com

‡Work supported in part by NSF awards CCF-0964474 and CCF-1016799, US-Israel BSF grant #2010426, and by a grant from the CMU-Microsoft Center for Computational Thinking. Email: anupamg@cs.cmu.edu

§Work supported in part by the Israel Science Foundation (grant #897/13), the US-Israel BSF (grant #2010418), and by the Citi Foundation. Email: robert.krauthgamer@weizmann.ac.il

1 Introduction

A powerful tool to deal with big graphs is to "compress" them by reducing their size — not only does it reduce their storage requirement, but often it also reveals opportunities for more efficient graph algorithms. Notable examples in this context include the cut and spectral sparsifiers of [BK96, ST04], which have had a huge impact on graph algorithmics. These sparsifiers reduce the number of edges of the graph, while preserving prominent features such as cut values and Laplacian spectrum, up to approximation factor $1+\varepsilon$. This immediately improves the runtime of graph algorithms that depend on the number of edges, at the expense of $(1+\varepsilon)$-approximate solutions. Such sparsifiers reduce only the number of *edges*, but it is natural to wonder whether more is to be gained by reducing the number of *nodes* as well. This vision — of "node sparsification" — already appears, say, in [FM95].

One promising notion of node sparsification is that of *flow or cut sparsifiers*, introduced in [HKNR98, Moi09, LM10], where we have a *network* (a term we use to denote edge-capacitated graphs) G, and the goal is to construct a smaller network H that supports the same flows as G, up to a factor $q \geq 1$ called the quality of sparsifier H. Specifically, we assume the network G contains a set T of k *terminals* shared with the network H, i.e., $T \subseteq V(G) \cap V(H)$, and we want H to preserve all multicommodity flows that can be routed between the terminals T. A somewhat simpler variant is a *cut sparsifier*, which preserves the single-commodity flow from every set $S \subset T$ to its complement $T \setminus S$, i.e., a minimum-cut in G of the terminals bipartition $T = S \cup (T \setminus S)$. Throughout, we consider undirected networks (although some of the results apply also for directed networks), and unless we say explicitly otherwise, flow and cut sparsifiers refer to their node versions, i.e., networks on few nodes that support (almost) the same flow.

The main question is: *what tradeoff can one achieve between the quality of a sparsifier and its size?* This question has received a lot of attention in recent years. In particular, if the sparsifier is only supported on T (achieves minimal size), one can guarantee quality

$q \leq O(\frac{\log k}{\log\log k})$ [Moi09, LM10, CLLM10, EGK+10, MM10]. On the other hand, with this minimal size, the (worst-case) quality must be $q \geq \tilde{\Omega}(\sqrt{\log k})$ [LM10, CLLM10, MM10], and thus a significantly better quality cannot be achieved without increasing the size of the sparsifier. The only other result for flow sparsifiers, due to [Chu12], achieves a constant quality sparsifiers whose size depends on the capacities in the original graph. (Her results give flow sparsifiers of size $C^{O(\log\log C)}$; here C is the total capacity of edges incident to terminals and hence may be $\Omega(nk)$ even for unit-capacity graphs.) For the simpler notion of cut sparsifiers, there are known constructions at the other end of the tradeoff. Specifically, one can achieve exact (quality $q = 1$) cut sparsifier of size 2^{2^k} [HKNR98, KRTV12], however, the size must still be at least $2^{\Omega(k)}$ [KRTV12, KR13] (for both cut and flow sparsifiers).

Taking cue from edge-sparsification results, and the above lower bounds, it is natural to focus on small sparsifiers that achieve quality $1 + \varepsilon$, for small $\varepsilon \geq 0$. Note that for flow sparsifiers, we do not know of any bound on the size of the sparsifier that would depend only on k (and $1/\varepsilon$), but not on n or edge capacities. In fact, we do not even know whether it is possible to represent the sparsifier *information theoretically* (i.e., by a small-size sketch), let alone by a graph.

1.1 Results Making a first step towards constructing high-quality sparsifiers of small size, we present constructions for several scenarios:

- Our main result is for *quasi-bipartite graphs*, i.e., graphs where the non-terminals form an independent set (see [RV99]), and we construct for such networks a $(1+\varepsilon)$-flow-sparsifier of size $\text{poly}(k/\varepsilon)$. In contrast, exact ($q = 1$) sparsifiers for this family of networks are known to require size $2^{\Omega(k)}$ [KRTV12, KR13]. (See Theorem 6.2.)
- For general networks G, we construct a *sketch* $sk(G)$, that stores all the feasible multicommodity flows up to factor $q = 1 + \varepsilon$, and has size (storage requirement) of $f(k, \varepsilon)$ words. This implies an affirmative answer to the above information-theoretic question on existence of flow sparsifiers, and raises the hope for a $(1+\varepsilon)$-flow-sparsifier of size $f(k, \varepsilon)$. (See Theorem 3.1.)
- For networks G of bounded treewidth w, we construct a flow-sparsifier with quality $q = O(\frac{\log w}{\log\log w})$ and size $O(w \cdot \text{poly}(k))$. (See Theorem 7.4.)
- Series-parallel networks admit an exact (quality 1) flow sparsifier with $O(k)$ vertices. (See Theorem 7.2.)

1.2 Techniques Perhaps our most important contribution is the introduction of the three techniques listed below, and indeed, one can view our results from the prism of these three rather different approaches. In particular, applying these three techniques to quasi-bipartite graphs yields $(1+\varepsilon)$-quality sparsifiers whose sizes are (respectively) doubly-exponential, exponential, and polynomial in k/ε.

1. *Clumping:* We first "discretize" the set of (almost) all possible multi-commodity demands into a finite set, whose size depends only on k/ε, and then partition the graph vertices into a small number of "clusters", so that clumping each cluster into a single vertex still preserves one (and eventually all) of the discretized demands. The idea of clumping vertices was used in the past to obtain exact (quality 1) cut sparsifiers [HKNR98]. Flow-sparsifiers require, in effect, to preserve all metrics between the terminals rather than merely all inter-terminal cut metrics, and requires new ideas.
2. *Splicing/Composition:* Our Splicing Lemma shows that it is enough for the sparsifier to maintain flows routed using paths that do not contain internally any terminals. Our Composition Lemma shows that for a network obtained by gluing two networks along some subset of their terminals, gluing the respective sparsifiers (in the same manner) gives us a sparsifier for the glued network. These lemmas enable us to do "surgery" on networks, to decompose and recompose them, so that we find good sparsifiers on smaller networks and then combine them together without loss of quality.
3. *Sampling:* This technique samples parts of the graph, while preserving the flows approximately. The main difficulty is to determine correct sampling probabilities (and correlations). This is the technical heart of the paper, and we outline its main ideas in Section 1.3.

We hope they will inspire ulterior constructions of high-quality flow sparsifiers for general graphs. The clumping techniques give information-theoretic bounds on flow-sparsification, and the splicing/composition approach proves useful for sparsification of bounded treewidth and series-parallel graphs (beyond what can be derived using their flow/cut gaps from known cut sparsifiers).

1.3 Outline of Our Sampling Approach A classic approach to obtain an *edge*-sparsifier [Kar94, BK96, SS11] is to sample the edges of the graph and rescale appropriately. Here, we outline instead how to sam-

ple the *vertices* of the graph to obtain a small flow-sparsifier. We outline our main idea on quasi-bipartite graphs (where the non-terminals form an independent set), considering for simplicity the (simpler) question of *cut* sparsifiers, where we want to construct a smaller graph G' that preserves the minimum cut between every bipartition of terminals $T = S \cup (T \setminus S)$. The main idea is to sample a small number of non-terminals v, keeping only their incident edges, and rescaling the corresponding capacities. For a fixed bipartition $T = S \cup \bar{S}$, we can write the value of the min-cut as

$$\alpha_{S,\bar{S}} = \sum_{v \notin T} \min\Big\{ \sum_{s \in S} c_{sv}, \sum_{t \in \bar{S}} c_{vt} \Big\}. \tag{1.1}$$

(Here c_{xy} is the capacity of the edge xy.) Suppose we assign each non-terminal v with some sampling probability p_v, then sample the non-terminals using these probabilities, letting I_v be an indicator variable for whether v was sampled. Then, for sampled v's we re-normalize the capacities on incident edges by $1/p_v$, i.e., the new capacities are $c'_{v,t} = c_{v,t}/p_v$ for all $t \in T$ (non-sampled v's are dropped). The new value of the min-cut in the sparsifier G' is

$$\alpha'_{S,\bar{S}} = \sum_{v \notin T} I_v/p_v \cdot \min\Big\{ \sum_{s \in S} c_{sv}, \sum_{t \in \bar{S}} c_{vt} \Big\}. \tag{1.2}$$

This classical estimator is unbiased, and hence each min-cut $\alpha_{S,\bar{S}}$ is preserved *in expectation*.

The main challenge now is to prove that the above random sum concentrates around its expectation for "small" values of p_v. For example, consider setting all p_v equal, say to $\text{poly}(k)/|V|$. Even if all $c_{s,v} \in \{0,1\}$ (i.e., all edges in E have unit capacity), due to the min operation, it is possible that only very few terms in the summation in Eqn. (1.1) have nonzero contribution to $\alpha_{S,\bar{S}}$, and are extremely unlikely to be sampled.

Our general approach is to employ *importance sampling*, where each p_v is related to v's contribution to the sum, namely $\min\{\sum_{s \in S} c_{sv}, \sum_{t \in \bar{S}} c_{vt}\}$. Applying this directly is hard — since that minimum depends on the bipartition $S \cup \bar{S}$, whereas p_v cannot. Instead, we exploit the fact that for any bipartition, we can estimate

$$\alpha'_{S,\bar{S}} \geq \max_{s \in S, t \in \bar{S}} \sum_v I_v/p_v \cdot \min\{c_{sv}, c_{vt}\} \geq \tfrac{1}{k^2} \alpha'_{S,\bar{S}}, \tag{1.3}$$

and hence arguing about the sum in Eqn. (1.3) should be enough for bounding the variance. Following this reasoning through, it turns out that a good choice is

$$p_v = M \cdot \max_{s \neq t} \frac{\min\{c_{sv}, c_{vt}\}}{\sum_{v'} \min\{c_{sv'}, c_{v't}\}}, \tag{1.4}$$

where $M = \text{poly}(k/\varepsilon)$ is an over-sampling factor. The underlying intuition of Eqn. (1.4) is that, replacing the max with a "correct" choice of $s \in S, t \in \bar{S}$, the denominator is just the entire potential contribution to the sum in Eqn. (1.3), and hence these p_v values can be used as importance sampling probabilities for the sum in Eqn. (1.2). Moreover, we prove that this setting of p_v allows for a high-probability concentration bound in the sum from Eqn. (1.2), and thus sampling $\text{poly}(k/\varepsilon)$ vertices suffices for the purpose of taking a union bound over all 2^k bipartitions.

So far we have described the approach for obtaining cut sparsifiers, but in fact we prove that the exact same approach works for obtaining flow sparsifiers as well. There are more issues that we need to take care in this generalized setting. First, we need to bound the "effective" number of demand vectors. Second, the flow does not have a simple closed-form formula like (1.1), so upper and lower bounds need to be proved by analyzing separately (the concentration of) the optimal value of the flow LP and of its dual.

2 Preliminaries

A *k-terminal network* is an edge-capacitated graph $G = (V, E, c)$ with a subset $T \subseteq V$ of k terminals. We will be interested only in *terminal flows*, i.e., flows that start and end only at the terminal vertices of G. Define $\mathcal{D}(G)$, the *demand polytope of G* as the set of all demand vectors $\mathbf{d}$ that are supported only on terminal-pairs, and admit a feasible multicommodity-flow in G, formally,

$$\mathcal{D}(G) := \{\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}} : \text{ demand } \mathbf{d} \text{ can be routed in } G\}, \tag{2.5}$$

where we denote $\mathbb{R}_+ := \{x \in R : x \geq 0\}$ and $\binom{T}{2} := \{S \subseteq T : |S| = 2\}$. Throughout, we assume G is connected.

LEMMA 2.1. *$\mathcal{D}(G)$ is a polytope, and is down-monotone.*

Proof. Let $\mathcal{P}_{ij}$ be the set of paths between terminals i and j. Consider the extended demand polytope $\mathcal{D}_{ext}(G)$ with variables d_{ij} for all $\{i,j\} \in \binom{T}{2}$, and f_P for each $P \in \cup_{i,j} \mathcal{P}_{ij}$.

$$\begin{aligned} \textstyle\sum_{P \in \mathcal{P}_{ij}} f_P &= d_{ij} \\ \textstyle\sum_{i,j} \sum_{P \in \mathcal{P}_{ij} : e \in P} f_P &\leq c_e \\ f_P, d_{ij} &\geq 0. \end{aligned}$$

This polytope captures all the feasible terminal flows, and hence all the routable demands between the terminals of G. The projection of this polytope $\mathcal{D}_{ext}(G)$ onto the variables d is exactly $\mathcal{D}(G)$; hence the latter is

also a polytope.[1] Finally, the down-monotonicity of the polytope follows from the downward-feasibility of flows, in turn due to the lack of lower-bounds on the flows on edges. ■

Dual linear program for concurrent flow. For any demand vector $\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}}$, we denote the *concurrent flow* problem (inverse of the congestion) by

$$\lambda_G(\mathbf{d}) := \sup\{\lambda \geq 0 : \ \lambda\mathbf{d} \in \mathcal{D}(G)\}.$$

This is well-defined because $\vec{0} \in \mathcal{D}(G)$. The following well-known lemma writes $\lambda_G(\mathbf{d})$ by applying linear programming duality to multicommodity flow, see e.g. [LR99, Shm97, Moi09].

LEMMA 2.2. *$\lambda_G(\mathbf{d})$ can be computed via the linear program* (LP1) *which has "edge-length" variables ℓ_e for edges $e \in E$ and "distance" variable $\delta_{uv} = \delta_{vu}$ for terminal pairs $\{s,t\} \in \binom{T}{2}$.*

3 A Data Structure for Multicommodity Flows

We present a data structure that "maintains" $\mathcal{D}(G)$ within approximation factor $1+\varepsilon$. More precisely, we preprocess the terminal network G into a data structure whose storage requirement depends only on k and ε (but not on $n = |V(G)|$). Given a query $\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}}$, this data structure returns an approximation to $\lambda_G(\mathbf{d})$ within factor $1+\varepsilon$ (without further access to G). The formal statement appears in Theorem 3.1. We assume henceforth that $0 < \varepsilon < 1/8$.

An approximate polytope. For each commodity $\{i,j\} \in \binom{T}{2}$, let L_{ij} be the maximum flow of commodity ij alone (i.e., as a single-commodity flow) in G. Discretize the set $\mathcal{D}(G)$ defined in (2.5) by defining the subset

$$\mathcal{D}_\varepsilon^{\text{discrete}} := \Big\{\mathbf{d} \in \mathcal{D}(G) : \text{ every nonzero } d_{ij} \text{ is a power of } 1+\varepsilon \text{ in the range } [\varepsilon/k^2 \cdot L_{ij}, L_{ij}]\Big\}.$$

The range upper bound L_{ij} (which is not really necessary, as it follows from $\mathbf{d} \in \mathcal{D}(G)$), immediately implies that

(3.6)
$$|\mathcal{D}_\varepsilon^{\text{discrete}}| \leq \left(1 + \tfrac{1}{\varepsilon}\log_{1+\varepsilon} k\right)^{\binom{k}{2}} \leq \left(\tfrac{O(1)}{\varepsilon}\log k\right)^{k^2}.$$

LEMMA 3.1. *The convex hull $\text{conv}(\mathcal{D}_\varepsilon^{discrete})$ is down-monotone, namely, if $\mathbf{d} \in \text{conv}(\mathcal{D}_\varepsilon^{discrete})$ and $0 \leq \widehat{\mathbf{d}} \leq \mathbf{d}$, then also $\widehat{\mathbf{d}} \in \text{conv}(\mathcal{D}_\varepsilon^{discrete})$.*

[1] As a aside, we can write $\mathcal{D}_{ext}(G)$ more compactly using edge-flow variables $f^{ij}(e)$ instead of path variables f_P; we omit such standard optimizations here.

Proof. Consider first the special case where $\widehat{\mathbf{d}}$ is obtained from $\mathbf{d}$ by scaling the coordinates in some subset $S \subseteq \binom{T}{2}$ by a scalar $0 \leq \beta < 1$. Write $\mathbf{d}$ as a convex combination of some vectors $\mathbf{d}_j \in \mathcal{D}_\varepsilon^{\text{discrete}}$, say $\sum_j \alpha_j \mathbf{d}_j$, where $\alpha_j > 0$ and $\sum_j \alpha_j = 1$. Let $\widehat{\mathbf{d}}_j$ be the vector obtained from $\mathbf{d}_j$ by zeroing all the coordinates in S, and observe it is also in $\mathcal{D}_\varepsilon^{\text{discrete}}$. Now write

$$\widehat{\mathbf{d}} = \sum_j \alpha_j[\beta\mathbf{d}_j + (1-\beta)\widehat{\mathbf{d}}_j] = \sum_j \alpha_j\beta\mathbf{d}_j + \sum_j \alpha_j(1-\beta)\widehat{\mathbf{d}}_j,$$

and observe the right-hand side is a convex combination of vectors in $\mathcal{D}_\varepsilon^{\text{discrete}}$, which proves the aforementioned special case. The general case follows by iterating this special-case argument several times. ■

The data structure. The next theorem expresses the space requirement of an algorithm in machine words, assuming every machine word can store $\log k$ bits and any single value L_{ij} (either exactly or within accuracy of $1+\varepsilon/2$ factor). This holds, in particular, when edge capacities in the graph G are integers bounded by $n = |V(G)|$, and a word has $2\log n$ bits.

THEOREM 3.1. *For every $0 < \varepsilon < 1/8$ there is a data structure that provides a $(1+\varepsilon)$-approximation for multicommodity flows, using space $\left(\frac{O(1)}{\varepsilon}\log k\right)^{k^2}$ and query time $O(\frac{1}{\varepsilon}k^2 \log k)$.*

Proof. We present a data structure achieving approximation $1 + O(\varepsilon)$; the theorem would then follow by scaling $\varepsilon > 0$ appropriately. The data structure stores the set $\mathcal{D}_\varepsilon^{\text{discrete}}$, using a dictionary (such as a hash table) to answer membership queries in time $O(k^2)$, the time required to read a single vector. It additionally stores all the values L_{ij}. (We assume these values can be stored exactly; the case of $1+\varepsilon/2$ approximation follows by straightforward modifications.)

Given a query $\mathbf{d}$, the algorithm first computes $\beta = \min_{i,j\in T}\{L_{ij}/d_{ij}\}$. We thus have that $\beta k^{-2} \leq \lambda_G(\mathbf{d}) \leq \beta$, because the commodity $i,j \in T$ attaining $\beta d_{ij} = L_{ij}$ limits $\lambda_G(\mathbf{d})$ to not exceed β, and because we can ship $\beta d_{ij} \leq L_{ij}$ units separately for every commodity $i,j \in T$, hence also their convex combination $\binom{k}{2}^{-1}\beta\mathbf{d}$.

The query algorithm then computes an estimate for $\lambda_G(\mathbf{d})$ by performing a binary search over all powers of $(1+\varepsilon)$ in the range $[\beta k^{-2}, \beta]$, where each iteration decides, up to $1+2\varepsilon$ multiplicative approximation, whether a given λ in that range is at most $\lambda_G(\mathbf{d})$. The number of iterations is clearly $O(\log_{1+\varepsilon} k^2) \leq O(\frac{1}{\varepsilon}\log k)$.

The approximate decision procedure is performed in two steps. In the first step, we let $\mathbf{d}^-$ be the vector

(LP1)

$$\begin{array}{lll} \lambda_G(\mathbf{d}) = \min & \sum_{e\in E} c_e \ell_e & \\ \text{s.t.} & \sum_{\{s,t\}\in\binom{T}{2}} d_{st}\delta_{st} \geq 1 & \\ & \delta_{st} \leq \sum_{e\in P} \ell_e & \forall \{s,t\} \in \binom{T}{2} \text{ and a path } P \text{ connecting them} \\ & \ell_e \geq 0 & \forall e \in E \\ & \delta_{st} \geq 0 & \forall \{s,t\} \in \binom{T}{2}. \end{array}$$

obtained from $\lambda\mathbf{d}$ by zeroing all coordinates that are at most $2\varepsilon/k^2 \cdot L_{ij}$. This vector can be written as

$$\mathbf{d}^- = \lambda\mathbf{d} - \sum_{i,j\in T} \alpha_{ij} L_{ij} \mathbf{e}_{ij},$$

where $\mathbf{e}_{ij} \in \mathbb{R}_+^{\binom{T}{2}}$ is the standard basis vector corresponding to $\{i,j\}$, and for every $i,j \in T$ we define $\alpha_{ij} := d_{ij}/L_{ij}$ if $d_{ij} \leq 2\varepsilon/k^2 \cdot L_{ij}$, and $\alpha_{ij} := 0$ otherwise. By definition, $\sum_{i,j} \alpha_{i,j} \leq \varepsilon$. The second step lets $\mathbf{d}'$ be the vector obtained from $\mathbf{d}^-$ by rounding down each coordinate to the nearest power of $1+\varepsilon$. Finally, decide whether $\lambda \leq \lambda_G(\mathbf{d})$ by checking whether $\mathbf{d}' \in \mathcal{D}_\varepsilon^{\text{discrete}}$, which is implemented using the dictionary in $O(k^2)$ time.

It remains to prove the correctness of the approximate decision procedure. For one direction, assume that $\lambda \leq \lambda_G(\mathbf{d})$. It follows that the demands $\lambda\mathbf{d} \geq \mathbf{d}^- \geq \mathbf{d}'$ can all be routed in G, and furthermore $\mathbf{d}' \in \mathcal{D}_\varepsilon^{\text{discrete}}$, implying that our procedure reports a correct decision. For the other direction, suppose our procedure reports that $\lambda \leq \lambda_G(\mathbf{d})$, which means that its corresponding $\mathbf{d}' \in \mathcal{D}_\varepsilon^{\text{discrete}} \subset \mathcal{D}(G)$. We can thus write

$$\lambda\mathbf{d} = \mathbf{d}^- + \sum_{ij} \alpha_{ij} L_{ij} \mathbf{e}_{ij} \leq (1+\varepsilon)\mathbf{d}' + \sum_{ij} \alpha_{ij} L_{ij} \mathbf{e}_{ij}.$$

The right-hand side can be described as a positive combination of vectors in $\mathcal{D}(G)$, whose sum of coefficients is $(1+\varepsilon) + \sum_{ij} \alpha_{ij} \leq 1 + 2\varepsilon$. Since $\mathcal{D}(G)$ is convex and contains $\underline{0}$, we have that also $(1+2\varepsilon)^{-1}\lambda\mathbf{d} \in \mathcal{D}(G)$, i.e., that $(1+2\varepsilon)^{-1}\lambda \leq \lambda_G(\mathbf{d})$, which proves the correctness of the decision procedure up to $1+2\varepsilon$ multiplicative approximation. Overall, we have indeed shown that the binary search algorithm approximates $\lambda_G(\mathbf{d})$ within factor $(1+\varepsilon)(1+2\varepsilon) \leq 1 + O(\varepsilon)$. ∎

4 The Clumping Lemma for Flow Sparsifiers

Let $G = (V, E, c)$ be a terminal network with terminal set T. A network $G' = (V', E', c')$ with $T \subseteq V'$ is called a *flow-sparsifier of G with quality $q \geq 1$* if

$$\forall d \in \mathbb{R}_+^{\binom{T}{2}}, \qquad \lambda_G(\mathbf{d}) \leq \lambda_{G'}(\mathbf{d}) \leq q \cdot \lambda_G(\mathbf{d}).$$

This condition is equivalent to writing $\mathcal{D}(G) \subseteq \mathcal{D}(G') \subseteq q \cdot \mathcal{D}(G)$.

For a subset $S \subseteq V$, denote the edges in the induced subgraph $G[S]$ by $E[S] := \{(u,v) \in E : \ u,v \in S\}$. Given a partition $\Pi = \{S_1, S_2, \ldots, S_m\}$ of the vertex set (i.e., $\cup_{i=1}^m S_i = V$), say a distance function δ on the vertex set V is Π-*respecting* if for all $l \in [m]$ and $\{i,j\} \in S_l$ it holds that $\delta_{ij} = 0$.

Proposition 4.1. *Let $G = (V, E, c)$ be a k-terminal network, and fix $0 < \varepsilon < 1/3$ and $b \geq 1$. Suppose there is an m-way partition $\Pi = \{S_1, \ldots, S_m\}$ such that for all $\mathbf{d}' \in \mathcal{D}_\varepsilon^{discrete}$, there exists a Π-respecting distance function δ that is a feasible solution to (LP1) with objective function at most $b \cdot \lambda_G(\mathbf{d}')$. Then the graph G' obtained from G by merging each S_i into a single vertex (keeping parallel edges[2]) is a flow-sparsifier of G with quality $(1+3\varepsilon)b$.*

Proof. The graph G' can equivalently be defined as taking G and adding the edges $E_0 := \cup_{i=1}^m \binom{S_i}{2}$, each with infinite capacity—merging all vertices in each S_i is the same as adding these infinite capacity edges. Formally, let $G' = (V, E \cup E_0, c')$ where $c'(e) = c(e)$ if $e \in E$ and $c'(e) = \infty$ if $e \in E_0$. Then for any $\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}}$, it is immediate that $\lambda_{G'}(\mathbf{d}) \geq \lambda_G(\mathbf{d})$—every flow that is feasible in G is also feasible in G', even without shipping any flow on E_0.

For the opposite direction, without loss of generality we may assume (by scaling) that $\lambda_G(\mathbf{d}) = 1$. Let $\mathbf{d}' \in \mathcal{D}_\varepsilon^{\text{discrete}}$ be the demand vector obtained from $\mathbf{d}$ in the construction of $\mathcal{D}_\varepsilon^{\text{discrete}}$ (by zeroing small coordinates and rounding downwards to the nearest power of $(1+\varepsilon)$). Clearly, $\mathbf{d}' \leq \mathbf{d}$.

First, we claim that $\lambda_G(\mathbf{d}') < 1 + 3\varepsilon$. Indeed, assume to the contrary that $(1+3\varepsilon)\mathbf{d}'$ is feasible in G, i.e., $(1+3\varepsilon)\mathbf{d}' \in \mathcal{D}(G)$. Then the demand

$$\mathbf{d}'' := (1-\varepsilon)(1+3\varepsilon)\mathbf{d}' + \sum_{i\in\binom{T}{2}} \varepsilon/\tbinom{k}{2} \cdot L_i \mathbf{e}_i,$$

[2] From our perspective of flows, parallel edges can also be merged into one edge with the same total capacity.

is a convex combination of demands in $\mathcal{D}(G)$, and thus also $\mathbf{d}'' \in \mathcal{D}(G)$. Observe that $\mathbf{d}'' > \mathbf{d}$ (coordinate-wise), because each coordinate of $\mathbf{d}'$ was obtained from $\mathbf{d}$ by rounding down and possible zeroing (if it is smaller than some threshold), but we more than compensate for this when $\mathbf{d}''$ is created by multiplying $\mathbf{d}'$ by $(1-\varepsilon)(1+3\varepsilon) \geq 1+\varepsilon$ and adding more than the threshold. By down-monotonicity of $\mathcal{D}(G)$ we obtain that $\lambda_G(\mathbf{d}) > 1$ in contradiction to our assumption, and the claim that $\lambda_G(\mathbf{d}') < 1+3\varepsilon$ follows.

To get a handle on the value $\lambda_{G'}(\mathbf{d})$, we rewrite (LP1) for $G' = G + E_0$ to obtain LP (LP2).

By our premise, the demand $\mathbf{d}' \in \mathcal{D}_\varepsilon^{\text{discrete}}$ has a Π-respecting feasible solution $\{\delta_{st}, \ell_e\}$ with value at most $b \cdot \lambda_G(\mathbf{d}')$; note that such a Π-respecting distance function is also a solution to (LP2). Hence $\lambda_{G'}(\mathbf{d}') \leq b \cdot \lambda_G(\mathbf{d}')$. Plugging in $\lambda_G(\mathbf{d}') \leq (1+3\varepsilon)$ and the normalization $\lambda_G(\mathbf{d}) = 1$, we conclude that $\lambda_{G'}(\mathbf{d}) \leq (1+3\varepsilon)b\lambda_G(\mathbf{d})$, which completes the proof of Proposition 4.1. ∎

The next proposition is similar in spirit to the previous one, but with the crucial difference that it allows (or assumes) a different partition of V for every demand $\mathbf{d} \in \mathcal{D}_\varepsilon^{\text{discrete}}$. Its proof is a simple application of the Proposition 4.1.

PROPOSITION 4.2. *Let $G = (V, E, c)$ be a k-terminal network, and fix $0 < \varepsilon < 1/3$, $b \geq 1$, and $m \geq 1$. Suppose that for every $\mathbf{d} \in \mathcal{D}_\varepsilon^{discrete}$, there is an m-way partition $\Pi_{\mathbf{d}} = \{S_1, \ldots, S_m\}$ (with some sets S_j potentially empty) and a $\Pi_{\mathbf{d}}$-respecting distance function that is a feasible solution to (LP1) with objective function value at most $b \cdot \lambda_G(\mathbf{d})$. Then there is G' that is a flow-sparsifier of G with quality $(1+3\varepsilon)b$ and has at most*

$$m^{|\mathcal{D}_\varepsilon^{discrete}|} \leq m^{(\varepsilon^{-1}\log k)^{k^2}}$$

vertices. Moreover, this graph G' is obtained by merging vertices in G.

Proof. For every demand $\mathbf{d} \in \mathcal{D}_\varepsilon^{\text{discrete}}$, we know there is an appropriate m-way partition $\Pi_{\mathbf{d}}$ of V. Imposing all these partitions simultaneously yields a "refined" partition $\Pi = \{S'_1, \ldots, S'_{m'}\}$ in which the number of parts is $m' \leq m^{|\mathcal{D}_\varepsilon^{\text{discrete}}|}$, and two vertices in the same part S'_j of this refined partition if and only if they are in the same part of every initial partition. Now apply Proposition 4.1 using this m'-way partition (using that any $\Pi_{\mathbf{d}}$-respecting distance function is also a Π-respecting one), we obtain graph G' that is a flow-sparsifier of G and has at most m' vertices. Finally, we bound $|\mathcal{D}_\varepsilon^{\text{discrete}}|$ using (3.6). ∎

4.1 Quasi-Bipartite Graphs via Clumping As a warm-up, we use Proposition 4.2 to construct a graph sparsifier for quasi-bipartite graphs with quality $(1+\varepsilon)$, where the size of the sparsifier is only a function of k and ε. Recall that a graph G with terminals T is *quasi-bipartite* if the non-terminals form an independent set [RV99]. For this discussion, we assume that the terminals form an independent set as well, by subdividing every terminal-terminal edge—hence the graph is just bipartite.

THEOREM 4.1. *Let $G = (V, E, c)$ be a k-terminal network whose non-terminals form an independent set, and let $\varepsilon \in (0, 1/8)$. Then G has a flow-sparsifier G' with quality $1+\varepsilon$, which has at most $\exp\{O(k \log \frac{k}{\varepsilon}(\frac{1}{\varepsilon}\log k)^{k^2})\}$ vertices.*

Proof. To apply Proposition 4.2, the main challenge is to bound m, the number of parts in the partition. To this end, fix a demand $\mathbf{d} \in \mathcal{D}_\varepsilon^{\text{discrete}} \subseteq \mathcal{D}(G)$, and let $\{\ell_e, \delta_{ij}\}$ be an optimal solution for the linear program (LP1), hence its value is $\sum_{e\in E} c_e\ell_e = \lambda_G(\mathbf{d})$. We will modify this solution into another one, $\{\ell'_e, \delta'_{ij}\}$, that satisfies the desired conditions. This modification will be carried out in a couple steps, where we initially work only with lengths ℓ_{uv} of edges $(u,v) \in E$, and eventually let δ' be the metric induced by shortest-path distances.

For every $s, t \in T$ define the interval $\Gamma_{st} := [\varepsilon d_{st}, d_{st}]$, and let $\Gamma = \{0\} \cup \left(\cup_{s,t\in T} \Gamma_{st}\right)$, and let Γ^ε contain 0 and all powers of $(1+\varepsilon)$ that lie in Γ. The following claim provides structural information about a "nice" near-optimal solution.

CLAIM 4.1. *Fix a non-terminal $v \in V \setminus T$, Then the edges between v and its neighbors set $N(v) \subset T$ (in G) admit edge lengths $\{\widehat{\ell}_{vt} : t \in N(v)\}$ that*

- *are dominated by l (namely, $\forall t \in N(v)$, $\widehat{\ell}_{vt} \leq \ell_{vt}$);*
- *use values only from Γ^ε (namely, $\forall t \in N(v)$, $\widehat{\ell}_{vt} \in \Gamma^\varepsilon$); and*
- *satisfy $\frac{1+\varepsilon}{(1-\varepsilon)^2}$-relaxed shortest distance constraints (namely, $\forall s, t \in T$, $\widehat{\ell}_{sv} + \widehat{\ell}_{vt} \geq \frac{(1-\varepsilon)^2}{1+\varepsilon}\delta_{st}$).*

Proof. Let every edge length $\widehat{\ell}_{vt}$ be defined as ℓ_{vt} rounded down to its nearest value from Γ^ε. The first two claimed properties then hold by construction. For the third property, recall that l is a feasible LP solution, thus $\ell_{sv} + \ell_{vt} \geq d_{st}$. Assume without loss of generality that $\ell_{sv} \leq \ell_{vt}$. If the large one $\ell_{vt} \geq (1-\varepsilon)^2 d_{st}$, the claim follows because also $\widehat{\ell}_{vt} \geq \frac{(1-\varepsilon)^2}{1+\varepsilon} d_{st}$, regardless of whether ℓ_{vt} is smaller or bigger than d_{st}, where the extra term of $(1+\varepsilon)$ comes from rounding down to the

(LP2)

$$
\begin{array}{lll}
\lambda_{G'}(\mathbf{d}) = \min & \sum_{e\in E} c_e\ell_e & \\
\text{s.t.} & \sum_{\{s,t\}\in\binom{T}{2}} d_{st}\cdot\delta_{st} \geq 1 & \\
& \ell_e = 0 & \forall e \in E_0 := \cup_{i\in[m]}\binom{S_i}{2}, \\
& \delta_{st} \geq \sum_{e\in P}\ell_e & \forall\{s,t\}\in\binom{T}{2} \text{ and } s\text{-}t \text{ path } P \text{ on } E\cup E_0 \\
& \ell_e \geq 0 & \forall e \in E \\
& \delta_{st} \geq 0 & \forall\{s,t\}\in\binom{T}{2}.
\end{array}
$$

nearest power of $(1+\varepsilon)$. Otherwise, $\ell_{vt} < (1-\varepsilon)^2 d_{st}$ hence the smaller one $\ell_{sv} \geq \varepsilon(2-\varepsilon)d_{st}$, and rounding down ensures $\widehat{\ell}_{sv} \geq \frac{\varepsilon(2-\varepsilon)}{1+\varepsilon}d_{st}$. The fact that $\varepsilon < 1/8$ means $\varepsilon(2-\varepsilon)1+\varepsilon \geq 1$, so rounding down does not zero out $\widehat{\ell}_{sv}$. We conclude that the new lengths are at least $\frac{1}{1+\varepsilon}$ times the old ones, namely $\widehat{\ell}_{sv} \geq \frac{\ell_{sv}}{1+\varepsilon}$ and $\widehat{\ell}_{vt} \geq \frac{\ell_{vt}}{1+\varepsilon}$. ■

We proceed with the proof of Theorem 4.1. Define new edge lengths $\{\ell'_e : e \in E\}$ by applying the claim and scaling edge lengths by $\frac{1+\varepsilon}{(1-\varepsilon)^2}$, namely for every $v \in V\setminus T$, and an adjacent $t \in T$, set $\ell'_{vt} := \frac{1+\varepsilon}{(1-\varepsilon)^2}\widehat{\ell}_{vt}$. This scaling and the third property of Claim 4.1 ensures that the shortest-path distances (using edge lengths ℓ'_e) between each pair of vertices i, j is at least δ_{ij}.

Now partition the non-terminals into buckets, where two non-terminals $u, v \in V\setminus T$ are in the same bucket if they "agree" about each of their neighbors $t \in T$: either (i) they are both non-adjacent to t, or (ii) they are both adjacent to t and $\ell'_{ut} = \ell'_{vt}$. Observe that this bucketing is indeed a well-defined equivalence relation. Now for every $u, v \in V\setminus T$ that are the same bucket, add an edge of length $\ell'_{uv} = 0$, and let E' denote this set of new edges. Let δ' be the shortest-path distances according to these new edge-lengths. Observe that the shortest-path distances between the terminals are unchanged by the addition of these new zero-length edges, even though the distances between some non-terminals have obviously changed. Hence (ℓ'_e, δ'_{ij}) is a feasible solution to (LP1), with objective function value at most $\frac{1+\varepsilon}{(1-\varepsilon)^2}\lambda_G(\mathbf{d}) \leq (1+5\varepsilon)\lambda_G(\mathbf{d})$.

We define the equivalence classes of the bucketing above as the sets S_i in Proposition 4.2. Each bucket corresponds to a "profile" vector with k coordinates that represent the lengths of k edges going to the k terminals, if at all there is an edge to the terminals. Each coordinate of this profile vector is an element of Γ^ε or it represents the corresponding edge does not exist. It follows that the number of buckets (or profile vectors) is $m \leq \left(\binom{k}{2}(\log_{1+\varepsilon}\frac{1}{\varepsilon}+3)\right)^k \leq (O(\frac{k^2}{\varepsilon}\log\frac{1}{\varepsilon}))^k \leq (O(\frac{k}{\varepsilon}))^{2k} \leq \exp\{O(k\log(k/\varepsilon))\}$. The theorem follows by applying Proposition 4.2, which asserts the existence of a flow-sparsifier with $m^{(\varepsilon^{-1}\log k)^{k^2}} \leq \exp\{O(k\log\frac{k}{\varepsilon}(\frac{1}{\varepsilon}\log k)^{k^2})\}$ vertices. ■

5 The Splicing and Composition Lemmas

We say that a path is *terminal-free* if all its internal vertices are non-terminals. This terminology shall be used mostly for flow paths, in which the paths' endpoints are certainly terminals. The lemma below refers to two *different* methods of routing a demand $\mathbf{d}$ in a network G. The first method is the usual (and default) meaning, where the demand is routed along arbitrary flow paths. The second method is to route the demand along terminal-free flow paths, and we will say this explicitly whenever we refer to this method. We use a parameter $\rho \geq 1$ to achieve greater generality, although the case $\rho = 1$ conveys the main idea.

LEMMA 5.1. (SPLICING LEMMA) *Let G_a and G_b be two networks having the same set of terminals T, and fix $\rho \geq 1$. Suppose that whenever a demand $\mathbf{d}$ between terminals in T can be routed in G_a using terminal-free flow paths, demand $\mathbf{d}/\rho$ can be routed in G_b (by arbitrary flow paths). Then for every demand $\mathbf{d}$ between terminals in T that can be routed in G_a, demand $\mathbf{d}/\rho$ can be routed in G_b.*

Proof. Consider a demand $\mathbf{d}$ that is routed in G_a using flow f^*, and let us show that it can be routed also in G_b. Fix for f^* a flow decomposition $D = \{(P_1, \phi(P_1)), (P_2, \phi(P_2)), \ldots\}$ for it, where each P_l is a terminal-to-terminal path, and $\phi(P_l)$ is the amount of flow sent on this path. A flow decomposition also specifies the demand vector since $d_{st} = \sum_{st\text{-paths } P\in D}\phi(P)$. If all the paths $(P,\phi) \in D$ are terminal-free, then we know by the assumption of the lemma that demand $\mathbf{d}/\rho$ can be routed in G_b. Else, take a path $(P,\phi) \in D$ that contains internally some terminal—say P routes flow between terminals t', t'' and uses another terminal s internally. We may assume without loss of generality that the flow paths are simple, so $s \notin \{t', t''\}$. We replace the flow $(P, \phi(P))$ in $\mathbf{d}$ by the two paths $(P[t', s], \phi)$ and $(P[s, t'], \phi)$ to get a new flow decomposition D', and de-

note the corresponding demand vector by $\mathbf{d}'$. Note that $d'_{t',t''} = d_{t',t''} - \phi$, whereas $d'_{t',s} = d_{t',s} + \phi$ and the same for $d'_{s,t''}$. Moreover, if $\mathbf{d}'/\rho$ can be routed on some graph G_b with an arbitrary routing, we can connect together ϕ/ρ amount of the flow from t' to s with ϕ/ρ flow from s to t'' to get a feasible routing for $\mathbf{d}/\rho$ in G_b. Moreover the total number of terminals occurring internally on paths in the flow decomposition D' is less than that in $\mathbf{d}$, so the proof follows by a simple induction. ■

The next lemma addresses the case where our network can be described as the gluing of two networks G_1 and G_2, and we already have sparsifiers for G_1 and G_2; in this case, we can simply glue together the two sparsifiers, provided that the vertices at the gluing locations are themselves terminals. Formally, let G_1 and G_2 be networks on disjoint sets of vertices, having terminal sets $T_1 = \{s_1, s_2, \ldots, s_a\}$ and $T_2 = \{t_1, t_2, \ldots, t_b\}$ respectively. Given a bijection $\phi := \{s_1 \leftrightarrow t_1, \ldots, s_c \leftrightarrow t_c\}$ between some subset of T_1 and T_2, the *ϕ-merge* of G_1 and G_2 (denoted $G_1 \oplus_\phi G_2$) is the graph formed by identifying the vertices s_i and t_i for all $i \in [c]$. Note that the set of terminals in G is $T := T_1 \cup \{t_{c+1}, \ldots, t_b\}$.

LEMMA 5.2. (COMPOSITION LEMMA) *Suppose $G = G_1 \oplus_\phi G_2$. For $j \in \{1, 2\}$, let G'_j be a flow-sparsifier for G_j with quality ρ_j. Then the graph $G' = G'_1 \oplus_\phi G'_2$ is a quality $\max\{\rho_1, \rho_2\}$ flow sparsifier for G.*

Proof. Consider a demand $\mathbf{d}$ that is routable in G using flow paths that do not have internal terminals. Since G is formed by gluing G_1 and G_2 at terminals, this means each of the flow paths lies entirely within G_1 or G_2. We can write $\mathbf{d} = \mathbf{d}_1 + \mathbf{d}_2$, where each $\mathbf{d}_j$ is the demand being routed on the flow paths among these that lie within G_j. By the definition of flow-sparsifiers, these demands are also routable in G'_1, G'_2 respectively, and hence demand $\mathbf{d}_1 + \mathbf{d}_2 = \mathbf{d}$ is routable in G' (in fact by paths that lie entirely within G_1 or G_2). Applying the Splicing Lemma (with $\rho = 1$), we get that every demand $\mathbf{d}$ routable in G is routable also in G'.

The argument in the other direction is similar. Assume $\mathbf{d}$ is routable in G' using terminal-free flow paths; then we get two demands $\mathbf{d}_1, \mathbf{d}_2$ routable entirely in G'_1, G'_2 respectively. Scaling these demands down by $\max\{\rho_1, \rho_2\}$, they can be routed in G_1, G_2 respectively, and hence we can route their sum $(\mathbf{d}_1 + \mathbf{d}_2)/\max\{\rho_1, \rho_2\}$ in G. Applying the Splicing Lemma with $\rho = \max\{\rho_1, \rho_2\}$, we get a similar conclusion for all demands routable in G' (on arbitrary flow paths), and this completes the proof. ■

Applications of Splicing/Composition. The Splicing and Composition Lemmas will be useful in many of our arguments: we use them to show a singly-exponential bound for quasi-bipartite graphs in Section 5.1 below, in the sampling approach for quasi-bipartite graphs in Section 6, and also in constructing flow-sparsifiers for series parallel and bounded treewidth graphs in Section 7.

5.1 Quasi-Bipartite Graphs via Splicing

We show how to use Splicing Lemma 5.1 to construct a flow sparsifier for the quasi-bipartite graph of size $(1/\varepsilon)^{\tilde{O}(k)}$.

THEOREM 5.1. *For every quasi-bipartite graph G with terminals T, we can construct a flow sparsifier $\hat{G}$ of $1+\varepsilon$ quality which has size $(1/\varepsilon)^{\tilde{O}(k)}$.*

Proof. The construction goes through several stages. First, we construct G' by rounding down the capacity to an integer power of $1 + \varepsilon$. The main idea is to define "types" for non-terminals v and then merge all vertices of the same type (i.e., the new edge capacity is the sum of the respective edge capacities incident to the merged vertices). The main difficulty is in defining the types.

To define the type, first of all partition all non-terminals v into "super-types", according to the set S of terminals that are connected to v by edges with non-zero capacity. Now fix one such super-type S, i.e., all vertices v such that $\{t \in T : c_{vt} \neq 0\} = S$. Without loss of generality, suppose $S = \{t_1, \ldots t_{h+1}\}$ and $c_{vt_i} \geq c_{vt_{i+1}}$ for $i \in [h]$. For a vertex v, consider the vector of ratios $r^*_v = \{c_{vt_1}/c_{vt_2}, c_{vt_2}/c_{vt_3}, \ldots c_{vt_h}/c_{vt_{h+1}})$. Note that r^*_v's entries are all power of $1+\varepsilon$. Now let $M = k^2/\varepsilon + 1$, and define r_v by thresholding all entries of r^*_t exceeding M by M. The r_v defines the type of the vertex v. Now we merge all vertices v with the same super-type S and type r_t. Denote the new capacities $\hat{c}_{t,u}$ for a terminal t and a non-terminal node u in $\hat{G}$.

Now we proceed to the analysis. First of all, notice that G' is a quality $1 + \varepsilon$ flow sparsifier, so we will care to preserve its flows only. Furthermore, since the main operation is merging of the nodes, we can only increase the set of feasible demands in $\hat{G}$. The main challenge is to prove that if we can route a demand vector $\mathbf{d}$ in G', we can route a demand $(1 - O(\varepsilon))\mathbf{d}$ in G'. Using the Splicing Lemma 5.1, it is enough to consider only demands $\mathbf{d}$ that are feasible using 2-hop paths.

Fix some demand vector $\mathbf{d}$ that is feasible in $\hat{G}$ using 2-hop paths only. Fix a non-terminal node $u \in \hat{G}$, and let $f_{s,t}$ be the flow (of the solution) between $s, t \in T$ via u. Suppose u has super-type S and type $r = (r_1, \ldots r_h)$. We will show that we can route $(1 - O(\varepsilon))f_{s,t}$ in G for all s to t via the nodes $v \in G'$ that have super-type S and type r. This would clearly be sufficient to conclude that $(1 - O(\varepsilon))\mathbf{d}$ is feasible in G'. Let $v_1, \ldots v_m$ be the nodes with super-type S and type r.

We proceed in stages, routing iteratively from the "small flows" to the "large flows" via u. Consider a suffix of r, denoted $r_i, \dots r_h$ where $r_i = M$ and $r_{i'} < M$ for all $i' > i$. For $j \in [m]$, let $\alpha_{v_j} = c_{t_i,v_j}/\hat{c}_{t_i,u}$. Now for all flows f_{st}, where $s \in \{t_{i+1}, \dots t_{h+1}\}$ and $t \in \{t_1, \dots t_h\}$, we route $(1-\varepsilon)\alpha_{v_j} f_{st}$ flow from s to t via v_j in G'. We argue this is possible (even when doing this for all s, t). Namely, consider any edge $e = (t_{i'}, v_j)$ for $t_{i'} \in \{t_{i+1}, \dots t_{h+1}\}$. The flow accumulated on this edge is:

$$\begin{aligned}\sum_t (1-\varepsilon)\alpha_{v_j} f_{t_{i'},t} &= (1-\varepsilon)c_{t_i,v_j}/\hat{c}_{t_i,u} \cdot \sum_t f_{t_{i'},t} \\ &\le (1-\varepsilon)c_{t_i,v_j}/\hat{c}_{t_i,u} \cdot \hat{c}_{t_{i'},u}.\end{aligned}$$

Note that $c_{t_i v_j}/c_{t_{i'},v_j} = r_i \cdot r_{i+1} \cdots r_{i'-1}$, and similarly $\hat{c}_{t_i,u}/\hat{c}_{t_{i'},u} = r_i \cdot r_{i+1} \cdots r_{i'-1}$. Hence the above formula is bounded by $(1-\varepsilon)c_{t_{i'},v_j}$, i.e., we satisfy the edge capacity (with a $1-\varepsilon$ slack, which will help later). Furthermore, we have routed $\sum_j (1-\varepsilon)\alpha_{v_j} f_{st} = (1-\varepsilon) f_{st}$ flow for each s, t.

We will repeat the above procedure for the next suffix of r until we are done routing flow G'. Note that we have at most k such stages.

We need to mention one more aspect in the above argument — what happens to the flow that is contributed to edges $(t_{i'}, v_j)$ where $i' \le i$? The total contribution is at most $k/M \le \varepsilon/k$ fraction of the capacity (since $r_i = M$), which, over all (at most) k stages is still at most ε fraction of the edge capacity. Since we left a slack of ε in the capacity for each edge in the above argument, we still satisfy the capacity constraint overall for each edge.

Finally, to argue about the size of $\hat{G}$, note that there are only 2^k super-types, and there are at most $O(k^2/\varepsilon)^k$ possible vectors r, and hence $\hat{G}$ has size at most $O(2^k \cdot O(k^2/\varepsilon)^k) = (1/\varepsilon)^{\tilde{O}(k)}$. ∎

6 Sampling Approach To Flow Sparsifiers

In this section we develop our sampling approach to construct flow sparsifiers. In particular, for quasi-bipartite graphs we construct in this method flow sparsifiers of size bounded by a polynomial in k/ε. This family includes the graphs for which a lower bound (for exact flow sparsification) was proved in [KR13], and we further discuss how our construction extends to include also the graphs for which a lower bound was proved in [KRTV12].

6.1 Preliminaries

We say that a random variable is *deterministic* if it has variance 0 (i.e., its value is determined with probability 1).

THEOREM 6.1. (A CHERNOFF VARIANT) *Let $X_1, \dots, X_m \ge 0$ be independent random variables, such that each X_i is either deterministic or $X_i \in [0, b]$, and let $X = \sum_{i=1}^m X_i$. Then*

$$\Pr\Big[X \le (1-\varepsilon)\,\mathbb{E}[X]\Big] \le e^{-\varepsilon^2\,\mathbb{E}[X]/(2b)}, \quad \forall \varepsilon \in (0,1),$$
$$\Pr\Big[X \ge (1+\varepsilon)\,\mathbb{E}[X]\Big] \le e^{-\varepsilon^2\,\mathbb{E}[X]/(3b)}, \quad \forall \varepsilon \in (0,1).$$

Proof. First, replace every deterministic X_i with multiple random variables that are still deterministic but are all in the range $[0, b]$. It suffices to prove the deviation bounds for the new summation, because the new variables trivially maintain the independence condition, and the deviation bound does not depend on the number m of random variables.

Assuming now that every random variable is in the range $[0, b]$, the deviation bounds follow from standard Chernoff bounds [MR95, DP09] by scaling all the random variables by factor $1/b$. ∎

6.2 Quasi-Bipartite graphs

Recall that a quasi-bipartite graph is one where the non-terminals form an independent set; i.e., there are no edges between non-terminals.

THEOREM 6.2. *Let $G = (V, E, c)$ be a quasi-bipartite k-terminal network, and let $0 < \varepsilon < 1/8$. Then G admits a quality $1+\varepsilon$ flow-sparsifier $\widehat{G}$ that has at most $\tilde{O}(k^7/\varepsilon^3)$ vertices.*

Our algorithm is randomized, and is based on importance sampling, as follows. Throughout, let $T \subset V$ be the set of k terminals, and assume the graph is connected. We may assume without loss of generality that T also forms an independent set, by subdividing every edge that connects two terminals (i.e., replacing it with a length 2 path whose edges have the same capacities as the edge being replaced). We use a parameter $M := C\varepsilon^{-3}k^5 \log(\frac{1}{\varepsilon}\log k)$, where $C > 0$ is a sufficiently large constant.

1. For every $s, t \in T$, compute a maximum st-flow in G along 2-hops paths. These path are edge-disjoint and each is identified by its middle vertex, this flow is given by

$$F_{st} := \sum_{v \in V \setminus T} F_{st,v}, \text{ where} \qquad (6.7)$$
$$F_{st,v} := \min\{c_{sv}, c_{vt}\}.$$

2. For every non-terminal $v \in V \setminus T$, define a sampling probability

(6.8) $\widetilde{p}_v := \min\{1, p_v\}$, where

$$p_v := M \cdot \max\Big\{ \frac{F_{st,v}}{F_{st}} : s,t \in T \text{ and } F_{st,v} > 0 \Big\}.$$

3. Sample each non-terminal with probability $\widetilde{p}_v$; more precisely, for each $v \in V \setminus T$ independently at random, with probability $\widetilde{p}_v$ scale the capacity of every edge incident to v by a factor of $1/\widetilde{p}_v$, and with the remaining probability remove v from the graph.
4. Report the resulting graph $\widehat{G}$.

For the sake of analysis, it will be convenient to replace step 3 with the following step, which is obviously equivalent in terms of flow.

3'. For each $v \in V \setminus T$, set independently at random $I_v = 1$ with probability $\widetilde{p}_v$ and $I_v = 0$ otherwise (with probability $1 - \widetilde{p}_v$), and scale the capacities of every edge incident to v by a factor of $I_v/\widetilde{p}_v$.

We first bound the size of $\widehat{G}$, and then show that with high probability $\widehat{G}$ is a flow-sparsifier with quality $1 + O(\varepsilon)$.

LEMMA 6.1. *With probability at least* 0.9*, the number of vertices in* $\widehat{G}$ *is at most* $O(k^2 M)$.

Proof. The number of vertices in $\widehat{G}$ is exactly $\sum_{v \in V \setminus T} I_v$, hence its expectation is

$$\begin{aligned}
\mathbb{E}\Big[\sum_{v \in V \setminus T} I_v \Big] &\le \sum_{v \in V \setminus T} p_v \\
&\le M \sum_{v \in V \setminus T} \; \sum_{s,t \in T:\ F_{st,v} > 0} \frac{F_{st,v}}{F_{st}} \\
&= M \sum_{s,t \in T:\ F_{st} > 0} \; \sum_{v \in V \setminus T} \frac{F_{st,v}}{F_{st}} \le O(k^2 M),
\end{aligned}$$

where the second inequality simply bounds the maximum in (6.8) with a summation, and the last inequality follows from (6.7). The lemma then follows by applying Markov's inequality. ∎

LEMMA 6.2. *Let* d *range over all nonzero demand vectors in* $\mathbb{R}_+^{\binom{T}{2}}$*. Then*

$$\Pr\Big[\forall d \neq 0,\ \lambda_{\widehat{G}}(d) \ge (1 - 3\varepsilon) \lambda_G(d) \Big] \ge 0.9, \tag{6.9}$$

$$\Pr\Big[\forall d \neq 0,\ \lambda_{\widehat{G}}(d) \le (1 + 4\varepsilon) \lambda_G(d) \Big] \ge 0.9. \tag{6.10}$$

Observe that Theorem 6.2 follows immediately from Lemmas 6.1 and 6.2. It remains to prove the latter lemma, and we do this next. We remark that the 0.9 probabilities above are arbitrary, and can be easily improved to be $1 - o(1)$.

6.2.1 Proving the lower bound (6.9) The plan for proving (6.9) is to discretize the set of all demand vectors, show a deviation bound for each one (separately), and then apply a union bound. We will thus need the next lemma, which shows that for every fixed demand vector $\mathbf{d}$ (satisfying some technical conditions), if this demand $\mathbf{d}$ is feasible in G, then with high probability almost the same demand $(1 - \varepsilon)\mathbf{d}$ is feasible in $\widehat{G}$.

Given a demand vector $\mathbf{d}$, the problem of concurrent flow along 2-hop paths can be written as linear program (LP3). It has variables f_v^{st} representing flow along a path $s - v - t$, for the commodity $s, t \in T$ and intermediate non-terminal $v \in V \setminus T$. Let $N_G(w)$ denote the set of neighbors of vertex w in the graph G.

LEMMA 6.3. *Fix* $\eta > 0$ *and let* $\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}} \setminus \{0\}$ *be vector that (i) demand* $\mathbf{d}$ *can be satisfied in* G *by flow along* 2*-hop paths, and (ii) every nonzero coordinate* d_{st} *in* $\mathbf{d}$ *is a power of* $1 + \varepsilon$ *in the range* $[\eta F_{st}, F_{st}]$.[3] *Then*

$$\Pr[\textit{demand } (1-\varepsilon)\mathbf{d} \textit{ admits a flow in } \widehat{G} \textit{ along 2-hop paths}] \ge 1 - \tbinom{k}{2} e^{-\varepsilon^2 \eta M/2}.$$

Proof. Given demand vector $\mathbf{d}$, fix a flow f that satisfies it in G along 2-hop paths. Thus, $f_{st} = d_{st} \ge \eta F_{st}$. Let $\widehat{G}$ be the graph constructed using the above randomized procedure, and recall that random variable I_v is an indicator for the event that non-terminal v is sampled in step 3', which happens independently with probability $\widetilde{p}_v$.

Define a flow $\widehat{f}$ in $\widehat{G}$ in the natural way: scale every flow-path in f whose intermediate vertex is $v \in V \setminus T$ by the corresponding $I_v/\widetilde{p}_v$. The resulting flow $\widehat{f}$ is indeed feasible in $\widehat{G}$ along 2-hop paths. It remains to prove that with high probability this flow $\widehat{f}$ routes at least (i.e., a demand that dominates) $(1 - \varepsilon)\mathbf{d}$.

Fix a demand pair (commodity) $s, t \in T$. The amount of flow shipped by $\widehat{f}$ along the path $s - v - t$ is $\widehat{f}_v^{st} := f_v^{st} I_v/\widetilde{p}_v$, and the total amount shipped by $\widehat{f}$ between s and t is

$$\widehat{f}^{st} := \sum_{v \in N_G(s) \cap N_G(t)} \widehat{f}_v^{st} = \sum_{v \in N_G(s) \cap N_G(t)} f_v^{st} \cdot I_v/\widetilde{p}_v. \tag{6.11}$$

By linearity of expectation, $\mathbb{E}[\widehat{f}^{st}] = \sum_v f_v^{st} \cdot \mathbb{E}[I_v]/\widetilde{p}_v = \sum_v f_v^{st} = f^{st}$. Furthermore, we wrote $\widehat{f}^{st}$ in (6.11) as the sum of independent non-negative random variables, where each of summand is either deterministic (when

[3] The range upper bound F_{st} follows anyway from requirement (i). We also remark that the requirement about power of $1 + \varepsilon$ is not necessary for the lemma's proof, but will appear later on.

(LP3)

$$\begin{array}{lll} \max & \lambda & \\ \text{s.t.} & \sum_{v \in N_G(s) \cap N_G(t)} f_v^{st} \geq d_{st}\lambda & \forall \{s,t\} \in \binom{T}{2} \\ & \sum_{s \in N_G(v) \setminus \{t\}} f_v^{st} \leq c_{vt} & \forall (v,t) \in E \\ & f_v^{st} \geq 0 & \forall \{s,t\} \in \binom{T}{2}, \forall v \in N_G(s) \cap N_G(t). \end{array}$$

$\widetilde{p}_v = 1$), or (when $\widetilde{p}_v = p_v < 1$) can be bounded using (6.8) by

$$f_v^{st} \cdot I_v / \widetilde{p}_v \leq f_v^{st} / \widetilde{p}_v \leq F_{st,v}/p_v \leq F_{st}/M.$$

Applying Theorem 6.1, we obtain, as required,

$$\Pr[\widehat{f}^{st} \leq (1-\varepsilon) f^{st}] \leq e^{-\varepsilon^2 f^{st}/(2F_{st}/M)} \leq e^{-\varepsilon^2 \eta M/2}.$$

A trivial union bound over the $\binom{k}{2}$ choices of s,t completes the proof. ■

We proceed now to prove (6.9) using Lemma 6.3.

Proof. [Proof of (6.9)] Set $\eta := \varepsilon/k^2$ and define

$$\mathcal{D}_{LB} := \Big\{ \mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}} \setminus \{0\} \text{ that satisfy requirements (i) and (ii) in Lemma 6.3} \Big\}.$$

Then clearly $|\mathcal{D}_{LB}| \leq \left(2 + \log_{1+\varepsilon} \frac{1}{\eta}\right)^{\binom{k}{2}} \leq \left(\frac{1}{\varepsilon} \log \frac{k}{\varepsilon}\right)^{k^2} \leq \left(\frac{\log k}{\varepsilon}\right)^{O(k^2)}$. Applying Lemma 6.3 to each $d \in \mathcal{D}_{LB}$ and using a trivial union bound, we get that with probability at least $1 - |\mathcal{D}_{LB}| \cdot \binom{k}{2} e^{-\varepsilon^2 \eta M/2} \geq 0.9$, for every $d \in \mathcal{D}_{LB}$ we have that $(1-\varepsilon)d$ can be satisfied in $\widehat{G}$ by 2-hop flow paths. We assume henceforth this high-probability event indeed occurs, and show that it implies the event described in (6.9).

To this end, fix a demand vector $\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}} \setminus \{0\}$, and let us prove that $\lambda_{\widehat{G}}(\mathbf{d}) \geq (1-3\varepsilon)\lambda_G(\mathbf{d})$. We can make two simplifying assumptions about the demand vector $\mathbf{d}$, both of which are without loss of generality. Firstly, we assume that $\lambda_G(\mathbf{d}) = 1$, as we can simply scale $\mathbf{d}$, observing that the event in (6.9) is invariant under scaling. Secondly, we assume that the demand $\mathbf{d}$ can be satisfied in G by 2-hop flow paths. If each such demand $\mathbf{d}$ can be satisfied in $\widehat{G}$ with congestion $1/(1-3\varepsilon)$, Lemma 5.1 implies that every demand can be satisfied with the same congestion.

So consider a demand $\mathbf{d} \in \mathbb{R}_+^{\binom{T}{2}} \setminus \{0\}$, such that $\lambda_G(\mathbf{d}) = 1$ and $\mathbf{d}$ can be satisfied in G by 2-hop flow paths. Let $\mathbf{d}^-$ be the vector obtained from $\mathbf{d}$ by zeroing every coordinate d_{st} that is smaller than $2\eta F_{st}$. This vector can be written as $\mathbf{d}^- = \mathbf{d} - \sum_{st} \alpha_{st} \mathbf{e}_{st}$, where $\mathbf{e}_{st} \in \mathbb{R}_+^{\binom{T}{2}}$ is the standard basis vector for pair (s,t), and $\alpha_{st} := d_{st}$ if this value is smaller than $2\eta F_{st}$, and zero otherwise. Rounding each nonzero coordinate of $\mathbf{d}^-$ down to the next power of $1+\varepsilon$ yields a demand vector $\mathbf{d}' \in \mathcal{D}_{LB}$, and thus by our earlier assumption, $(1-\varepsilon)\mathbf{d}' \geq \frac{1-\varepsilon}{1+\varepsilon}\mathbf{d}^-$ can be satisfied in $\widehat{G}$ by 2-hop flow paths. For each $s,t \in T$, consider the demand vector $F_{st}\mathbf{e}_{st}$. By rounding its single nonzero coordinate down to the next power of $1+\varepsilon$, we obtain a vector in $\mathcal{D}_{LB}$. Hence we conclude that $\frac{1-\varepsilon}{1+\varepsilon} F_{st}\mathbf{e}_{st}$ can be satisfied in $\widehat{G}$ by 2-hop flow paths. The set of demands satisfiable by 2-hop flow paths in $\widehat{G}$ is clearly convex, so taking a combination of such demand vectors with coefficients that add up to $(1-\varepsilon) + \sum_{s,t \in T} \frac{\alpha_{st}}{F_{st}} \leq (1-\varepsilon) + \frac{k^2}{2} \cdot 2\eta = 1$, we conclude that

$$\begin{aligned} (1-\varepsilon) \cdot \tfrac{1-\varepsilon}{1+\varepsilon} \mathbf{d}^- + \sum_{st} \frac{\alpha_{st}}{F_{st}} \cdot \tfrac{1-\varepsilon}{1+\varepsilon} F_{st}\mathbf{e}_{st} \\ \geq \tfrac{(1-\varepsilon)^2}{1+\varepsilon} \Big[\mathbf{d}^- + \sum_{st} \alpha_{st}\mathbf{e}_{st} \Big] \geq (1-3\varepsilon)\mathbf{d} \end{aligned}$$

can be satisfied in $\widehat{G}$. This implies that $\lambda_{\widehat{G}}(\mathbf{d}) \geq 1-3\varepsilon$, which completes the proof of (6.9). ■

6.2.2 Proving the upper bound (6.10) The plan for proving (6.10) is similar, i.e., to prove a deviation bound for every demand in a small discrete set and then apply a union bound. However, we need to bound the deviation in the opposite direction, and thus use the LP that is dual to flow (which can be viewed as "fractional cut"). This requires more care and a more subtle argument.

Moreover, the sampling argument is very general, and we can extend it as follows. Let G be a terminal network such that if we delete the terminal set T then each component of $G \setminus T$ has at most w nodes in it. (The case of quasi-bipartite graphs is precisely when $w = 1$.) The proof above extends to this setting and gives a graph of size w times larger.

The details appear in the full version of the paper [AGK13].

7 Results Using Flow/Cut Gaps

Given the k-terminal network and demand matrix $\mathbf{d}$, recall that $\lambda_G(\mathbf{d})$ is the maximum multiple of $\mathbf{d}$ that

can be sent through G. We can also define the *sparsity* of a cut $(S, V \setminus S)$ to be

$$\Phi_G(S; \mathbf{d}) := \frac{\sum_{e \in \partial S} c_e}{\sum_{i,j:|\{i,j\} \cap S|=1} d_{ij}},$$

and the *sparsest cut* as

$$\Phi_G(\mathbf{d}) := \min_{S \subseteq V} \Phi_G(S; \mathbf{d}).$$

Define the *flow-cut gap* as

$$\gamma(G) := \max_{\mathbf{d} \in \mathcal{D}(G)} \Phi_G(\mathbf{d}) / \lambda_G(\mathbf{d}).$$

It is easy to see that $\Phi_G(\mathbf{d}) \geq \lambda_G(\mathbf{d})$ for each demand vector $\mathbf{d}$ (and hence $\gamma(G) \geq 1$); a celebrated result of [LLR95, AR98] shows that the gap $\gamma(G) = O(\log k)$ for any k-terminal network G. Many results known about the flow-cut gap based on the structure of the graph G and that of the support of the demands in $\mathcal{D}(G)$; in this section we use these results and results about cut sparsifiers to derive new results about flow sparsifiers.

It will be convenient to generalize the notion of a k-terminal network. Given a k-terminal network $G = (V, E, c)$ with its associated subset $T \subseteq V$ of k terminals, define the *demand-support* to be another undirected graph $H = (T, F)$ with some subset of edges F between the terminals T. The *demand polytope* with respect to (G, H) is the set of all demand vectors $\mathbf{d} = (d_e)_{e \in H}$ which are supported on the edges in the demand-support H, that are routable in G; i.e.,

(7.12)
$$\mathcal{D}(G, H) := \{\mathbf{d} \in \mathbb{R}_+^F : \text{ demand } \mathbf{d} \text{ can be routed in } G\}.$$

This is a generalization of (2.5), where we defined H to be the complete graph on the terminal set T. Define the flow-cut gap with respect to the pair (G, H) as

$$\gamma(G, H) := \max_{\mathbf{d} \in \mathcal{D}(G,H)} \Phi_G(\mathbf{d}) / \lambda_G(\mathbf{d}).$$

Analogously to a flow sparsifier, we can define cut-sparsifiers. Given a k-terminal network G with terminals T, a *cut-sparsifier for G with quality $\beta \geq 1$* is a graph $G' = (V', E', c')$ with $T \subseteq V'$, such that for every partition (A, B) of T, we have

$$\text{mincut}_G(A, B) \leq \text{mincut}_{G'}(A, B) \leq \beta \cdot \text{mincut}_G(A, B).$$

A cut-sparsifier G' is *contraction-based* if it is obtained from G by increasing the capacity of some edges, and identifying vertices (which, from the perspective of cuts and flows, is equivalent to adding an infinite capacity edges between vertices).

THEOREM 7.1. *Given a k-terminal network G with terminals T, let G' be a quality $\beta \geq 1$ cut-sparsifier for G. Then for every demand-support H and all $\mathbf{d} \in \mathbb{R}_+^{E(H)}$,*

$$\frac{1}{\gamma(G', H)} \leq \frac{\lambda_{G'}(\mathbf{d})}{\lambda_G(\mathbf{d})} \leq \beta \cdot \gamma(G, H). \tag{7.13}$$

Therefore, the graph G' with edge capacities scaled up by $\gamma(G', H)$ is a quality $\beta \cdot \gamma(G, H) \cdot \gamma(G', H)$ flow sparsifier for G for all demands supported on H.

Moreover, if G' is a contraction-based cut-sparsifier, then trivially $\lambda_G(\mathbf{d}) \leq \lambda_{G'}(\mathbf{d})$, and hence G' itself is a quality $\beta \cdot \gamma(G, H)$ flow sparsifier for G for demands supported on H.

Proof. Consider a demand $\mathbf{d} \in \mathcal{D}(G, H)$; the maximum multiple of it we can route is $\lambda_G(\mathbf{d})$. For any partition (A, B) of the terminal set T, let $d(A, B) := \sum_{\{i,j\}:|\{i,j\} \cap A|=1} d_{ij}$. The flow across a cut cannot exceed that cut's capacity, hence $\lambda_G(\mathbf{d}) \cdot d(A, B) \leq \text{mincut}_G(A, B)$. Since G' is a cut sparsifier of G, we have $\text{mincut}_G(A, B) \leq \text{mincut}_{G'}(A, B)$, and together we obtain

$$\lambda_G(\mathbf{d}) \leq \frac{\text{mincut}_G(A, B)}{d(A, B)}.$$

Minimizing the right-hand-side over all partitions (A, B) of the terminals, we have $\lambda_G(\mathbf{d}) \leq \Phi_{G'}(\mathbf{d})$. The flow-cut gap for G' implies that $\lambda_G(\mathbf{d}) \leq \gamma(G', H) \cdot \lambda_{G'}(\mathbf{d})$, which shows the first inequality in (7.13). For the second one, we just reverse the roles of G and G' in the above argument, but now have to use that $\text{mincut}_{G'}(A, B) \leq \beta \cdot \text{mincut}_G(A, B)$ to get $\frac{\lambda_{G'}(\mathbf{d})}{\beta} \leq \frac{\text{mincut}_G(A,B)}{d(A,B)}$, and hence eventually that $\lambda_{G'}(\mathbf{d}) \leq \beta \cdot \gamma(G, H) \cdot \lambda_G(\mathbf{d})$.

For the second part of the theorem, observe that if G' is contraction-based, then it is a better flow network than G, which means $\lambda_{G'}(\mathbf{d}) \geq \lambda_G(\mathbf{d})$. ∎

This immediately allows us to infer the following results.

COROLLARY 7.1. (SINGLE-SOURCE FLOW SPARSIFIERS) *For every k-terminal network G, there exists a graph G' with 2^{2^k} vertices that preserves (exactly) all single-source and two-source flows.*[4]

Proof. Hagerup et al. [HKNR98] show that all graphs have (contraction-based) cut-sparsifiers with quality $\beta = 1$ and size 2^{2^k}. Moreover, it is known that

[4] A two-source flow means that there are two terminals $t', t'' \in T$ such that every non-zero demand is incident to at least one of t', t''. Single-source flows are defined analogously with a single terminal.

whenever H has a vertex cover of size at most 2, the flow-cut gap is exactly $\gamma(G,H) = \gamma(G',H) = 1$ [Sch03, Theorem 71.1c]. ■

Corollary 7.2. (Outerplanar Flow Sparsifiers) *If G is a planar graph where all terminals T lie on the same face, then G has an exact (quality 1) flow sparsifier with $O(k^2 2^{2k})$ vertices. In the special case where G is outerplanar, the size bound improves to $O(k)$.*

Proof. Okamura and Seymour [OS81] show that the flow-cut gap for planar graphs with all terminals on a single face is $\gamma(G,H) = 1$, and Krauthgamer and Rika [KR13] show that every planar graph G has a contraction-based cut-sparsifier G' with quality $\beta = 1$ and size $O(k^2 2^{2k})$. And since the latter is contraction-based, also this G' is planar with all terminals on a single face, hence $\gamma(G',H) = 1$.

To improve the bound when G is outerplanar, we use a result of Chaudhuri et al. [CSWZ00, Theorem 5(ii)] that every outerplanar graph G has a cut-sparsifiers G' with quality $\beta = 1$ and size $O(k)$, and moreover, this also G' is outerplanar and thus $\gamma(G',H) = 1$. ■

Corollary 7.3. (4-terminal Flow Sparsifiers) *For $k \leq 4$, every k-terminal network has an exact (quality 1) flow sparsifier with at most $k+1$ vertices.*

Proof. Lomonosov [Lom85] shows that the flow-cut gap for at most 4 terminals is $\gamma(G,H) = \gamma(G',H) = 1$, and Chaudhuri et al. [CSWZ00] show that all graphs with $k \leq 5$ terminals have cut-sparsifiers with quality $\beta = 1$ and at most $k+1$ vertices. (See also [KRTV12, Table 1].) ■

The above two results are direct corollaries, but we can use Theorem 7.1 to get flow-sparsifiers with quality 1 from results on cut-sparsifiers, even when the flow-cut gap is more than 1. E.g., for series-parallel graphs we know that the flow-cut gap is exactly 2 [CJLV08, CSW13, LR10], but we give flow-sparsifiers using cut-sparsifiers for series-parallel graphs in the next section.

7.1 Series-Parallel Graphs and Graphs of Bounded Treewidth To begin, we give some definitions. An *s-t series-parallel graph* is defined recursively: it is either (a) a single edge $\{s,t\}$, or (b) obtained by taking a parallel combination of two smaller s-t series-parallel graphs by identifying their s and t nodes, or (c) obtained by a series combination of an s-x series-parallel graph with an x-t series-parallel graph. See Figure 7.1. The vertices s,t are called the *portals* of G, and the rest of the vertices will be called the *internal* vertices.

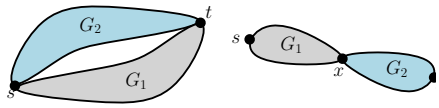

Figure 7.1: Series and Parallel Compositions

Theorem 7.2. (Series-Parallel Graphs) *Every k-terminal series-parallel network G admits an exact (quality 1) flow sparsifier with $O(k)$ vertices.*

Proof. The way we build series-parallel graphs gives us a decomposition tree $\mathcal{T}$, where the leaves of $\mathcal{T}$ are edges in G, and each internal node prescribes either a series or a parallel combination of the graphs given by the two subtrees. We can label each node in $\mathcal{T}$ by the two portals. We will assume w.l.o.g. that $\mathcal{T}$ is binary.

Consider some decomposition tree $\mathcal{T}$ where the two portals for the root node are themselves terminals, and let the number of internal terminals in $\mathcal{T}$ be k. (Giving us a total of $k+2$ terminals, including the portals.) Consider the two subtrees $\mathcal{T}_1, \mathcal{T}_2$. The easy case is when the number of internal terminals in $\mathcal{T}_1, \mathcal{T}_2$, which we denote k_1, k_2, are both strictly less than k. Let G_i be the graph defined by $\mathcal{T}_i$. In case G_1, G_2 are in parallel, recursively construct sparsifiers G_1', G_2' for them, and compose them in parallel to get G'; the Composition Lemma implies this is a sparsifier for G. In case they are in series, the middle vertex may not be a terminal: so add it as a new terminal, recurse on G_1, G_2, and again compose G_1', G_2'. In either case, the number of internal vertices in the new graph is $S(k) \leq S(k_1) + S(k_2) + 1$, where we account for adding the middle vertex as a terminal.

Now suppose all the k internal terminals of the root of $\mathcal{T}$ lie are also internal terminals of $\mathcal{T}_1$. In this case, find the node in $\mathcal{T}$ furthest from the root such that the subtree $\mathcal{T}'$ rooted at this node still has k internal terminals, but neither of its child subtrees $\mathcal{T}_1', \mathcal{T}_2'$ contains all the k internal terminals.[5] There must be such a node, because the leaves of $\mathcal{T}$ contain no internal terminals. Say the portals of $\mathcal{T}'$ are s', t'. And say the graphs given by $\mathcal{T}', \mathcal{T}_1', \mathcal{T}_2'$ are G', G_1', G_2'. The picture looks like one of the two cases in Figure 7.2.

Add the two portals s', t' of G' and the middle vertex x between G_1', G_2' (if it was a series combination) as new terminals. Observe that G is obtained by composing $G \setminus G'$ with G' (at the new terminals s', t'),

[5] The easy case above is a special case of this, where $\mathcal{T}' = \mathcal{T}$.

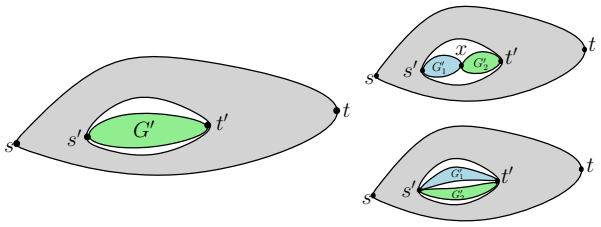

Figure 7.2: The subgraph G' within G

hence we can apply the Composition Lemma to the sparsifiers for $G \setminus G'$ and for G'. We can use Corollary 7.3 to find a flow sparsifier for $G \setminus G'$: it has only 4 terminals s, t, s', t'. To find a flow sparsifier for G', we recurse on G'_1, G'_2 and then combine the resulting sparsifiers H'_1, H'_2 by the Composition Lemma to get a sparsifier H' for G'. Overall, we obtain a flow sparsifier for G with at most $S(k) \leq S(k_1) + S(k_2) + (c_4 - 4) + 3$ internal vertices, where the number of new vertices generated by Corollary 7.3 is at most $c_4 - 4$, and we added in at most 3 new terminals (namely s', t' and possibly x).

In either case, we arrive at the recurrence $S(k) \leq S(k_1) + S(k_2) + c_4$, where $k_1 + k_2 \leq k$ and $k_1, k_2 \geq 1$. The base case is when there are at most 2 internal terminals, in which case we can use Corollary 7.3 again to get $S(1), S(2) \leq c_4$. The recurrence solves to $S(k) \leq (2k-1) \cdot c_4$. Adding the two portal terminals of $\mathcal{T}$ still remains $O(k)$, and proves the theorem. ■

7.2 Extension to Treewidth-w Graphs The general theorem about bounded treewidth graphs follows a similar but looser argument. The only fact about a treewidth-w graph $G = (V, E)$ we use is the following.

THEOREM 7.3. ([REE92]) *If a graph $G = (V, E)$ has treewidth w, then for every subset $T \subseteq V$, there exists a subset $X \subseteq V$ of w vertices such that each component of $G - X$ contains at most $\frac{2}{3}|T \setminus X|$ vertices of T.*

THEOREM 7.4. *Suppose every k-terminal network admits a flow sparsifier of quality $q(k)$ and size $S(k)$. Then every k-terminal network G with treewidth w has a $q(6w)$-quality flow sparsifier with at most $k^4 \cdot S(6w)$ vertices.*

Proof. The proof is by induction. Consider a graph G: if it has at most $6w$ terminals, we just build a $q(6w)$-quality vertex sparsifier of size $S(6w)$.

Else, let T be the set of terminals in G, and use Theorem 7.3 to find a set X such that each component of $G - X$ contains at most $\frac{2}{3}|T \setminus X|$ terminals. Suppose the components have vertex sets $V_1, V_2, \ldots, V_l$; let $G_i := G[V_i \cup X]$. Recurse on each G_i with terminal set $(T \cap V_i) \cup X$ to find a flow sparsifier G'_i of quality $q(6w)$. Now use the Composition lemma to merge these sparsifiers G'_i together and give the sparsifier G' of the same quality. Now use the Composition lemma to merge these sparsifiers G'_i together and give the sparsifier G' of the same quality.

If the number of terminals in G was k_G, the number of terminals in each G_i is smaller by at least $\frac{1}{3}k_G - w = k_G/6$, and hence $k_{G_i} \leq 5/6\, k_G$. Hence the depth of the recursion is at most $h := \log_{6/5}(k/w) \leq \log_{6/5} k$, and the number of leaves is at most 2^h. Each leaf gives us a sparsifier of size $S(6w)$, and combining these gives a sparsifier of size at most $S(6w) \cdot k^{\log_{6/5} 2} \leq S(6w) \cdot k^4$. ■

Using, e.g., results from Englert et al. [EGK+10] we can achieve $q(k) = O\big(\frac{\log k}{\log \log k}\big)$ and $S(k) = k$, which gives the results stated in Section 1.

References

[AGK13] A. Andoni, A. Gupta, and R. Krauthgamer. Towards $(1+\varepsilon)$-approximate flow sparsifiers. Submitted to arXiv, 2013.

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

[BK96] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *28th Annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996.

[Chu12] J. Chuzhoy. On vertex sparsifiers with Steiner nodes. In *44th symposium on Theory of Computing*, pages 673–688. ACM, 2012.

[CJLV08] A. Chakrabarti, A. Jaffe, J. R. Lee, and J. Vincent. Embeddings of topological graphs: Lossy invariants, linearization, and 2-sums. In *49th Annual IEEE Symposium on Foundations of Computer Science*, pages 761–770. IEEE Computer Society, 2008.

[CLLM10] M. Charikar, T. Leighton, S. Li, and A. Moitra. Vertex sparsifiers and abstract rounding algorithms. In *51st Annual Symposium on Foundations of Computer Science*, pages 265–274. IEEE Computer Society, 2010.

[CSW13] C. Chekuri, F. B. Shepherd, and C. Weibel. Flow-cut gaps for integer and fractional multiflows. *J. Comb. Theory Ser. B*, 103(2):248–273, March 2013.

[CSWZ00] S. Chaudhuri, K. V. Subrahmanyam, F. Wagner, and C. D. Zaroliagis. Computing mimicking networks. *Algorithmica*, 26:31–49, 2000.

[DP09] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 2009.

[EGK+10] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar. Vertex sparsifiers: New results from old techniques. In *13th*

International Workshop on Approximation, Randomization, and Combinatorial Optimization, volume 6302 of *Lecture Notes in Computer Science*, pages 152–165. Springer, 2010.

[FM95] T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. *J. Comput. Syst. Sci.*, 51(2):261–272, 1995.

[HKNR98] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57:366–375, 1998.

[Kar94] D. R. Karger. Random sampling in cut, flow, and network design problems. In *26th Annual ACM Symposium on Theory of Computing*, pages 648–657. ACM, 1994.

[KR13] R. Krauthgamer and I. Rika. Mimicking networks and succinct representations of terminal cuts. In *24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1789–1799. SIAM, 2013.

[KRTV12] A. Khan, P. Raghavendra, P. Tetali, and L. A. Végh. On mimicking networks representing minimum terminal cuts. *CoRR*, abs/1207.6371, 2012.

[LLR95] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[LM10] F. T. Leighton and A. Moitra. Extensions and limits to vertex sparsification. In *42nd ACM symposium on Theory of computing*, STOC, pages 47–56. ACM, 2010.

[Lom85] M. V. Lomonosov. Combinatorial approaches to multiflow problems. *Discrete Appl. Math.*, 11(1):1–93, 1985.

[LR99] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[LR10] J. R. Lee and P. Raghavendra. Coarse differentiation and multi-flows in planar graphs. *Discrete Comput. Geom.*, 43(2):346–362, January 2010.

[MM10] K. Makarychev and Y. Makarychev. Metric extension operators, vertex sparsifiers and lipschitz extendability. In *51st Annual Symposium on Foundations of Computer Science*, pages 255–264. IEEE, 2010.

[Moi09] A. Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *50th Annual Symposium on Foundations of Computer Science*, FOCS, pages 3–12. IEEE, 2009.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[OS81] H. Okamura and P. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981.

[Ree92] B. A. Reed. Finding approximate separators and computing tree width quickly. In *STOC*, pages 221–228, 1992.

[RV99] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric steiner tree problem. In *SODA*, pages 742–751, 1999.

[Sch03] A. Schrijver. *Combinatorial Optimization*. Springer, 2003.

[Shm97] D. Shmoys. Cut problems and their applications to divide-and-conquer. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[SS11] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, December 2011.

[ST04] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *36th Annual ACM Symposium on Theory of Computing*, pages 81–90. ACM, 2004.