VERTEX SPARSIFICATION AND OBLIVIOUS REDUCTIONS *

ANKUR MOITRA †

Abstract. Given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals of size k, we construct an undirected, capacitated graph $G' = (K, E')$ for which the cut-function approximates the value of *every* minimum cut separating any subset U of terminals from the remaining terminals $K - U$. We refer to this graph G' as a *cut-sparsifier*, and we prove that there are cut-sparsifiers that can approximate all these minimum cuts in G to within an approximation factor that only depends poly-logarithmically on k, the number of terminals. We prove such cut-sparsifiers exist through a zero-sum game, and we construct such sparsifiers through oblivious routing guarantees. These results allow us to derive a more general theory of Steiner cut and flow problems, and allow us to obtain approximation algorithms with guarantees independent of the size of the graph for a number of graph partitioning, graph layout and multicommodity flow problems for which such guarantees were previously unknown.

Key words. sparsification, approximation algorithm, graph partitioning, routing, metric space

AMS subject classifications.

1. Introduction.

1.1. Background. In the seminal paper of [29], Leighton and Rao established an $O(\log n)$ approximate min-cut max-flow theorem for uniform maximum concurrent flows (where n is the size of the graph) and used this to gave an $O(\log n)$ approximation algorithm for uniform sparsest cut. The sparsest cut problem is a fundamental primitive in many divide-and-conquer based approximation algorithms (see also [38]). In fact through the divide-and-conquer paradigm, Leighton and Rao were able to give the first polylog(n) approximation algorithms for a variety of NP-hard graph partitioning and graph layout problems - including balanced separators, min-cut linear arrangement, crossing number, VLSI layout and minimum feedback arc set. It is hard to over-state the influence of this approximate min-cut max-flow theorem in theoretical computer science; this theorem has found applications in everything from approximating how quickly a Markov chain mixes [22], [23], [37], to bounding the all-pairs network coding rate in an undirected graph [1].

A number of subsequent papers considered the question of giving an approximate min-cut max-flow theorem in a more general setting. Klein, Rao, Agrawal and Ravi [26] considered the general maximum concurrent flow problem in which the demands can be arbitrary, rather than a fixed, constant demand between all pairs as in the uniform maximum concurrent flow problem. In the case in which all capacities and all demands are integer, [26] gave an $O(\log C \log D)$ approximate min-cut max-flow theorem, where C and D are the sum of all capacities and all demands respectively. [26] then used this result to give polylog(n) approximation algorithms for generalized multi-way edge-cut and node-cut problems. Note that this result is weak when either the ratio of the largest capacity to the smallest capacity or the ratio of the largest demand to the smallest demand is very large compared to the graph size. Plotkin and

*This article previously appeared in the conference proceedings for FOCS 2009, under the title "Approximation Algorithms for Multicommodity-Type Problems with Guarantees Independent of the Graph Size."

†moitra@mit.edu. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (EECS) and Computer Science and Artificial Intelligence Laboratory (CSAIL), Cambridge, MA 02139. This research was supported in part by a Fannie and John Hertz Foundation Fellowship.

Tardos [34] combined the results of [29], [26], [17], and [25] with a variety of intricate flow scaling techniques for handling demands of varying quantity, and were able to given an $O(\log^2 k)$ approximate min-cut max-flow theorem, where k is the number of demands to be routed (i.e. the number of commodities).

Linial, London and Rabinovich [30] and Aumann and Rabani [6] solved a major open question and gave an optimal $O(\log k)$ approximate min-cut max-flow theorem for general maximum concurrent flow problems. This bound does not require the capacities or demands to be integer, and both [30] and [6] use discrete metric embeddings as an elegant way to handle demands of varying quantity without re-scaling. Also Garg, Vazirani and Yannakakis [17] gave an $O(\log k)$ approximate min-cut max flow theorem for maximum multi-flow using a combinatorial approach based on region growing.

Here we derive a more general theory of Steiner cut and flow problems, and we prove bounds that are poly-logarithmic in k. These bounds apply to a much broader class of multicommodity flow and cut problems, rather than to just maximum concurrent flow (as in [30] and [6]) or maximum multi-flow (as in [17]). Our results are motivated by the following meta question:

META QUESTION 1. *Suppose we are given a* polylog(n) *approximation algorithm for a flow or cut problem - when can we give a* polylog(k) *approximation algorithm for a generalization of this problem to a Steiner cut or flow problem?*

In particular, if we are given a polylog(n) approximation algorithm and if we consider an instance in which there are only k "interesting" nodes (which we will refer to as terminals), can we give a polylog(k) approximation algorithm for this problem using the previous approximation algorithm as a *black box*?

Thus our goal is an approximation guarantee that is independent of the size of the graph, and only depends on the number of commodities (or the number of terminals in a Steiner cut problem). For many natural applications of multicommodity flows and cuts, we expect that the number of commodities k is much smaller than n, and for such problems we get approximation algorithms that have much stronger guarantees. In §4, we give a more detailed argument for why such guarantees are likely to be more applicable in practice (to the sample problem of oblivious routing).

1.2. Vertex Sparsification. The key to this paper is a structural result, which we can use to map an optimization problem over cuts or flows (for which there are only k "interesting" nodes) to a corresponding optimization problem on a graph on exactly k nodes while approximately preserving the cost of the optimal solution. Actually, this mapping will be *oblivious* to the particular optimization problem that we want to solve, and will only require that the problem be approximately characterized by the values of minimum cuts separating subsets of terminals.

If our mapping doesn't depend on the particular optimization problem, in order to guarantee that this mapping approximately preserves the optimum, we will need to ensure that this mapping approximately preserves *everything* that the optimization problem could possibly depend on! But how well can we do this? This is the question which we formalize in this section, and the answer to this question is central to this paper and all approximation algorithms that we derive.

Suppose we are given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals of size k. Let $h : 2^V \to \mathbb{R}^+$ denote the cut function of G:

$$h(A) = \sum_{e \in \delta(A)} c(e)$$

where $\delta(A)$ denotes the set of edges crossing the cut $(A, V-A)$. We define the function $h_K : 2^K \to \mathbb{R}^+$ which we refer to as the terminal cut function on K:

$$h_K(U) = \min_{A \subset V \text{ s.t. } A \cap K = U} h(A)$$

We will also use $\delta_K(U)$ to denote the set of pairs $(a, b) \in K$ which cross the partition $(U, K - U)$. The combinatorial interpretation of the terminal cut function is that $h_K(U)$ is just the minimum edge cut separating U from $K - U$ in G. Note that U is required to be a subset of K and that we can compute the value and cut achieving $h_K(U)$ using any max-flow algorithm. We prove that there is an undirected, capacitated graph $G' = (K, E')$ *on just the terminals* so that the cut function $h' : 2^K \to \mathbb{R}^+$ satisfies

$$h_K(U) \leq h'(U) \leq O\left(\frac{\log k}{\log \log k}\right) h_K(U)$$

for all $U \subset K$. So we can approximate the terminal cuts in G on a graph G' on k nodes and this approximation factor is independent of the size of the graph G. Note that $h'(U)$ is just the value of a cut, and there is no longer any optimization problem that is implicit in this function. This is in contrast to the terminal cut function $h_K(U)$, for which we need to compute a max-flow to find any particular value of this function. So even though there is an optimization problem implicit in $h_K(U)$ and there is no optimization problem implicit in $h'(U)$, h' still approximates h_K everywhere! Also, note that there are in principle *exponentially* (in k) many degrees of freedom for the terminal cut function, yet there are only $\binom{k}{2}$ degrees of freedom for h' - one degree of freedom for each choice of an edge capacity in G'.

We will refer to such a graph $G' = (K, E')$ as a vertex sparsifier for cuts, or *cut-sparsifier* and we will refer to the multiplicative factor by which h' approximates h_K as the *quality*. In fact, if G is planar or if G excludes a fixed minor, then this bound improves and there are $O(1)$-quality cut-sparsifiers. Many naturally occurring networks do in fact exclude a small, fixed minor: road networks are often (close to) planar, the internet graph is well-known to have small treewidth, and even social networks have the so-called padded decomposition property which is also enough to guarantee that constant quality cut-sparsifiers exist.

This question is quite different from more classical questions in combinatorial optimization. In particular, many previously considered questions only attempt to approximate small terminal-cuts (i.e. approximating *local connectivity* in Mader's Theorem). Yet approximating these small terminal-cuts in general says *nothing* non-trivial about approximating the terminal cut function *everywhere*. We give a more detailed discussion in §2, and we also relate this question posed here to the notion of distortion in metric embeddings.

Our approach to proving this result is through metric geometry. Even though this question is a purely structural question about graphs, constructing good cut-sparsifiers by reasoning directly about the terminal cut function h_K seems daunting. Approximation is necessary - in fact the best known lower bounds ($\tilde{\Omega}(\sqrt{\log k})$ due to Makarychev and Makarychev [31]) are polynomially related to the upper bound that we give here. There are exponentially many min-cuts in G that we are required to (approximately) preserve, and the structure of a good quality cut-sparsifier seems to depend not only on all these values, but also on *how* big and small min-cuts in

G interact. Instead, our approach is to prove this result using a zero-sum game to transform this global, structural question about graph cut functions into a local question about metric spaces. In fact, this dual question turns out to be a special case of the well-studied 0-extension problem [24], [9], [13].

Also, for the applications of cut-sparsification in this paper there is no reason to restrict a cut-sparsifier to be a graph on just the terminals as opposed to the relaxed restriction that the number of total nodes be at most $poly(k)$. Yet even if we allow $poly(k)$ nodes in addition to the terminals, we do not know how to better approximate the terminal-cut function beyond the results presented here.

1.3. Constructive Results and Oblivious Reductions. The result asserted in the previous section is *non-constructive* because we prove existence by bounding the game value of a zero-sum game. In this game, both players have strategy sets of exponential size, and the best response question in both cases is NP-hard. Even the width of the game can be unbounded with respect to n and k.

So we need to do considerably more work to give a polynomial time construction of a cut-sparsifier G' that approximates the terminal cut function of the original graph G (to within a worse but still polylog(k) factor). Perhaps of independent interest, our construction uses oblivious routing in a somewhat surprising way: Given polytopes $Q, P \subset \mathbb{R}^{poly(k)}$ (that are given by a separation oracle), consider the question of finding a unit vector $u \in \mathbb{R}^{poly(k)}$ that maximizes the ratio

$$\frac{\max \lambda_P \text{ s.t. } \lambda_P u \in P}{\max \lambda_Q \text{ s.t. } \lambda_Q u \in Q}$$

We call this question the *Max-Min Congestion Problem* for P, Q. We could use Lowner-John ellipsoids to get an $O(poly(k))$-approximation to this problem [20], but in the special case in which Q is the set of demand vectors that are routable in an undirected, capacitated graph G' (on k nodes) with congestion at most 1, we give an $O(\log k)$ approximation algorithm for this problem. This approximation guarantee is based on using oblivious routing schemes to give a geometric relaxation Q' for Q. We believe that this notion of interpreting oblivious routing as a *geometric phenomenon* is of independent interest and may be useful in the context of designing other approximation algorithms.

The outline of our constructive result is to use the existential result to show that a certain polytope (which captures the question of whether there is a good quality cut-sparsifier) is non-empty. Thus our constructive result is predicated on our existential result. We then use the approximation algorithm mentioned above to give an approximate separation oracle for this polytope, and this in turn allows us to find an approximately feasible point, and hence a good quality cut-sparsifier for the original graph.

This construction allows us to reduce a broad class of multicommodity-type problems to a uniform case (on k nodes) at the cost of a multiplicative loss of a polylog(k) in the approximation guarantee. Interestingly, these reductions are oblivious to the actual optimization problem that we want to solve. All that we require is that our optimization problem be (at least approximately) characterized by the terminal cut function. Some optimization problems depend very explicitly on just the terminal cut function. In such cases it is clear that mapping the optimization problem to a good cut-sparsifier approximately preserves the value of the optimum. Yet there are many other optimization problems which at first do not appear to depend on just

the terminal cut function but nevertheless can be (approximately) re-written as an optimization problem that only depends on h_K.

We give three main patterns for how to perform this re-writing. In §6 we do this for graph partitioning problems and for graph layout problems, and in §4 we do this for multicommodity flow problems (using the approximate relationship between generalized sparsest cut and maximum concurrent flow). For these general categories of problems, mapping the optimization problem to a good cut-sparsifier approximately preserves the optimum and as it turns out, we will also be able to find a good solution in the original graph based on a good solution in the cut-sparsifier as well. So this mapping is independent of the actual optimization problem that we want to solve, and utilizing our re-writing techniques we will see that this reduction is successful for basically all non-pathological cut or flow problems.

Using these patterns for how to apply our result, we give polylog(k) approximation algorithms for a number of problems for which such results were previously unknown, such as requirement cut[2], ℓ-multicut, and natural Steiner generalizations of oblivious routing, min-cut linear arrangement and minimum linear arrangement.

2. Preliminaries.

2.1. Cut-Sparsifiers and Quality. Here we formalize the notion of a cut-sparsifier and the quality of a cut-sparsifier. Suppose we are given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals of size k. Let $h : 2^V \to \mathbb{R}^+$ denote the cut function of G:

$$h(A) = \sum_{e \in \delta(A)} c(e)$$

We define the function $h_K : 2^K \to \mathbb{R}^+$ which we refer to as the terminal cut function on K:

$$h_K(U) = \min_{A \subset V \text{ s.t. } A \cap K = U} h(A)$$

DEFINITION 2.1. *G' is a* cut-sparsifier *for the graph $G = (V, E)$ and the terminal set K if G' is a graph on just the terminal set K (i.e. $G' = (K, E')$) and if the cut function $h' : 2^K \to \mathbb{R}^+$ of G' satisfies (for all $U \subset K$)*

$$h_K(U) \leq h'(U).$$

The goal of a cut-sparsifier is to everywhere approximate the terminal cut function. So then we can define a notion of quality for any particular cut-sparsifier, which captures how faithfully the cut function of G' *everywhere* approximates the terminal cut function:

[2]Subject to the mild technical restriction that the number of groups g be at most quasi-polynomial in k. Otherwise we give an $O(\text{polylog}(k) \log g)$-approximation algorithm and if g is not quasi-polynomial in k then this approximation algorithm is dominated by $O(\log g)$ and there is a lower bound of $\Omega(\log g)$ for the approximability of this problem via a reduction from set cover [33]. Independently, [18] also considered the question of giving an approximation algorithm for Requirement Cut with guarantees independent of the graph size, and were able to directly improve the previous best approximation algorithm due to [33] and give a better polylog(k)-approximation algorithm than the one we present here.

DEFINITION 2.2. *The* quality *of a cut-sparsifier G' is defined as*

$$max_{U \subset K} \frac{h'(U)}{h_K(U)}.$$

We will abuse notation and define $\frac{0}{0} = 1$ so that when U is disconnected from $K - U$ in G or if $U = \emptyset$ or $U = K$, the ratio of the two cut functions is 1 and we ignore these cases when computing the worst-case ratio and consequently the quality of a cut-sparsifier.

The starting point of this paper is to prove that in general, for any capacited graph $G = (V, E)$ and any set $K \subset V$ of $|K| = k$ terminals, there is an $O(\frac{\log k}{\log \log k})$-quality cut-sparsifier. In fact, this bound improves to $O(r^2)$ in the case in which G excludes $K_{r,r}$ as a minor.

There are many results in combinatorial optimization that demonstrate that certain graph connectivity properties can be approximated on a smaller graph. For example, if we define the *local connectivity* of two terminals $a, b \in K$ as the minimum cut in G separating a and b, then Mader's Theorem (see [27]) implies that there is a graph $G' = (K, E')$ so that for all pairs of terminals $a, b \in K$, G' exactly preserves the local connectivity.

Yet results of this form only guarantee that G' preserves minimum cuts separating subsets of terminals for *small* subsets of terminals. Here, preserving the local connectivity only requires preserving the minimum cuts separating single pairs of terminals from each other. Consider, for example, the graph G which is the complete bipartite graph with the k terminals on one side and 2 nodes on the other. The local connectivity between any pair of terminals is 2, and applying the splitting-off operation in Mader's Theorem iteratively results in the graph $G' = (K, E')$ which is a cycle. This preserves the local connectivity exactly, and yet if we bisect the cycle we get an edge cut of size 2 in G' that cuts the graph into two $\Omega(k)$-sized sets of terminals U and $K - U$. But the capacity of the minimum cut separating U from $K - U$ in G is $\Omega(k)$. So the cut function of G' does not well approximate the terminal cut function everywhere. And in general, results in combinatorial optimization about preserving minimum cuts separating small subsets of terminals will be useless for our purposes.

We also note that many graph partitioning problems (for example generalized sparsest cut, or generalized bisection type problems) depend on minimum cuts separating U and $K - U$ for *large* sized sets U. So if we are given a graph G' which approximately preserves just the small terminal cuts, we cannot guarantee that mapping, say, a generalized sparsest cut problem to G' approximately preserves the value of the optimum. So if we want to perform reductions that are oblivious to the particular optimization problem, we really do need to preserve all minimum cuts separating every subset of terminals, and the above question really is the right question in this context.

2.2. Relations to Distortion. This notion of the quality of a cut-sparsifier should be regarded as very similar to the notion of distortion in discrete metric embeddings. The canonical approach to applying discrete metric embeddings to an optimization problem is to embed a complicated metric into (usually a distribution on) simpler metrics. Then one can often solve the problem exactly (or within a constant factor) on this simpler class of metrics, but one pays a price in the approximation guarantee that is proportional to distortion.

Similarly, the way in which we will apply cut-sparsifiers to the problems considered in this paper is to take an optimization problem in the original graph and map this to a

corresponding problem on the cut-sparsifier. As long as the optimization problem can be (at least approximately) characterized by the terminal cut function, this mapping will approximately preserve the cost of the optimal solution.

Both quality and distortion capture how faithfully a "simpler" optimization problem approximates the original optimization problem. In fact, as we will see the notion of quality and the notion of distortion should be thought of as duals to each other.

This analogy can be extended: our reductions are oblivious to the specifics of the optimization problem. This resembles the way in which metric embeddings are applied to on-line optimization problems: A distribution on simpler metrics is chosen in a way that depends only on the original metric space, and is oblivious to the actual requests in an on-line optimization problem. See Bartal's seminal paper [8] for applications of low-distortion embeddings to problems such as metrical task systems, server problems, distributed paging and dynamic storage rearrangement.

2.3. Maximum Concurrent Flow. A basic object of study in this paper will be the maximum concurrent flow problem. An instance of this problem consists of an undirected graph $G = (V, E)$, a capacity function $c : E \to \mathbb{R}^+$ that assigns a non-negative capacity to each edge, and a set of demands $\{(s_i, t_i, d_i)\}$ where $s_i, t_i \in V$ and d_i is a non-negative real value. For such problems we set $K = \cup_i \{s_i, t_i\}$ and let k denote $|K|$. The maximum concurrent flow question asks, given such an instance, what is the largest fraction of the demand that can be simultaneously satisfied? This problem can be formulated as a polynomial-sized linear program, and hence can be solved in polynomial time. However, a more natural formulation of the maximum concurrent flow problem can be written using an exponential number of variables.

For any $a, b \in V$ let $P_{a,b}$ be the set of all (simple) paths from a to b in G. Then the maximum concurrent flow problem can be written as :

$$\begin{array}{ll} \max & \lambda \\ \text{s.t.} & \\ & \sum_{p \in P_{s_i,t_i}} x(p) \geq \lambda d_i \\ & \sum_{p \ni e} x(p) \leq c(e) \\ & x(p) \geq 0. \end{array}$$

For a maximum concurrent flow problem, let λ^* denote the optimum. We defined the cut function h and the terminal cut function h_K in §1.2. We also define the demand function $d : 2^K \to \mathbb{R}^+$ as

$$d(U) = \sum_{(s_i,t_i) \in \delta_K(U)} d_i$$

which given $U \subset K$ is just the total demand that has exactly one endpoint in U and one endpoint in $K - U$ (since $s_i, t_i \in K$).

THEOREM 2.3. *[30] [6] If all demands are supported in K, and $|K| = k$, then there exists a cut $A \subset V$ such that*

$$\frac{h(A)}{d(A \cap K)} \leq O(\log k)\lambda^*.$$

We are interested in multicommodity-type problems, which we informally define as problems that are only a function of the terminal cut function h_K and the congestion of multicommodity flows with demands supported in the set K. We want to find

a graph $G' = (K, E')$ and a capacity function $c' : E' \to \mathbb{R}^+$ such that for all $U \subset K$:

$$h_K(U) \leq h'(U) \leq \text{polylog}(k) h_K(U)$$

where $h' : K \to \mathbb{R}^+$ is the cut function defined on the graph G'. For non-pathological Steiner cut problems (such as, for example the requirement cut problem), mapping solutions between G and G' will preserve the value of the solution to within a polylog(k) factor. This strategy is the basis for the approximation algorithms designed in this paper.

But for multicommodity-type problems which depend on the congestion of certain multicommodity flows, we need a method to preserve the congestion of all multicommodity flows within a polylog(k) factor. The above theorem due to Linial, London and Rabinovich [30] and Aumann and Rabani [6] gives an $O(\log k)$-approximate min-cut max-flow relation for maximum concurrent flows, and this theorem allows us to use reductions that approximately preserve the terminal cut function to approximately preserve the congestion of all multicommodity flows too, within a worse but still polylog(k) factor.

Throughout we will use the notation that graphs G_1, G_2 (on the same node set) are "summed" by taking the union of their edge set (and allowing parallel edges).

3. Good Cut-Sparsifiers Exist.

Here we prove that there are $O(\frac{\log k}{\log \log k})$-quality cut-sparsifiers. This is a global structural result, but we prove this by introducing a zero-sum game between an extension player and a cut player. The extension player attempts to construct such a graph G', and the cut player verifies that the cut function of this graph approximates the terminal cut function. We define this game in such a way that bounding the game value of this game implies the above structural result.

We can then bound the game value of this game by proving that there is a good response for the extension player for every distribution on checks that the cut player makes. We use a rounding procedure due to Fakcharoenphol, Harrelson, Rao and Talwar [13] for the 0-extension problem to produce such a good response.

Thus, the intuition for why good quality cut-sparsifiers should exist *does not* come from looking at a graph G, and a set $K \subset V$ of terminals, and determining that there is some way to approximately preserve all these exponentially many minimum cuts on a graph G' on just the set of terminals. Rather, the intuition comes from imagining an adversary trying to disprove the statement that G has a good quality cut-sparsifier, and showing that this adversary in fact cannot disprove this statement. The beauty of the Min-Max Theorem is that this is enough to imply that good quality cut-sparsifiers exist.

3.1. 0-Extensions.

The 0-extension problem was originally formulated in [24] by Karzanov who introduced the problem as a natural generalization of the minimum multiway cut problem. Suppose we are given an undirected, capacitated graph $G = (V, E)$, $c : E \to \mathbb{R}^+$, a set of terminals $K \subset V$ and a semi-metric D on the terminals. Then the goal of the 0-extension problem is to assign each node in V to a terminal in K (and each terminal $t \in K$ must be assigned to itself) such that the sum over all edges (u, v) of $c(u, v)$ times the distance between u and v under the metric D is minimized.

Formally, the goal is to find a function $f : V \to K$ (such that $f(t) = t$ for all $t \in K$) so that $\sum_{(u,v)\in E} c(u, v) D(f(u), f(v))$ is minimized over all such functions. Then when D is just the uniform metric on the terminals K, this exactly the minimum

multiway cut problem. Karzanov gave a (semi)metric relaxation of the 0-extension problem [24]:

$$\begin{array}{ll} \min & \sum_{(u,v)\in E} c(u,v)\beta(u,v) \\ \text{s.t.} & \\ & \beta \text{ is a semi-metric on } V \\ & \forall_{t,t'\in K}\beta(t,t') = D(t,t'). \end{array}$$

Note that the semi-metric β is defined on V while D is defined only on K. Let OPT^* denote the value of an optimal solution to the above linear programming relaxation of the 0-extension problem. Clearly $OPT^* \leq OPT$. Calinescu, Karloff and Rabani [9] gave a randomized rounding procedure to round any feasible solution β of value C to a 0-extension that has expected value at most $O(\log k)C$. Fakcharoenphol, Harrelson, Rao and Talwar [13] gave an improved randomized rounding procedure that achieves an $O(\frac{\log k}{\log\log k})$ approximation ratio:

THEOREM 3.1. *[13]*

$$OPT^* \leq OPT \leq O\left(\frac{\log k}{\log\log k}\right) OPT^*$$

Given a function $f : V \to K$ such that $f(t) = t$ for all $t \in K$, we can define the capacitated graph G_f on K that results from the function f as:

$$c_f(a,b) = \sum_{u,v|f(u)=a,f(v)=b} c(u,v)$$

We will abuse notation and refer to the graph G_f generated by f as a 0-extension of the graph G. We will use h_f to denote the cut function of the resulting graph G_f. Then in fact, for any 0-extension f, the graph G_f is a cut-sparsifier:

CLAIM 1. *For any subset of terminals $U \subset K$,*

$$h_K(U) \leq h_f(U).$$

Proof. For any $u \in K$, let $f^{-1}(u) = \{a \mid f(a) = u\}$. Let $A = \cup_{u\in U} f^{-1}(u)$. By definition, $h_f(U) = h(A)$ and because $A \cap K = U$ this implies that $h(A) \geq h_K(U)$, because the cut $A, V - A$ is a cut separating U from $K - U$ so it is at least the minimum cut-separating U from $K - U$. □

We will use the above theorem due to Fakcharoenphol, Harrelson, Rao and Talwar to show that, existentially, there is a graph G' such that for all $U \subset K$ $h_K(U) \leq h'(U) \leq O(\frac{\log k}{\log\log k})h_K(U)$. In fact, this graph will be a convex combination of 0-extensions of G. Since for any 0-extension f, the graph G_f is a cut-sparsifier this implies that all graphs that can be realized as a convex combination of graphs generated by 0-extensions are also cut-sparsifiers. So all that remains is to show that there is some distribution γ on 0-extensions f for which the resulting average graph $G' = \sum_f \gamma(f)G_f$ is never too much larger.

3.2. A Zero-Sum Game. Here we introduce and analyze an appropriately chosen zero-sum game, so that a bound on the game value of this game will imply the desired structural graph theory result.

Given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals, an extension player (P1) and a cut player (P2) play the following zero-sum game that we will refer to as the extension-cut game:

The **extension player** (P1) chooses a 0-extension $f : V \to K$ such that $f(t) = t$ for all terminals t

The **cut player** (P2) chooses a cut from 2^K

Given a strategy f for P1 and a strategy A for P2, P2 wins $\frac{1}{h_K(A)}$ units for each unit of capacity crossing the cut $(A, K - A)$ in P1's 0-extension. Also, we restrict P2 to play only strategies A for which $h_K(A) \neq 0$. So if P1 plays a strategy f and P2 plays a strategy A then P2 wins:

$$N(f, A) = \sum_{(u,v)\in E} \frac{1_{(f(u),f(v))\in\delta_K(A)} c(u, v)}{h_K(A)}$$

Definition 3.2. *Let ν denote the game value of the extension-cut game.*

Using von Neumann's Min-Max Theorem, we can bound the game value by bounding the cost of P1's best response to any fixed, randomized strategy for P2. So consider any randomized strategy μ for P2. μ is just a probability distribution on 2^K. We can define an ℓ_1 metric on K:

$$D_\mu(t, t') = \sum_{A\subset K} \mu(A) \frac{1_{(t,t')\in\delta_K(A)}}{h_K(A)}$$

D_μ is just a weighted sum of cut-metrics on K. Given D_μ we can define a semi-metric β that is roughly consistent with D_μ. This semi-metric will serve as a feasible solution to the linear programming relaxation for the 0-extension problem. A bound on the cost of this feasible solution will imply that there is a 0-extension that has not too much cost, and this will imply that the extension player has a good response to the strategy μ. We define β as:

Initially set all edge distances $d(u, v)$ to zero. Then for each $A \subset K$, if there is no unique minimum cut separating A and $K - A$, choose one such minimum cut arbitrarily. For this minimum cut, for each edge (u, v) crossing the cut, increment the distance $d(u, v)$ by $\frac{\mu(A)}{h_K(A)}$.

Then let β be the semi-metric defined as the shortest path metric on G when distances are $d(u, v)$.

Claim 2. *$\beta(t, t') \geq D_\mu(t, t')$ for all terminals t, t'.*

Proof. Consider any particular pair of terminals t, t'. Consider the contribution of any particular $A \subset K$ to the semi-metric D_μ. A only contributes if t and t' are separated with respect to A - i.e. if exactly one of $\{t, t'\}$ is in A. We will show that any set A that contributes to D_μ, has at least as large a contribution to β. So consider a set A that contains t but not t'. Then any path in G from t to t' must cross the cut in G that actually achieves the minimum $h_K(A)$. But when considering the set A, we increased the distance on all edges crossing this cut by $\frac{\mu(A)}{h_K(A)}$, so the contribution to $\beta(t, t')$ of the set A is at least as large as the contribution of A to $D_\mu(t, t')$ and this implies the claim. ∎

CLAIM 3. *β is an ℓ_1 semi-metric*

Proof. Using the description of β, one can easily write this semi-metric as a linear combination of cut-metrics on the set V, and this implies the claim. □

CLAIM 4. $\sum_{(u,v)\in E(G)} \beta(u,v)c(u,v) = 1$

Proof. Again, for each set $A \subset K$ consider the total distances $\times$ capacity units that are allocated when we increase the distances of all edges crossing the cut that achieves $h_K(A)$ by $\frac{\mu(A)}{h_K(A)}$. We know that the total capacity crossing this cut is $h_K(A)$ and for each such edge, the distance on that edge (according to β) is incremented by $\frac{\mu(A)}{h_K(A)}$. So this implies that the total contribution of the set $A \subset K$ to the total distance $\times$ capacity units is $\mu(A)$ and if we sum over all A we get the desired claim, because μ is a probability distribution. □

THEOREM 3.3.

$$\nu \leq O\left(\frac{\log k}{\log\log k}\right)$$

Proof. Using Theorem 3.1, there exists a 0-extension $f : V \to K$ (such that $f(t) = t$ for all terminals t) and such that

$$\sum_{(u,v)\in E} c(u,v)\beta(f(u),f(v)) \leq O\left(\frac{\log k}{\log\log k}\right)$$

Then suppose P1 plays such a strategy f:

$$\begin{aligned} &E_{A\leftarrow\mu}[N(f,A)] \\ &\quad= \sum_{(u,v)\in E}\sum_{A} \frac{1_{(f(u),f(v))\in\delta_K(A)}c(u,v)\mu(A)}{h_K(A)} \\ &\quad= \sum_{(u,v)\in E} c(u,v)D_\mu(f(u),f(v)) \\ &\quad\leq \sum_{(u,v)\in E} c(u,v)\beta(f(u),f(v)) \leq O\left(\frac{\log k}{\log\log k}\right) \end{aligned}$$

□

We can improve this result by noticing that the semi-metric D_μ corresponding to the cost function associated with the randomized strategy μ for P2 and the feasible solution β that we produced are both ℓ_1. In particular, let $G = (V, E)$ be an undirected, capacitated graph and let $K \subset V$ be a subset of terminals. Let η denote a bound on the maximum integrality gap of the semi-metric relaxation when the semi-metric D is ℓ_1 and the feasible solution to the linear program β is also ℓ_1.

COROLLARY 3.4. $\nu \leq \eta$

Also, we can use a rounding procedure due to Calinescu, Karloff and Rabani [9] that improves upon the one in [13] in the case in which G excludes a fixed minor.

THEOREM 3.5. *[9] If G excludes $K_{r,r}$ as a minor then*

$$OPT^* \leq OPT \leq O(r^2)OPT^*$$

The bound of $O(r^3)$ is implicit in [9], and can be immediately improved to $O(r^2)$ using the results of [15] (which provide improved bounds for padded decompositions for graphs excluding $K_{r,r}$ over those in [25]). Then this immediately implies as a corollary:

COROLLARY 3.6. *For any capacitated graph $G = (V, E)$ that excludes $K_{r,r}$ as a minor,*

$$\nu \leq O(r^2).$$

We can immediately use any bound on the game value to obtain the desired structural result:

THEOREM 3.7. *For any capacitated graph $G = (V, E)$, for any set $K \subset V$ of $|K| = k$ terminals, there is an ν-quality cut-sparsifier $G' = (K, E')$ and in fact such a graph can be realized as a convex combination of graphs G_f generated by 0-extensions f.*

Proof. We can again apply von Neumann's Min-Max Theorem, and get that there exists a distribution γ on 0-extensions ($f : V \to K$ s.t. $f(t) = t$ for all $t \in K$) such that for all $A \subset K$: $E_{f \leftarrow \gamma}[N(f, A)] \leq \nu$.

For any 0-extension f, let G_f be the corresponding 0-extension of G generated by f, and let $h_f : 2^K \to \mathbb{R}^+$ be the cut function defined on this graph. Further, let $G' = \sum_{f \in supp(\gamma)} \gamma(f) G_f$. Then for any $A \subset K$:

$$h_K(A) \leq h_f(A) \text{ and } h_K(A) \leq h'(A) = \sum_{f \in supp(\gamma)} \gamma(f) h_f(A)$$

Also because $E_{f \leftarrow \gamma}[N(f, A)] \leq \nu$:

$$\sum_{(u,v) \in E} \sum_{f \in supp(\gamma)} \gamma(f) \frac{1_{(f(u), f(v)) \in \delta_K(A)} c(u, v)}{h_K(A)} = \frac{1}{h_K(A)} \sum_{f \in supp(\gamma)} \gamma(f) h_f(A) = \frac{h'(A)}{h_K(A)}$$

and so for any $A \subset K$: $h'(A) \leq \nu h_K(A)$. ∎

In particular, in general there are $O(\frac{\log k}{\log \log k})$-quality cut-sparsifiers. If G excludes $K_{r,r}$ as a minor, then there is an $O(r^2)$-quality cut-sparsifier. And if η is an upper bound on the maximum integrality gap of the 0-extension LP when both the semi-metric Δ and the feasible solution D are required to be ℓ_1, then there is an η-quality cut-sparsifier.

We will eventually use this existential result to prove that a certain polytope (with exponentially many constraints) is feasible, and then use entirely different techniques to find a point in the polytope and to constructively find such a graph G' that approximates the terminal cut function to within polylog(k).

4. Applications to Oblivious Routing. Here we give a sample application of cut-sparsification to a question of oblivious routing. We give additional applications to graph partitioning and graph layout problems in §6. But the application we give here is somewhat different in that we do not require a constructive result for cut-sparsifiers. Rather, the existential result proven in §3 is enough to guarantee a good oblivious routing scheme exists (for the variant of the problem we consider) and we can use an alternative method due to [7] to actually construct the oblivious routing scheme.

More generally, this section gives a pattern for how to apply our results on cut-sparsification to questions about the congestion of multicommodity flows. In particular, we can use the approximate relationship between generalized sparsest cut and maximum concurrent flow to re-write multicommodity flow questions as cut questions and this in turn implies that our cut-sparsifiers also preserve the value of all maximum concurrent flow problems to within a polylog(k) factor.

4.1. A Variant of Oblivious Routing. An *oblivious routing scheme* makes routing decisions based only on the starting and ending nodes of a routing request (and independently of the current load in the network and what other routing requests have been made). So routing decisions are based only on local knowledge, and consequently an oblivious routing scheme can be easily implemented in a distributed manner. We consider the routing goal of minimizing the congestion in a network - i.e. minimizing the maximum ratio of load to capacity on any edge.

Then the competitive ratio of an oblivious routing scheme is measured by how well the oblivious routing scheme performs (with respect to congestion) compared to the performance of the optimal routing scheme with fore-knowledge of what demands need to be routed. Valiant and Brebner [39] were the first to prove any performance guarantees for oblivious routing on any network topology, and gave an $O(\log n)$-competitive oblivious routing algorithm for the hypercube (on n nodes). In a breakthrough paper, Räcke [35] proved that for arbitrary graphs, there are oblivious routing algorithms that are polylog(n)-competitive. Recently, Räcke [36] proved that there are in fact $O(\log n)$-competitive oblivious routing algorithms for general graphs. This performance guarantee even matches the competitive ratio of (optimal) adaptive routing algorithms that are given updated knowledge of the loads in the network before being asked to route each successive request!

But consider a related problem: Suppose a service provider is only responsible for routing requests between some small set of clients. In this setting, an oblivious routing protocol can make the assumption that all routing requests will have endpoints in some small set of terminals K. Suppose that $|K| = k$, and that $k^2 \ll O(\log n)$. These routing requests are certainly allowed to use paths that contain nodes in the entire (web)graph. In this restricted scenario, how well can oblivious routing perform? We could ignore this promise that all routing requests will be between terminals in K, and we can use the oblivious routing scheme in [36] and get an oblivious routing scheme that is $O(\log n)$-competitive, and still based only on local knowledge. But if $k^2 \ll O(\log n)$, then this is a trivial guarantee: For each pair (a, b) of terminals, we could alternatively find a minimum congestion routing of a unit flow from a to b. Then a naive union bound over all $\binom{k}{2}$ pairs yields a performance guarantee that outperforms the $O(\log n)$-competitive oblivious routing guarantee!

But here, an oblivious routing protocol (which only needs to pre-define a routing for all pairs of terminals) that can achieve a polylog(k)-competitive ratio would provide a much more powerful guarantee in this practical scenario. This would provide a competitive guarantee that is polylogarithmic in the number of clients, and not logarithmic in the number of nodes in the entire (web)graph. Using this example as a guide, getting approximation guarantees for multicommodity-flow and cut problems that only depend (poly-logarithmically) on the number of interesting nodes can be crucial in making powerful theoretical algorithms powerful in practice too.

4.2. Existential Results. Suppose we are given a capacitated, undirected graph G and a subset $K \subset V$ of size k. Suppose also that we are promised all the demands we will be asked to route will have both endpoints in K. Here we prove that for

this problem there is an oblivious routing scheme that is $O(\frac{\log^3 k}{\log\log k})$-competitive. In the next subsection, we also give a polynomial (in n and k) time algorithm for constructing such schemes. There are many previously known oblivious routing schemes, but if $k^2 \ll \log n$ then these schemes cannot beat the trivial $\binom{k}{2}$ competitive ratio resulting from choosing $\binom{k}{2}$ independent minimum congestion unit flows (one for each $a, b \in K$).

Let G' be a convex combination of 0-extensions of G, and suppose that for all $A \subset K$:

$$h_K(A) \leq h'(A) \leq O\left(\frac{\log k}{\log\log k}\right) h_K(A)$$

Here we consider G' to be a demand graph on the terminals K.

LEMMA 4.1. *The demands in G' can be routed in G with congestion at most $O(\frac{\log^2 k}{\log\log k})$.*

Proof. Let $T \subset V$ be arbitrary. Let $A = T \cap K$ and also let $d(T)$ be the demands in G' that cross the cut $(T, V - T)$ .Then $h(T) \geq h_K(A)$. So

$$\Omega\left(\frac{\log\log k}{\log k}\right) \leq \frac{h_K(A)}{h'(A)} \leq \frac{h(T)}{d(T)}$$

and this holds for all $T \subset V$, so the sparsity of a cut in G (when demands are given by G') is at least

$$\Omega\left(\frac{\log\log k}{\log k}\right)$$

Using the Theorem 2.3 due to Linal, London and Rabinovich [30] and Aumann and Rabani [6], this implies that all the demands in G' can be routed using congestion at most

$$O\left(\frac{\log^2 k}{\log\log k}\right)$$

□

We also note that G' is a better communication network than G:

LEMMA 4.2. *Any set of demands (which have support only in K) that can be routed with congestion at most C, can also be routed in G' with congestion at most C.*

Proof. We have that

$$G' = \sum_{f \in supp(\gamma)} \gamma(f) G_f$$

and each G_f is a 0-extension of G. So given a flow that satisfies the demands and achieves a congestion of at most C in G, we can take a flow path decomposition of this flow. Then consider any path in the flow decomposition and suppose that this flow path carries δ units of flow. Decompose this path into subpaths that connect nodes in K and contain no nodes of K as internal nodes. For each such subpath, suppose that the subpath connects a and b in K, then add δ units of flow along the

edge (a, b) in G_f. This scheme will satisfy all demands because the original flow paths in G satisfied all demands. And also, each edge in G_f will have congestion at most C because the edges in G_f are just a subset of the edges in G and each edge in G_f is assigned exactly the same total amount of flow as it is in the flow in G.

So for each $f \in supp(\gamma)$, route $\gamma(f)$ fraction of all demands according to the routing scheme given for G_f above. The contribution to the congestion of any edge $(u, v) \in E'$ from any f is at most $\gamma(f)C$, and so the total congestion on any edge is at most C. Yet all demands are met because $\sum_{f \in supp(\gamma)} \gamma(f) = 1$. □

We can now construct an oblivious routing scheme in G' and compose this with the embedding of G' into G to get an oblivious Steiner routing scheme in G:

THEOREM 4.3. *[36] There is an oblivious routing scheme for G' (on k nodes) that on any set of demands incurs congestion at most $O(\log k)$ times the off-line optimum.*

THEOREM 4.4. *There is an oblivious Steiner routing scheme that on any set of demands (supported in K) incurs congestion at most $O(\frac{\log^3 k}{\log \log k})$ times the off-line optimum.*

Proof. Given G', use Räcke's Theorem [36] to construct an oblivious routing scheme in G'. This can be mapped to an oblivious routing scheme in G using the existence of a low-congestion routing for the demand graph G' in G: Given $a, b \in K$, if the oblivious routing scheme in G' assigns δ units of flow to a path $P_{a,b}$ in G', then construct a set of paths in G that in total carry δ units of flow as follows:

Let $P_{a,b} = (a, p_1), (p_1, p_2), ...(p_l, b)$. Let $p_0 = a$ and $p_{l+1} = b$. Then consider an edge (p_i, p_{i+1}) contained in this path and suppose that $c'(p_i, p_{i+1})$ is α in G'. Then for each flow path P connecting p_i to p_{i+1} in the low-congestion routing of G' in G, add the same path and multiply the weight by $\frac{\delta}{\alpha}$. The union of these flow paths sends δ units of flow from a to b in G. Räcke's oblivious routing scheme sends one unit of flow from a to b for all $a, b \in K$ in G'. So this implies that we have constructed a set of flows in G such that for all $a, b \in K$, one unit of flow is sent from a to b in G.

So consider any set of demands that have support contained in K. Suppose that this set of demands can be routed in G with congestion C. Then there exists a flow satisfying these demands that can be routed in G' with congestion at most C using Lemma 2. Räcke's oblivious routing guarantees imply that the oblivious routing scheme in G' incurs congestion at most $O(\log k)C$ on any edge in G'. This implies that we have scaled up each edge in G' by at most $O(\log k)C$ and so we have scaled up the amount of flow transported on each path in an optimal routing of the (demand) graph G' into G by at most $O(\log k)C$. So the congestion incurred by this oblivious routing scheme is at most

$$O\left(\frac{\log^3 k}{\log \log k}\right) C$$

□

This result is non-constructive, because the proof of Theorem 3.7 is non-constructive.

4.3. Constructive Results. Azar et al [7] formulate the problem of deciding (for a given graph G) whether there exists an oblivious routing scheme that is T-competitive against the off-line optimal algorithm as a linear program. This algorithm can be adapted to yield:

THEOREM 4.5. *An optimal oblivious Steiner routing scheme can be constructed in polynomial (in n and k) time.* So an $O(\frac{\log^3 k}{\log \log k})$-competitive oblivious Steiner routing scheme can be constructed in polynomial time.

5. Constructing a Vertex Sparsifier. In this section we consider the problem of constructing - in time polynomial in n and k - an undirected, capacitated graph $G' = (K, E')$ for which the cut function h' approximates the terminal cut function h_K. We will use existential results (in Theorem 3.7) to conclude that a particular polytope is non-empty, and we will design approximate separation oracles for this polytope to give a polynomial time construction for finding such a graph G'.

Rather surprisingly, we use oblivious routing guarantees to design an approximate separation oracle. So apart from the original motivation for studying oblivious routing schemes, we actually use oblivious routing *to solve an optimization problem.* These ideas lead us to believe that the remarkable oblivious routing guarantees due to Räcke [36] can also be understood as a geometric phenomenon particular to undirected multicommodity polytopes.

The constructive results in this section have been subsequently improved in [31], [11] and [12], which independently give three separate techniques for efficiently constructing cut-sparsifiers that match the existential results in Theorem 3.7. However, these techniques do not improve upon the approximate separation oracles constructed in this section, and our geometric interpretation of oblivious routing may be of independent interest.

5.1. The Terminal Cut Polytope. Constructing such a graph G' can be naturally represented as a feasibility question for a linear program. We can define a non-negative variable $x_{a,b}$ for each pair $a, b \in K$. Then finding a G' for which the cut function h' $f(k)g(k)$-approximates the terminal cut function is equivalent to finding a feasible point in the polytope:

$$\begin{array}{lll} \textbf{Type 1:} & \sum_{(a,b)\in\delta_K(A)} x_{a,b} \leq f(k) h_K(A) & \text{for all } A \subset K \\ \textbf{Type 2:} & h_K(A) \leq g(k) \sum_{(a,b)\in\delta_K(A)} x_{a,b} & \text{for all } A \subset K \\ & 0 \leq x_{a,b} & \text{for all } a, b \in K \end{array}$$

Theorem 3.7 implies that this polytope is non-empty for $f(k) = O(\frac{\log k}{\log\log k})$, $g(k) = 1$. However there are 2^{k+1} linear constraints, and we cannot check all constraints in time polynomial in n and k. We will construct approximate separation oracles for both Type 1 and Type 2 Inequalities. Then we can use the ellipsoid algorithm to find feasible edge weights. And we will choose $f(k)$ and $g(k)$ to be polylogarithmic in k.

LEMMA 5.1. *There is a polynomial time algorithm to find a Type* 1 *Inequality that is within an* $O(\sqrt{\log k}\log\log k)$ *factor approximately the maximally violated Type* 1 *Inequality.*

Proof. Given non-negative edge weights $x_{a,b}$ we can consider the problem of (approximately) minimizing

$$\frac{h_K(A)}{h'(A)}$$

over all sets $A \subset K$. This is exactly the sparsest cut problem when the graph G' (with edge weights $x_{a,b}$) is considered to be the demand graph and we are attempting to route this demand with low congestion in G. Then we can use the current best approximation algorithm to sparsest cut due to [4] which is an $O(\sqrt{\log k}\log\log k)$ approximation algorithm to this problem, and we will find a set B for which

$$\frac{h_K(B)}{h'(B)} \leq O(\sqrt{\log k}\log\log k)\min_{A\subset K}\frac{h_K(A)}{h'(A)}$$

And so the Type 1 Inequality for the set B is within an $O(\sqrt{\log k}\log\log k)$ factor approximately the maximally violated Type 1 Inequality, and we can find such a set constructively (with high probability). □

We will use (constructive) algorithms for oblivious routing to find an approximately maximally violated Type 2 Inequality. Suppose there is a Type 2 constraint that is violated by a factor OPT. Then there exists a terminal cut A such that $h_K(A) \geq OPTh'(A)$.

LEMMA 5.2. *There exists a maximum concurrent flow $\vec{f}$ that can be routed with congestion 1 in G, but cannot be routed with congestion less than OPT in G'.*

Proof. Place a super-source s and connect s via infinite capacity (directed) edges to each node in A. Also place a super-sink t and connect each node in $K - A$ via an infinite capacity (directed) edge to t. Compute a maximum $s - t$ flow. The value of this flow is $h_K(A)$. So choose a maximum concurrent flow problem $\vec{f}$ in which $f_{a,b}$ is just the amount of a to b flow in a path decomposition of the above maximum flow. In particular, $f_{a,b} = 0$ if a and b are either both in A or both in $K - A$. This flow can clearly be routed in G with congestion at most 1, because we constructed this demand vector from such a routing.

However because $\frac{h'(A)}{h_K(A)} \leq \frac{1}{OPT}$ there is a cut of sparsity $\frac{1}{OPT}$ and so $\vec{f}$ cannot be routed with congestion less than OPT in G'. □

Hence we consider the problem of finding a demand vector $\vec{f}$ that can be routed with congestion 1 in G, but cannot be routed with congestion $\leq O(\log k)g(k)$ in G'. Given such a demand vector $\vec{f}$, we can find a cut of sparsity at most $\leq \frac{1}{g(k)}$ in G' i.e. we can find a cut $A \subset K$ for which

$$\frac{h'(A)}{d(A)} \leq \frac{1}{g(k)}$$

where $d(A)$ is the total demand in $\vec{f}$ with one endpoint in A and one in $K - A$. Because $\vec{f}$ can be routed with congestion 1 in G, we are guaranteed:

$$\frac{h_K(A)}{d(A)} \geq 1$$

So this implies that we have found a Type 2 Inequality that is violated. So using Lemma 4 and the above argument, up to an $O(\log k)$ factor, the problem of finding an (approximately) maximally violated Type 2 Inequality is equivalent to finding a demand vector $\vec{f}$ that can be routed with congestion at most 1 in G, and maximizes the minimum congestion needed to route this demand vector in G'. We define this problem formally as the *Max-Min Congestion Problem*, and we give a polylog(k) approximation algorithm for this problem:

Formally, given an undirected, capacitated graph $G = (V, E)$, a subset $K \subset V$ of size k, and an undirected, capacitated graph $G' = (K, E')$ the goal of the Max-Min Congestion Problem is to find a demand vector $\vec{f}$ (such that the demands are supported in K) that can be routed with congestion 1 in G and maximizes the minimum congestion needed to route $\vec{f}$ in G'. We given an $O(\log k)$ approximation algorithm for this problem in §5.3, and use this approximation algorithm to construct an $O(\log^{3.5} k)$-quality cut-sparsifier in the next subsection.

5.2. Constructing a Vertex Sparsifier. In this subsection, we give an approximate separation oracle for the set of Type 2 Inequalities (based on the approximation algorithm given in §5.3 for the Max-Min Congestion Problem). We use this approximate separation oracle to construct an $O(\log^{3.5} k)$-quality cut-sparsifier by running the ellipsoid algorithm on a linear program on $\binom{k}{2}$ variables.

LEMMA 5.3. *There is a polynomial time algorithm to find a Type* 2 *Inequality that is within an* $O(\log^2 k)$ *factor approximately the maximally violated Type* 2 *Inequality.*

Proof. Let A be the maximally violated Type 2 Inequality. Let $OPT = \frac{h_K(A)}{h'(A)}$. Then using Lemma 5.2 there is a maximum concurrent flow demand vector $\vec{d}$ such that $\vec{d}$ can be routed with congestion at most 1 in G and cannot be routed with congestion $< OPT$ in G'. So the solution to the Max-Min Congestion Problem is at least OPT, which implies that using the approximation algorithm for the Max-Min Congestion Problem given in §5.3 we will find a demand vector $\vec{d}$ which can be routed in G with congestion at most 1 and cannot be routed in G' with congestion $< \frac{OPT}{O(\log k)}$.

Hence we can find a cut $A \subset K$ for which

$$\frac{h'(A)}{d(A)} < \frac{O(\log^2 k)}{OPT} \text{ and } \frac{h_K(A)}{d(A)} \geq 1$$

and this implies

$$\frac{h_K(A)}{h'(A)} > \frac{OPT}{O(\log^2 k)}.$$

□

Returning to the linear program for finding such a G', we can use the ellipsoid algorithm to find (in time polynomial in n and k) a point $\vec{x} \geq 0$ such that the graph G' defined by these edge weights satisfies:

$$\frac{h_K(A)}{g(k)} \leq \sum_{a,b s.t. a \in A, b \notin A} x_{a,b} \leq f(k) h_K(A)$$

for $g(k) = O(\log^2 k)$ and $f(k) = \log^{1.5} k$.

THEOREM 5.4. *For any capacitated graph* $G = (V, E)$*, for any set* $K \subset V$ *of* $|K| = k$ *terminals, there is a polynomial (in* n *and* k*) time algorithm to construct an* $O(\log^{3.5} k)$*-quality cut-sparsifier.*

Note that such a cut-sparsifier is *not* guaranteed to be realizable as a convex combination of graphs generated by 0-extensions.

5.3. An Approximation Algorithm via Oblivious Routing. In this subsection, we give an approximation algorithm for the Max-Min Congestion Problem.

THEOREM 5.5. *There exists a polynomial time* $O(\log k)$*-approximation algorithm for the Max-Min Congestion Problem.*

We first construct an $O(\log k)$-competitive oblivious routing scheme f' for G'. Such an oblivious routing scheme is guaranteed to exist, and can be found in polynomial time using the results due to Räcke [36]. For each $a, b \in K$, f' specifies a unit flow from a to b in G' and we will let $f'^{a,b} : E' \to \mathbb{R}^+$ be the corresponding assignment of flows to edges for this unit flow. Since f' is $O(\log k)$-competitive, f' has the property

that for any demand vector $\vec{d}$, the congestion that results from routing according to f' is within an $O(\log k)$ factor of the optimal congestion for routing $\vec{d}$ in G'.

For any edge $(u,v) \in E'$, we consider the following linear program $LP(u,v)$:

$$\begin{aligned}
\max D_{u,v} \quad & = \frac{\sum_{a,b} d_{a,b} f'^{a,b}(u,v)}{c(u,v)} \\
\text{s.t.} \quad & \\
& \textstyle\sum_{t \in U, t \neq a} x^{a,b}(a,t) = d_{a,b} && \text{for all } a,b \\
& \textstyle\sum_{t \in U} x^{a,b}(s,t) = 0 && \text{for all } s \in U, s \neq a,b \\
& \textstyle\sum_{a,b} x^{a,b}(e) \leq c(e) && \text{for all } e \in E \\
& x^{a,b}(e) \geq 0 && \text{for all } e \in E
\end{aligned}$$

The interpretation of this linear program is that it finds a demand vector ($d_{a,b}$ for all $a,b \in K$) that can be routed as a multicommodity flow in G, which maximizes the congestion of the oblivious routing scheme on the edge $(u,v) \in E'$ among all such flows. We will solve the above linear program for all $(a,b) \in E'$ and output the demand vector $\vec{d}$ that achieves the maximum $D_{a,b}$ over all $(a,b) \in E'$. Let

$$D = \max_{(a,b) \in E'} D_{a,b}$$

LEMMA 5.6. *Let $\vec{d}$ be the output, then $\vec{d}$ can be routed with congestion at most* 1 *in G and cannot be routed with congestion $< \frac{D}{O(\log k)}$ in G'.*

Proof. The linear program enforces that $\vec{d}$ can be routed in G with congestion at most 1. Suppose $\vec{d}$ achieves value D on an edge (i,j) -i.e. $(i,j) = \arg\max_{(a,b)} D_{a,b}$ and $\vec{d}$ is the optimizing demand. Then $\vec{d}$ achieves congestion D on edge (i,j) when routed according to the oblivious routing scheme. The oblivious routing guarantees for G' imply that no routing of $\vec{d}$ achieves congestion smaller than

$$\frac{D}{O(\log k)}$$

□

Let OPT be the optimal value for the Max-Min Congestion Problem.

LEMMA 5.7. *There exists a feasible demand $\vec{d}$ which achieves $D_{a,b} \geq OPT$ for some $(a,b) \in E'$.*

Proof. Let $\vec{d'}$ be the demand that achieves the optimal value for the Max-Min Congestion Problem. The oblivious routing scheme is a routing, and must then achieve congestion at least OPT on some edge $(a,b) \in E'$ (not necessarily the same edge that achieves congestion OPT in the optimal off-line routing scheme for $\vec{d'}$ in G'). □

Proof. This implies Theorem 5.5, and the vector $\vec{d}$ computed by solving a polynomial (in k) number of linear programs will be an $O(\log k)$ approximation for the Max-Min Congestion Problem. □

6. Oblivious Reductions. Here we demonstrate two additional techniques for how to apply our results. The pattern for applying cut-sparsifiers is always the same: Construct a good-quality cut-sparsifier, and then run a pre-existing approximation algorithm on the cut-sparsifer, and map the solution back to the original graph G. If the optimization problem can be approximately written as depending only on the

terminal cut function h_K, then this technique can be used to reduce the question of designing a polylog(k) approximation algorithm to the question of designing a polylog(n) approximation algorithm. In fact, this general technique is *oblivious* to the actual optimization problem that we want to solve, as long as it can be approximately written as depending only on the terminal cut function. However, there is an art to re-writing an optimization problem in a form that only depends on the terminal cut function, and here we give two such approaches (in addition to the approach we gave in §4 for re-writing multicommodity flow problems). The two approaches we give here are useful for re-writing graph partitioning problems and graph layout problems. In particular these techniques apply to any optimization problems that can be written either as a cost-minimization problem over some feasible set of partitions, or to ay cost-minimization problem over a laminar family of cuts, can be reduced to a uniform case using cut-sparsifiers.

6.1. Applications to Partitioning Problems. Here we give some sample applications that demonstrate how to reduce graph-partitioning problems using cut-sparsifiers to a uniform case. In particular, by approximately preserving the minimum cut separating any subset of terminals A from the remaining terminals $K - A$ we also approximately preserve the cost of any partition $A_1, A_2, ..A_r$ of K, not just for bi-partitions. We use this observation to give an improved approximation (independent of the size of the graph) for the Requirement Cut Problem.

Given an undirected, capacitated graph $G = (V, E)$ and g groups of nodes $X_1, ...X_g \subset V$, each group X_i is assigned a requirement $r_i \in \{0, ...|X_i|\}$. Then the goal of the requirement cut problem is to find a minimum capacity set of edges whose removal separates each group X_i into at least r_i disconnected components.

[33] gives an $O(\log n \log gR)$ approximation algorithm for this problem, where R is the maximum requirement $\max_i r_i$. Then given an instance of the requirement cut problem in which $X_1 \cup X_2 ... \cup X_g = K$ and $|K| = k$, we can use Theorem 5.4 to reduce to a uniform case. Let OPT be the value of the optimal solution in G. We denote the optimal solution in G' as OPT'.

Claim 5.

$$OPT' \leq O(\log^{3.5} k) OPT$$

Proof. Interpret the optimal solution to the requirement cut problem in G as a partition $P = \{P_1, P_2, ...P_r\}$ of K that satisfies the requirement cut - i.e. for all i, the nodes in X_i are contained in at least r_i elements of the partition P. Then

$$OPT \geq \frac{1}{2} \sum_i h_K(P_i)$$

and $P = \{P_1, P_2, ...P_r\}$ is a valid partition for the requirement cut problem mapped to G' so

$$OPT' \leq \sum_i h'(P_i) \leq O(\log^{3.5} k) \sum_i h_K(P_i)$$

□

Note that Steiner generalization of sparsest cut, min-bisection, ρ-separator, also satisfy this type of reducibility property.

THEOREM 6.1. *There is a polynomial (in n and k) time $O(\log^{4.5} k \log gR)$-approximation algorithm for the requirement cut problem.*

Proof. Construct G' as in Theorem 5.4 and we can run the approximation algorithm due to [33] to find a set of edges of capacity at most $C \leq O(\log k \log gR)OPT'$ deleting which (in G') results in a partition $P' = \{P'_1, P'_2, ...P'_q\}$ that satisfies the requirement cut. Then

$$\sum_i h'(P'_i) = 2C$$

For each i, define F_i as a set of edges in G that achieves $h_K(P'_i)$ and separates P'_i and $K - P'_i$. Delete all edges in $F = F_1 \cup F_2 ... \cup F_q$, and this results (in G) in a sub-partition P'' of P' that also satisfies the requirement cut and the capacity of these edges is at most $2C$. □

An almost identical argument that uses the result due to [19] implies:

COROLLARY 6.2. *There is a polynomial (in n and k) time $O(\log^{4.5} k)$-approximation algorithm for the l-multicut problem.*

Note that here k is the number of demand pairs. Previous approximation algorithms for these problems [33], [19] and later [36] all rely on a decomposition tree for the graph G that approximates the cuts and such a decomposition tree cannot approximate cuts better than the $\Omega(\log n)$ lower bound for oblivious routing. But we were able to use a black-box reduction to the uniform case to get an approximation guarantee that is polylog(k).

6.2. Applications to Layout Problems. Here we demonstrate how an uncrossing argument can be used to re-write graph layout problems as graph partitioning problems. This observation allows us to re-write Steiner generalizations of Minimum Cut Linear Arrangement and Minimum Linear Arrangement as graph partitioning problems and in turn allows us to give approximation guarantees independent of the size of the graph for these problems as well.

The Minimum Cut Linear Arrangement Problem is defined as: Given an undirected, capacitated graph $G = (V, E)$ we want to find an ordering of the vertices $v_1, v_2, ...v_n$ which minimizes the value of

$$C = \max_{1 \leq i \leq n} h(\{v_1, v_2, ...v_i\})$$

We can define a natural generalization of this problem in which we are given a set $K \subset V$ of size k, and we want to find an ordering of the nodes in K, $u_1, u_2, ...u_k$ and a partition $A_1, A_2, ...A_k$ of the remaining nodes $V - K$ (and let $B_i = A_i \cup \{u_i\}$) which minimizes the value of

$$C_K = \max_{1 \leq i \leq k} h(\cup_{1 \leq j \leq i} B_i)$$

We refer to this problem as the *Steiner Min-Cut Linear Arrangement Problem*. We can also give an identical generalization of Minimum Linear Arrangement to the *Steiner Minimum Linear Arrangement Problem*.

Applying our generic reduction procedure to these problems will require a more intricate uncrossing argument (that relies on the sub-modularity of the cut function) to map a solution in G' back to a solution in G. But the reduction procedure is quite

robust, and will work in this setting too: Suppose that the optimal solution to the Steiner Min-Cut Linear Arrangement Problem has value OPT. Again we construct a graph G' as in Theorem 5.4. Let OPT' be the value of an optimal solution to the min-cut linear arrangement problem on G'.

Claim 6.

$$OPT' \leq O(\log^{3.5} k)OPT$$

Proof. Suppose that the optimal solution to the Steiner Min-Cut Linear Arrangement Problem in G has an ordering $u_1, u_2, ...u_k$ of the nodes in K. Then consider this ordering as a solution to the min-cut linear arrangement problem in G'.

Each set $\cup_{1\leq j\leq i} B_i$ defines a cut in G that separates $\{u_1, u_2, ...u_i\}$ from $\{u_{i+1}, u_{i+2}, ...u_k\}$. So $h_K(\{u_1, u_2, ...u_i\}) \leq h(\cup_{1\leq j\leq i} B_i)$ and so $h'(\{u_1, u_2, ...u_i\}) \leq O(\log^{3.5} k)h(\cup_{1\leq j\leq i} B_i)$ and this is true for all $\{u_1, u_2, ...u_i\}$.

Because for all $\{u_1, u_2, ...u_i\}$, $h(\cup_{1\leq j\leq i} B_i) \leq OPT$ this implies that for all $\{u_1, u_2, ...u_i\}$

$$h'(\{u_1, u_2, ...u_i\}) \leq O(\log^{3.5} k)OPT$$

$$OPT' \leq \max_{1\leq i\leq k} h'(\{u_1, u_2, ...u_i\}) \leq O(\log^{3.5} k)OPT$$

□

Claim 7. *Suppose we can find a min-cut linear arrangement of G' of value C'. Then we can find a solution to the Steiner Min-Cut Linear Arrangement Problem in G of value at most C'.*

Proof. The cut function $h : 2^V \to \mathbb{R}^+$ is a submodular function. So for all $S, T \subset V$:

$$h(S) + h(T) \geq h(S \cap T) + h(S \cup T)$$

So consider a solution to the min-cut linear arrangement problem in G' of value C'. And suppose that the ordering for K is $\{u_1, u_2, ...u_k\}$. For each i, find a set $B_i \subset V$ s.t. $h(B_i) = h_K(\{u_1, u_2, ...u_i\})$.

Consider the sets B_1 and B_2. We can find sets B_1', B_2' such that $B_1' \subset B_2'$ and $h(B_1) = h(B_1'), h(B_2) = h(B_2')$ and $B_1' \cap K = B_1 \cap K$, $B_2' \cap K = B_2 \cap K$. This is true via submodularity: Choose $B_1' = B_1 \cap B_2$ and $B_2' = B_1 \cup B_2$.

So $B_1' \cap K = (B_1 \cap K) \cap (B_2 \cap K) = (B_1 \cap K)$ and also $B_2' \cap K = (B_1 \cap K) \cup (B_2 \cap K) = B_2 \cap K$ because $B_1 \cap K \subset B_2 \cap K$.

Also

$$h(B_1) + h(B_2) \geq h(B_1') + h(B_2')$$

via submodularity. However $h(B_1)$ is the minimal value of an edge cut separating $B_1 \cap K$ from $K - (B_1 \cap K)$ and B_1' also separates $B_1 \cap K$ from $K - (B_1 \cap K)$, and a similar statement holds for B_2'. So the above inequality implies

$$h(B_1) = h(B_1'), h(B_2) = h(B_2')$$

We can continue the above argument and get sets $B_1', B_2', ...B_k'$ such that

$$h(B_i') = h_K(\{u_1, u_2, ...u_i\}) \text{ and } B_1' \subset B_2' ... \subset B_k'$$

Then we can choose $A'_i = (B'_i - B'_{i-1}) \cap (V - K)$ as our partition of $V - K$ (and let $D_i = A'_i \cup \{u_i\}$) and then for any $1 \leq i \leq k$:

$$h(\cup_{1 \leq j \leq i} D_j) = h_K(\{u_1, u_2, ...u_i\}) \leq h'(\{u_1, u_2, ...u_i\})$$

So

$$\max_{1 \leq i \leq k} h(\cup_{1 \leq j \leq i} D_j) \leq \max_{1 \leq i \leq k} h'(\{u_1, u_2, ...u_i\}) \leq C'$$

□

Using approximation algorithms due to [29] for min-cut linear arrangement and due to [10] for minimum linear arrangement, we get:

THEOREM 6.3. *There is a polynomial (in n and k) time $O(\log^4 k \log\log k)$-approximation algorithm for the Steiner Minimum Linear Arrangement Problem, and a polynomial time $O(\log^{5.5} k)$-approximation algorithm for the Steiner Min-Cut Linear Arrangement Problem.*

7. Discussion. We note that good quality cut-sparsifiers realized as a convex combination of 0-extension graphs can also be viewed as an *oblivious* algorithm for the 0-extension problem in which the semi-metric D is required to be ℓ_1. Here, by an oblivious algorithm we mean that the algorithm is given knowledge of the graph $G = (V, E)$ and the set of terminals K, but not the semi-metric D on the terminals. The algorithm must nevertheless choose a 0-extension without knowledge of this cost function D. Then using the well-known result that any ℓ_1-metric can be expressed as a non-negative combination of cut-metrics, any C-quality cut-sparsifier that is realized as a convex combination of 0-extension graphs is also a C-competitive oblivious algorithm for the 0-extension problem (in which D is required to be ℓ_1). The results in [28] prove that there are vertex sparsifiers that approximately preserve the congestion of all multicommodity flows, and again these vertex sparsifiers are realized as a convex combination of 0-extension graphs. So one can view the results in [28] as a refinement and extension of those presented here; in particular, the results in [28] imply that there is an $O(\frac{\log k}{\log\log k})$-competitive oblivious algorithm for the 0-extension problem *with no restriction on the semi-metric D.*

Viewing both the cut-sparsifiers considered here and the stronger flow-sparsifiers considered in [28] as oblivious algorithms, one can think of vertex sparsification as another instance in which an oblivious algorithm leads to a structural insight that yields new approximation algorithms. A famous result in this category is due to Räcke [36], who gives an oblivious algorithm for congestion minimization. The particular tree-like structure of this oblivious algorithm yields a number of surprising consequences in approximation algorithms. So, we can draw the conclusion that some oblivious algorithms reveal structural insight into general graphs. In our case, this structural insight is that good vertex sparsifiers exist. In the case of oblivious routing, this structural insight is that one can exchange capacities and distances in probabilistic mappings.

8. Open Questions. The main open question in this paper is:

OPEN QUESTION 1. *Are there $\tilde{O}(\sqrt{\log k})$ quality cut-sparsifiers in general graphs?*

Such a result would be implied by an improved rounding algorithm (for the linear program given in [24] and [9]) for the 0-extension problem when the input semi-metric is ℓ_1. A lower bound on the integrality gap better than $\Omega(\sqrt{\log k})$ would not

immediately imply a negative result to the above question – because the best cut-sparsifiers are not always realizable through contractions. In general, there can be a super-constant gap [11]. Also, the lower bounds given in [28], [11] and [31] only apply to the case in which a "cut-sparsifier" contains only terminals - i.e. $G' = (K, E')$. Of course, almost all of the results presented in this paper remain intact even if a small number (say, $k^{O(1)}$) of non-terminal nodes are allowed. So another intriguing open question is:

OPEN QUESTION 2. *Does allowing $k^{O(1)}$ additional non-terminal nodes in a cut-sparsifier allow better approximation of the terminal cut function?*

In fact, we know of no examples that require more than k additional non-terminal nodes to get an *exact* approximation of the terminal cut function!

Acknowledgments. We would like to thank Tom Leighton, Harald Räcke and Satish Rao for many helpful discussions.

REFERENCES

[1] M. Adler, N. Harvey, K. Jain, R. Kleinberg, and A. R. Lehman, "On the capacity of information networks," *Symposium on Discrete Algorithms*, pp. 251–260, 2006.
[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor, "A general approach to online network optimization problems," *ACM Transactions on Algorithms*, pp. 640–660, 2006.
[3] R. Andersen and U. Feige. "Interchanging distance and capacity in probabilistic mappings," *Arxiv*, 2009.
[4] S. Arora, J. Lee, and A. Naor, "Euclidean distortion and the sparsest cut," *Journal of the AMS*, pp. 1–21, 2008.
[5] S. Arora, S. Rao, and U. Vazirani. "Expander flows, geometric embeddings and graph partitioning,", *Journal of the ACM*, 2009.
[6] Y. Aumann and Y. Rabani, "An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm," *SIAM Journal on Computing*, vol. 27, pp. 291–301, 1998.
[7] Y. Azar, E. Cohen, A. Fiat, K. Haim, and H. Räcke, "Optimal oblivious routing in polynomial time," *Symposium on Theory of Computing*, pp. 383–388, 2003.
[8] Y. Bartal, "Probabilistic approximation of metric spaces and its algorithmic applications," *Foundations of Computer Science*, pp. 184–193, 1996.
[9] G. Calinescu, H. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, pp. 358–372, 2004.
[10] M. Charikar, M. Hajiaghayi, H. Karloff, and S. Rao, "ℓ_2^2 spreading metrics for vertex ordering problems," *Algorithmica*, pp. 577–604, 2010.
[11] M. Charikar, T. Leighton, S. Li, A. Moitra. Vertex sparsifiers and abstract rounding algorithms. In *Foundations of Computer Science*, pp. 265–274, 2010.
[12] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar. Vertex sparsifiers: new results from old techniques. In *APPROX-RANDOM* pp. 152-165, 2010.
[13] J. Fakcharoenphol, C. Harrelson, S. Rao, and K. Talwar, "An improved approximation algorithm for the 0-extension problem," *Symposium on Discrete Algorithms*, pp. 257–265, 2003.
[14] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, pp. 485–497, 2004.
[15] J. Fakcharoenphol and K. Talwar, "An improved decomposition theorem for graphs excluding a fixed minor," *RANDOM-APPROX*, pp. 36–46, 2003.
[16] N. Garg and J. Könemann, "Faster and simpler approximation algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, pp. 680–652, 2007.
[17] N. Garg, V. Vazirani, and M. Yannakakis, "Approximate max-flow min-(multi)cut theorems and their applications," *SIAM Journal on Computing*, vol. 25, pp. 235–251, 1996.
[18] A. Gupta, V. Nagarajan, and R. Ravi, "An improved approximation algorithm for requirement cut," *Operations Research Letters*, pp. 322–325, 2010.
[19] D. Golovin, V. Nagarajan, and M. Singh, "Approximating the k-multicut problem," *Symposium on Discrete Algorithms*, pp. 621–630, 2006.
[20] M. Grötschel, L. Lovász, and A. Schrijver, "Geometric algorithms and combinatorial optimization," *Springer Verlag*, 1993.

[21] C. Harrelson, K. Hildrum, and S. Rao, "A polynomial-time tree decomposition to minimize congestion," *Symposium on Parallel Algorithms and Architectures*, pp. 34–43, 2003.
[22] M. Jerrum and A. Sinclair, "Approximating the permanent," *SIAM Journal on Computing*, pp. 1149-1178, 1989.
[23] M. Jerrum, A. Sinclair, and E. Vigoda, "A polynomial-time approximation algorithm for the permenant of a matrix with non-negative entries," *Symposium on Theory of Computing*, pp. 712–721, 2001.
[24] A. Karzanov, "Minimum 0-extensions of graph metrics," *European Journal of Combinatorics*, pp. 71–101, 1998.
[25] P. Klein, S. Plotkin, and S. Rao, "Excluded minors, network decomposition, and multicommodity flow," *Symposium on Theory of Computing*, pp. 682–690, 1993.
[26] P. Klein, S. Rao, A. Agrawal, and R. Ravi, "An approximate max-flow min-cut relation for multicommodity flow, with applications," *Combinatorica*, pp. 187–202, 1995.
[27] L. C. Lau, "An approximate max-steiner-tree packing min-steiner-cut theorem," *Combinatorica*, pp. 71–90, 2007.
[28] T. Leighton and A. Moitra, "Extensions and limits to vertex sparsification," *Symposium on Theory of Computing*, pp. 47–56, 2010.
[29] T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," *Journal of the ACM*, pp. 787–832, 1999.
[30] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215–245, 1995.
[31] K. Makarychev and Y. Makarychev. Metric extension operators, vertex sparsifiers and Lipschitz extendability. In *Foundations of Computer Science*, pp. 255-264, 2010.
[32] A. Moitra, "Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size," *Foundations of Computer Science*, pp. 3–12, 2009.
[33] V. Nagarajan and R. Ravi, "Approximation algorithms for requirement cut on graphs," *RANDOM-APPROX*, pp. 209–220, 2005.
[34] S. Plotkin and E. Tardos, "Improved bounds on the max-flow min-cut ratio for multicommodity flows," *Combinatorica*, pp. 425–434, 1995.
[35] H. Räcke, "Minimizing congestion in general networks," *Foundations of Computer Science*, pp. 43–52, 2003.
[36] H. Räcke, "Optimal hierarchical decompositions for congestion minimization in networks," *Symposium on Theory of Computing*, pp. 255–264, 2008.
[37] D. Randall, "Mixing," *Foundations of Computer Science*, pp. 4–15, 2003.
[38] D. Shmoys, *Approximation algorithms for cut problems and their application to divide-and-conquer* In *Approximation Algorithms for NP-hard Problems*, PWS, 1997.
[39] L. Valiant and G. Brebner, "Universal schemes for parallel communication," *Symposium on Theory of Computing*, pp. 263–277, 1981.