Vertex Sparsification of Undirected Graphs

Christina Lee (celee@mit.edu)

December 12, 2012

1 Introduction

Graph sparsification was first introduced by Benzcur and Karger [BK96]. Given a graph $G = (V, E)$, they proposed a randomized algorithm that samples a subset of the edges to construct a capacitated graph $G' = (V, E')$ such that $E' \subset E$ with edge capacities $c : E' \to \mathbb{R}_+$. G' is a $(1 \pm \epsilon)$-cut sparsifier if the value of any cut over G' is within a $(1 \pm \epsilon)$ multiplicative factor of the corresponding cut value in G. Therefore, any algorithmic problem that involves computing cut values over G can be approximated by the cut values over G'. In fact the goal is for G' to be sparse such that $|E'| << |E|$, and so that it becomes much more efficient to compute properties of G' with fewer edges. Therefore G' is an edge-sparse approximation for G. This work was followed by work by Spielman Teng [ST11] to construct sparse graph approximations that not only preserve cut values, but in fact the entire spectrum of the Laplacian matrix of the graph.

However, the focus of this paper is different. Instead of finding an edge-sparse approximation of the graph G over the same set of vertices V, the goal will be to find a graph $H = (K, E_H)$ with edge capacities $c_H : E_H \to \mathbb{R}_+$ such that $K \subset V$ and E_H may not be a subset of E. Again we will require the "vertex sparsifier" H to preserve properties such as cuts and flows over $K \subset V$, which will be precisely defined below. This captures a similar idea as edge sparsifiers, in the sense that we can use this graph H to approximate certain properties of G, and when $|K| << |V|$, this could save a lot of computation since we will be computing over a graph with much fewer vertices.

Vertex sparsifiers were first introduced by Moitra [Moi09]. They are applicable to settings when we only care about a certain subset of the vertices $K \subset V$. Examples include multi-commodity flow problems, where we want to route multiple items between different source and sink terminals. Therefore, the set K is often called the set of "terminal" vertices. Moitra presented constructions for vertex sparsifiers H where the approximation quality is only dependent on $|K|$, and not $|V|$. This is especially surprising when $|K| << |V|$. Moitra and Leighton introduced two types of vertex sparsifiers: cut and flow sparsifiers [LM10].

Section 1 of this paper provides the formal definitions, notation, and setup for vertex cut and flow sparsifiers. Section 2 will show that flow sparsfiers are strictly stronger than cut sparsifiers. We focus on the case of cut sparsifiers for the remaining proofs. Section 3 will present the existential arguments for cut sparsifiers with intuition and examples. Section 3 will present a polynomial time construction algorithm for cut sparsifiers. Finally Section 5 will conclude by mentioning connections between these proofs for flow and cut sparsifiers, as well as recent work that has extended these results.

1.1 Cut Sparsifiers

Assume we are given an undirected graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}_+$ and a subset $K \subset V$ of nodes that we are interested in. Throught the paper we always assume $|K| = k$. Let $\delta_G(A) \subset E$ denote the set of edges in G in the cut $(A, V \setminus A)$. The cut function of G is defined as follows:

$$h_G(A) = \sum_{e \in \delta_G(A)} c(e)$$

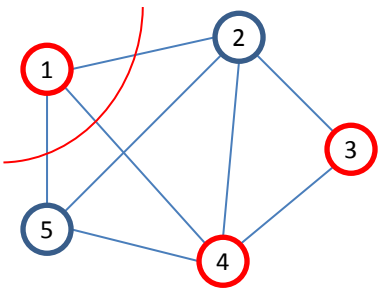
The terminal cut function on K for a subset $U \subset K$ is defined as:

$$h'(U) = \min_{a \in V s.t. A \cap K = U} h_G(A)$$

In words, the terminal cut value of $U \subset K$ is the minimum cut over graph G that separates subsets A and $(K \setminus A)$. An example of this is shown in figure 1, where the terminal nodes are shown in red, and the cut that achieves the minimum value of $h'(\{1\})$ is shown in red. The goal of vertex cut sparsification is to find a graph $H = (K, E_H)$ over the terminal vertices such that the cut function over H upper bounds the terminal cut function.

Figure 1:

$K = \{1, 3, 4\}$
$h'(\{1\}) = 3$
$h'(\{3, 4\}) = 3$



Definition 1.1. *[Moi09]*
H is a vertex cut sparsifier if for all $U \subset K$,

$$h'(U) \leq h_H(U)$$

H is a quality α vertex cut sparsifier if for all $U \subset K$,

$$h'(U) \leq h_H(U) \leq \alpha h'(U)$$

In words, H is an α quality cut sparsifier if the cut value of U over graph H is within an α upper bound of the terminal cut value of U over graph G for all $U \subset K$. Note that the resulting graph H could introduce edges not present in G and result in a dense graph over the K nodes. In contrast to the former notion of edge sparsifiers [BK96], a vertex cut sparsifier must strictly upper bound the cut withing some α factor, rather than approximating the cut within some $(1 \pm \epsilon)$ factor.

1.2 Flow Sparsifiers and Maximum Concurrent Flow

A vertex flow sparsifier is a graph $H = (K, E_H)$ over a subset of the original vertices V, that approximately preserves the congestion of every multicommodity flow with endpoints spported in K. In fact, flow sparsifiers are generalizations of cut sparsifiers, strictly preserving more properties of G. To fully define a vertex flow sparsifier, we first define the maximum concurrent flow problem, which is also a useful concrete application that illustrates the utility of vertex sparsifiers.

Assume we are given 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 $D = \{(s_i, t_i, d_i)\}$. Each i can be thought of as a commodity such that d_i units of i need to be routed from source s_i to sink t_i. Set $K = \cup_i \{s_i, t_i\}$, such that set K contains only nodes that are either source or sink for a commodity. The maximum concurrent flow is the largest λ such that λ fraction of each demand d_i can be simultaneously satisfied while obeying capacity constraints.

We can formulate this with the following linear program:

$$\begin{aligned} \lambda^*(D) = \max \lambda & \\ \text{s.t} \quad \forall i, & \sum_{p \in P_{s_i,t_i}} x(p) \geq \lambda d_i \\ \forall e, & \sum_{p \ni e} x(p) \leq c(e) \\ \forall p, & \quad x(p) \geq 0 \end{aligned}$$

This uses an exponential number of variables $x(p)$, which denotes the amount of flow over a path p. P_{s_i,t_i} denotes the set of paths from source s_i to sink t_i. The first constraint ensures that λ fraction of the demand for each item is satisfied. The second constraint ensures that the flow satisfies edge capacity constraints. The third constraint ensures that the flow is positive. We overload the notation and let the demand D be represented as a vector $\mathbb{R}^{\binom{k}{2}}$ specifying the demand between any pair of terminal nodes in K. Congestion is defined as $cong(D) = \frac{1}{\lambda^*(D)}$.

Definition 1.2. *[LM10]*
$H = (K, E_H)$ is a vertex flow sparsifier if for all demands D,

$$\lambda_G^*(D) \leq \lambda_H^*(D)$$

$H = (K, E_H)$ is a quality α vertex flow sparsifier if for all demands D,

$$\lambda^*_G(D) \leq \lambda^*_H(D) \leq \alpha\lambda^*_G(D)$$

In words, for any demand $D \in \mathbb{R}^{\binom{k}{2}}$ the vertex flow sparsifier H can concurrently satisfy a larger fraction of the demand than graph G, within a factor of α.

1.3 Summary of Highlighted Results

Theorem 1.3. *[LM10] Given an α-quality vertex flow sparsifier $H = (K, E_H)$ for a capacitated graph $G = (V, E)$, H is also an α-quality vertex cut sparsifier for G.*

This theorem proves that flow sparsifiers are strictly stronger and more general than cut sparsifiers. It follows from a simple application of the min-cut max-flow theorem.

Theorem 1.4. *[Moi09] For any capacitated graph $G = (V, E)$, for any set $K \subset V$ such that $|K| = k$, there is an $O(\log k/ \log\log k)$-quality vertex cut sparsifier $H = (K, E_H)$.*

This theorem is proved by setting up a two player game, such that the bounds for the game value imply bounds on the quality of a vertex flow sparsifier. This proof is built on the 0-extension problem.

Theorem 1.5. *[CLLM10] For any capacitated graph $G = (V, E)$, for any set $K \subset V$ such that $|K| = k$, there is an algorithm that computes an $O(\log k/ \log\log k)$-quality cut-sparsifier $H = (K, E_H)$ in time polynomial in n and k.*

Vertex sparsification is written as a linear program with a polynomial number variables and polynomial number of constraints. Each of the constraints are checkable in polynomial time. The existence proof guarantees the linear program is feasible. Thus, any linear programming algorithm such as the ellipsoid algorithm can be used to construct a solution.

2 Flow Sparsifiers vs. Cut Sparsifiers [LM10]

Let's prove Lheorem 1.3 in two parts. Given an α-quality vertex flow sparsifier $H = (K, E_H)$ for a capacitated graph $G = (V, E)$, H is also a vertex cut sparsifier for G if for all $U \subset K$,

$$h'(U) \leq h_H(U) \leq \alpha h'(U)$$

Lemma 2.1. *For all $U \subset K, h'(U) \leq h_H(U)$.*

Suppose that there was some $U \subset K$ such that $h'(U) > h_H(U)$. Since $h'(U)$ is the min-cut in G separating U from $(K \setminus U)$, then by the min-cut max-flow theorem, there is some feasible flow in G from U to $(K \setminus U)$ such that the total flow across the cut is exactly $h'(U)$. But since $h_H(U) < h'(U)$, this flow cannot be feasible in H, since it is larger than the corresponding cut in H. This contradicts the assumption that H is a vertex flow sparsifier for G. Thus the lemma is proven true by contradiction.

Lemma 2.2. *For all $U \subset K, h_H(U) \leq \alpha h'(U)$.*

Choose the flow demand vector D such that the demand d_i from s_i to t_i is equal to the capacity $c_H(s_i, t_i)$ of that edge in graph H. Therefore, by construction, flow D is feasible in H, and $\lambda^*_H(D) = 1$. Suppose that there was some $U \subset K$ such that $h_H(U) > \alpha h'(U)$. There will be $h_H(U)$ units of demand crossing the cut between U and $(K \setminus U)$ in H. Likewise, there will be $h_H(U)$ units of demand crossing any cut between U and $(K \setminus U)$ in graph G. However since the minimum cut $h'(U) < \frac{h_H(U)}{\alpha}$, then a maximum of $\frac{1}{\alpha}$ fraction of the demand can be routed simultaneously in graph G. Thus $\lambda^*_G(D) < \frac{1}{\alpha} < \frac{\lambda^*_H(D)}{\alpha}$. However this contradictions the assumption that H is an α-quality vertex flow sparsifier for G. Thus the lemma is proven true by contradiction.

Theorem 2.3. *Given a flow sparsifier H, the quality of H is equal to the congestion of the hardest flow $\tilde{d}$ over G, where $\tilde{d} \in \mathbb{R}_+^{\binom{k}{2}}$ is the demand vector such that for all $a, b \in K, \tilde{d}_{a,b} = c_H(a, b)$.*

This demand $\tilde{d}$ is feasible in graph H by simply saturating all the edges (see this by construction). In fact, this demand $\tilde{d}$ is the "hardest" demand for G to satisfy. This makes intuitive sense, because all other flows that are routable in H can be satisfied in G with the same flow in G that routes this hardest $\tilde{d}$ demand. Since this is the "hardest" demand, the quality of H as a flow sparsifier is equal to $\frac{1}{\lambda^*_G(\tilde{d})}$, or the congestion of $\tilde{d}$ over G. Therefore, finding the quality of a flow sparsifier H can be formulated as a single minimum congestion routing problem. Following from Theorem 1.3, this can also be used to upper bound the quality of a cut sparsifier H. Thus we have a polynomial time algorithm to determine the quality of a vertex sparsifier. We will use this property in the construction algorithm for a vertex sparsifier.

3 Existence of a "Good" Cut Sparsifier [Moi09]

3.1 0-extension problem

Definition 3.1. *$f : V \to K$ is a 0-extension if for all $a \in K$, $f(a) = a$.*

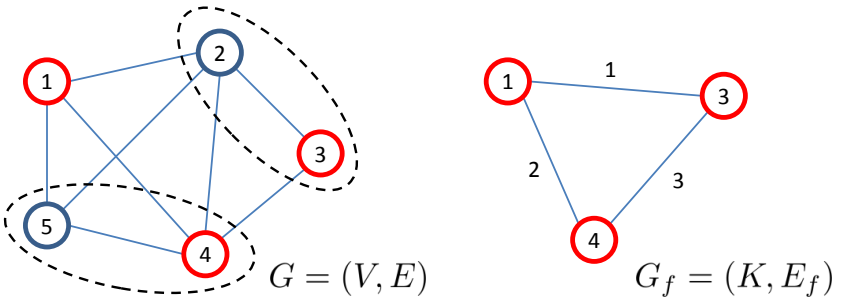
Definition 3.2. *Given a graph $G = (V, E)$, a set $K \subset V$, and a 0-extension $f : V \to K$, let $G_f = (K, E_f)$ denote a capacitated graph with capacities defined as:*

$$c_f(a, b) = \sum_{(u,v) \in E} c(u, v) \mathbb{1}_{\{f(u)=a, f(v)=b\}}$$

In words, a 0-extension can be thought of as a contraction of the vertices V onto the vertices K, and G_f is the graph obtained by combining the edges of vertices that are contracted together. This is illustrated in figure 2.

Figure 2:

$K = \{1, 3, 4\}$
$f(2) = 3$
$f(5) = 4$
$f(a) = a,\ \forall\ a \in K$



Lemma 3.3. *For any 0-extension f, G_f is a vertex cut sparsifier for G. Equivalently, for all $U \subset K$, $h_{G_f}(U) \geq h'(U)$.*

This lemma is obvious because any cut over G_f corresponds to a cut over G where each node $x \in V$ is placed on the same side of the cut as its corresponding terminal node $f(x)$. Therefore, the cut value $h_{G_f}(U)$ is always an upper bound to the true terminal cut value $h'(U)$ since $h'(U)$ minimizes over other cuts of G separating U and $(K \setminus U)$. However, this does not guarantee that it is a good quality cut sparsifier. From figure 2 above, we see that $h_{G_f}(\{3\}) = 4$, while $h'(\{3\}) = 2$, thus indeed $h_{G_f}(\{3\}) \geq h'(\{3\})$. In fact for the graph in figure 2, G_f is a 2-quality vertex cut sparsifier.

In fact, any single 0-extension graph G_f could do really badly for certain cuts. The big idea is that since different 0-extensions do badly for different cuts, we take a convex combination of 0-extension graphs, and show that the resulting graph sparsifier does relatively well for all cuts. This argument is fully flushed out in the following section.

3.2 Zero-Sum Game

Player 1: P1 chooses a 0-extension $f : V \to K$.
Player 2: P2 chooses a cut over K denoted by $U \subset K$
Payoff: P2 wins $\frac{1}{h'(U)}$ for each unit of capacity crossing the cut $(U, K \setminus U)$ in G_f. Thus

$$\text{Payoff}(P1) = -\frac{h_{G_f}(U)}{h'(U)}, \quad \text{Payoff}(P2) = \frac{h_{G_f}(U)}{h'(U)}$$

Comparing the equation for the payoff to definition 1.1, if we find strategies for P1 and P2 such that the game value (payoffs) is bounded, then the bound will correspond to the quality of a cut sparsifier constructed through the players' strategies. P1 wants to choose the 0-extension that corresponds to the best quality sparsifier G_f. Remember that a lower value for quality is better (because it more closely bounds the cuts). P2 wants to choose the cut that the corresponding G_f upper bounds most loosely. Note that the payoff of P2 is always greater than 1, and less than the quality of G_f for a fixed strategy f for P1. Each player only has finite number of possible strategies since $|V|$ and $|K|$ are finite. We use von Neumann's Minimax Theorem for finite strategy space games to bound the game value.

Theorem 3.4. (von Neumann's Minimax Theorem)
For every two-player zero-sum game with finite strategy sets, there exists $Q \in \mathbb{R}$ called the game value such that Q is the best payoff or cost either player can expect to recieve from the game. Furthermore, there exists mixed strategies μ_1, μ_2 for P1 and P2, such that if P1 plays μ_1, P2 cannot expect to do better than payoff Q. Likewise if P2 plays μ_2, P1 cannot expect to do better than cost Q. These mixed strategies forming a Nash equilibrium for the game.

In order to upper bound Q, we can upper bound the cost of P1's best response to any fixed randomized strategy μ_2 for P2. μ_2 is a distribution over cuts $U \subset K$. For each $U \subset K$, let $A_U \subset V$ denote the cut in the full graph G such that $A_U \cap K = U$ and $h_G(A_U) = h'(U)$. Basically, A_U is a cut over G that achieves the minimum cut value for separating $(U, K \setminus U)$. For a given U, there may be multiple choices for A_U, but simple choose any set that achieves the terminal cut value. We can define a function $D : V \times V \to \{0\} \cup \mathbb{R}$, which is similar to the expected cost per unit of capacity over each edge. For any $u, v \in V$,

$$D(u,v) = \sum_{\text{cuts } U \subset K} \mu_2(U) \left(\frac{\mathbb{1}_{\{u,v \text{ in diff partitions of } (A_U, V \setminus A_U)\}}}{h'(U)} \right)$$

D is simply a linear combination of partition functions

$$D_{A_U} = \begin{cases} 1, & \text{if } (u,v) \text{ crosses the partitioning } (A_U, V \setminus A_U) \\ 0, & \text{if } (u,v) \text{ are in the same partition, either } A_U \text{ or } (V \setminus A_U) \end{cases}$$

Definition 3.5. *$q : X \times X \to \{0\} \cup \mathbb{R}$ is a semimetric over X if it satisfies:*

$$\begin{aligned} &q(a,b) = 0 && \text{if } a = b \\ &q(a,b) = \delta(b,a) && \text{for all } a, b \in X \\ &q(a,b) + \delta(b,c) \geq \delta(a,c) && \text{for all } a, b, c \in X \end{aligned}$$

Furthermore, q is a metric over X if in addition $q(a,b) = 0$ only if $a = b$.

It is easy to check that D_{A_U} is a semimetric over V, and D_{A_U} restricted to domain $K \times K$ is a metric over K. Therefore D is also a semimetric over V. Let $D' : K \times K \to \{0\}\mathbb{R}$ denote the function D restricted to domain $K \times K$. We can also verify that D' is a metric over K.

We are now ready to formalize our problem as the 0-extension problem. Suppose we are given a graph $G = (V, E)$ with capacity $c : E \to \mathbb{R}$, a set of terminals $K \subset V$, and a metric D' on K. The goal is to find a 0-extension $f : V \to K$ that minimizes $\sum_{(u,v) \in E} c(u,v) D'(f(u), f(v))$. Fakcharoenphol, Harrelson, Rao, and Talwar give a $O(\log k / \log \log k)$ approximation algorithm for this problem [FHRT03]. They relax the problem to a minimization of $\sum_{(u,v) \in E} c(u,v) D(u,v)$, where D is a semimetric over V such that $D(a,b) = D'(a,b)$ for $a, b \in K$. Thus, using the approximation algorithm in Fakcharoenphol, Harrelson, Rao, and Talwar [FHRT03], there exists a 0-extension f such that

$$\sum_{(u,v) \in E} c(u,v) D'(f(u), f(v)) \leq O(\log k / \log \log k) \sum_{(u,v) \in E} c(u,v) D(u,v)$$

We simplify expressions on both sides of the inequality:

$$\begin{aligned}\sum_{(u,v)\in E} c(u,v)D'(f(u),f(v)) &= \sum_{(u,v)\in E} c(u,v) \sum_{\text{cuts } U\subset K} \mu_2(U)\left(\frac{\mathbb{1}_{\{(f(u),f(v))\in\delta_{G_f}(U)\}}}{h'(U)}\right)\\ &= \sum_{\text{cuts } U\subset K} \frac{\mu_2(U)}{h'(U)} \sum_{(u,v)\in E} c(u,v)\mathbb{1}_{\{(f(u),f(v))\in\delta_{G_f}(U)\}}\\ &= \sum_{\text{cuts } U\subset K} \frac{\mu_2(U)h_{G_f}(U)}{h'(U)}\\ &= \mathbb{E}[\text{Payoff}(P2) \mid P1 \text{ plays } f, P2 \text{ plays } \mu_2]\end{aligned}$$

$$\begin{aligned}\sum_{(u,v)\in E} c(u,v)D(u,v) &= \sum_{(u,v)\in E} c(u,v) \sum_{\text{cuts } U\subset K} \mu_2(U)\left(\frac{\mathbb{1}_{\{u,v\in\delta_G(A_U)\}}}{h'(U)}\right)\\ &= \sum_{\text{cuts } U\subset K} \frac{\mu_2(U)}{h'(U)} \sum_{(u,v)\in E} c(u,v)\mathbb{1}_{\{u,v\in\delta_G(A_U)\}}\\ &= \sum_{\text{cuts } U\subset K} \frac{\mu_2(U)h'(U)}{h'(U)} = 1\end{aligned}$$

Thus, for all μ_2, there exists some f such that

$$\mathbb{E}[\text{Payoff}(P2) \mid P1 \text{ plays } f, P2 \text{ plays } \mu_2] \le O(\log k/\log\log k)$$

Therefore, the game value $Q < O(\log k/\log\log k)$ because for any P2 strategy μ_2, P1 can choose a f such that P2 cannot do better than $O(\log k/\log\log k)$. By von Neumann's Minimax Theorem, there must also exist a mixed strategy μ_1 for P1 such that P2 can choose a cut $U \subset K$ that ensures the expected game value is Q. Then construct the $O(\log k/\log\log k)$-quality cut sparsifier H by taking the convex combination of G_f according to the distribution μ_1. For all $a, b \in K$,

$$c_H(a,b) = \sum_f \mu_1(f)c_f(a,b)$$

Then by von Neumann's Minimax Theorem,

$$\max_{\text{cuts } U\subset K} \mathbb{E}[\text{Payoff}(P2) \mid P1 \text{ plays } \mu_1, P2 \text{ plays } U] = Q \le O(\log k/\log\log k)$$

$$\begin{aligned}\mathbb{E}[\text{Payoff}(P2) \mid P1 \text{ plays } \mu_1, P2 \text{ plays } U] &= \sum_f \mu_1(f)\frac{h_{G_f}(U)}{h'(U)}\\ &= \frac{1}{h'(U)}\sum_f \mu_1(f)h_{G_f}(U)\\ &= \frac{h_H(U)}{h'(U)}\end{aligned}$$

Therefore, for all cuts $U \subset K$, $\frac{h_H(U)}{h'(U)} \leq O(\log k / \log\log k)$, guaranteeing that H is a $O\left(\frac{\log k}{\log\log k}\right)$-quality vertex flow sparsifier. Von Neumann's Minimax Theorem guarantees the existence of H by guaranteeing the existence of μ_1; however, it does not provide a construction algorithm for μ_1. Thus, we can't use it to construct H.

4 Constructing a "Good" Flow Sparsifier [CLLM10]

One construction algorithm simple follows from a linear program representation of the problem introduced in [CLLM10]. Charikar et al give the general algorithm for flow sparsifiers, and here by reducing the algorithm specifically for cut sparsifiers, we are able to provide much clearer intuition for the constraints of the linear program. We want to choose a sparsifier represented by the edge capacities $c_H(a,b)$ for all $a, b \in K$ that minimizes the quality α. Therefore the linear program is:

$$
\begin{aligned}
\min \quad & \alpha \\
\text{s.t} \quad & h'(U) \leq \sum_{a,b \in K} c_H(a,b) \mathbb{1}_{\{a,b \text{ in opp partitions } (U, K\setminus U)\}}, && \forall \ U \subset K \\
& \sum_{(a,b)\in K} c_H(a,b) \mathbb{1}_{\{a,b \text{ in opp partitions } (U, K\setminus U)\}} \leq \alpha h'(U), && \forall \ U \subset K \\
& c_H(a,b) \geq 0, && \forall \ a, b \in K
\end{aligned}
$$

The first type of constraints ensures that H is a cut sparsifier ($h'(U) \leq h_H(U)$). The second type of constraints ensures that H is an α-quality cut sparsifier($h_H(U) \leq \alpha h'(U)$). Note that there an exponential in k number of constraints. We know from the proof of existence that the polytope defined by these constraints is nonempty for some $\alpha = O(\log / \log\log k)$. By Theorem 2.3, we can reduce the second type of constraints to solving the minimum congestion routing problem for the hardest flow $\tilde{d}$. However, the first type of constraints (to ensure H is a cut sparsifier) is difficult to check in polynomial time since $c_H(a,b)$ may not have any structure.

Notice that there may exist a better sparsifier that does not use a convex combination of 0-extensions; however, since we only are looking for a $O(\log / \log\log k)$-quality cut sparsifier, we can restrict ourselves to finding a graph H that is a convex combination of 0-extensions f. This would then enforce that H is a cut sparsifier by Lemma 3.3. However, it would require an exponential number of variables, since there are an exponential number of 0-extensions.

Charikar et. al. suggests relaxing the linear program using a "lifting operation" with "Earth mover constraints" [CLLM10]. However, we observe here, that for the specific case of cut sparsifiers, this reformulation and relaxation of the linear program corresponds to defining variables over the edges and vertices representing the probability that a vertex $u \in V$ is mapped to a vertex $a \in K$, and the probability an edge $(u,v) \in E$ is mapped to an edge

$(a,b) \in E_H$. Given a distribution μ over 0-extensions, for all $a, b \in K$, and $u, v \in V$, let

$$x_{a,b}^{u,v} = \mathbb{P}(f(u) = a, f(v) = (b)) = \sum_f \mu(f) \mathbb{1}_{\{f(u)=a, f(v)=b\}}$$

$$x_a^u = \mathbb{P}(f(u) = a) = \sum_f \mu(f) \mathbb{1}_{\{f(u)=a\}}$$

Therefore, it is clear to see that for all $a, b \in K$ and $u, v \in V$,

$$x_{a,b}^{u,v} = x_{b,a}^{v,u}, \quad \sum_{b \in K} x_{a,b}^{u,v} = x_a^u, \quad x_{a,b}^{u,v} \geq 0, \quad \sum_{a \in K} x_a^u = 1, \quad x_a^a = 1$$

Furthermore,

$$c_H(a,b) = \sum_f \mu(f) c_f(a,b) = \sum_f \mu(f) \sum_{(u,v) \in E} c(u,v) \mathbb{1}_{\{f(u)=a, f(v)=b\}} = \sum_{(u,v) \in E} c(u,v) x_{a,b}^{u,v}$$

Therefore, instead of using an exponential number of variables to describe the distribution over f, we use variables $x_{a,b}^{u,v}$ and x_a^u with the constraints above. Thus, it is a relaxation because it may allow graphs H that are not actually convex combinations of 0-extensions. However, Charikar et. al. showed that those constraints above are enough to guarantee that H is a cut sparsifier, therefore we can simply check each of those polynomially many constraints to verify H is a cut sparsifier. Thus we have:

$$\begin{aligned}
\min \quad & \alpha \\
\text{s.t} \quad & x_{a,b}^{u,v} = x_{b,a}^{v,u}, \quad \sum_{b \in K} x_{a,b}^{u,v} = x_a^u, \quad x_{a,b}^{u,v} \geq 0, && \forall \ \ a, b \in K, \ u, v \in V \\
& \sum_{a \in K} x_a^u = 1, \quad x_a^a = 1, && \forall \ \ a \in K, u \in V \\
& c_H(a,b) = \sum_{(u,v) \in E} c(u,v) x_{a,b}^{u,v}, && \forall \ \ a, b \in K \\
& cong_H(\tilde{d}) \leq \alpha, && \text{for } \tilde{d} \in \mathbb{R}^{\binom{k}{2}} \ s.t. \ \tilde{d}_{a,b} = c_H(a,b)
\end{aligned}$$

Therefore, we have reduced and relaxed the linear program to a program with polynomial number of constraints, where each constraint can be verified in polynomial time. Furthermore, the existential proof in section 3 showed that there exists a feasible solution for $\alpha = O(\log / \log \log k)$. Thus solution to this linear program will give a $O(\log / \log \log k)$-quality cut sparsifier, matching the bound from the existential proof.

5 Conclusion

In this paper, we have presented the vertex sparsification problem. We have provided a clear proof and intuition for the existence of $O(\log k / \log \log k)$-quality cut sparsfiers [Moi09]. Then

we presented the construction algorithm for finding a $O(\log k / \log\log k)$-quality cut sparsfier in polynomial time [CLLM10].

In addition, we have introduced flow sparsifiers and showed why they are strictly generalizations of cut sparsifiers. There has been much work extending each of the existence and construction results for cut sparsifiers to flow sparsifiers [LM10] [CLLM10]. In fact the exact bounds of $O(\log k / \log\log k)$ for both the existence and construction arguments extend from cut to flow sparsifiers. The proofs follow the same structure except instead of using cuts, they use more general metrics over V and K. The Moitra style constructions may return dense sparsifiers H that may not be convex combinations over 0-extensions. Englert et. al. presented a different style construction algorithm that constructs flow sparsifiers with "simpler" structure, specficially sparsifiers H that are a convex combination of trees, obtaining the same approximation quality of $O(\log / \log\log k)$.

In addition to the existence and construction arguments, there are proven lower bounds of $\Omega(\log^{1/4} k)$ for cut sparsifiers [CLLM10] and $\Omega(\log\log k)$ for flow sparsifiers [LM10]. This proves that there may not exist a constant approximation vertex sparsifier for a given graph G and terminal set K.

References

[BK96] András A. Benczúr and David R. Karger. Approximating s-t Minimum Cuts in $\tilde{O}(n^2)$ Time. In *STOC*, pages 47–55, 1996.

[CLLM10] Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. *CoRR*, abs/1006.4536, 2010.

[FHRT03] Jittat Fakcharoenphol, Chris Harrelson, Satish Rao, and Kunal Talwar. An improved approximation algorithm for the 0-extension problem. In *SODA*, pages 257–265, 2003.

[LM10] Frank Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *STOC*, pages 47–56, 2010.

[Moi09] Ankur Moitra. Approximation Algorithms for Multicommodity-Type Problems with Guarantees Independent of the Graph Size. In *FOCS*, pages 3–12, 2009.

[ST11] Daniel A. Spielman and Shang-Hua Teng. Spectral Sparsification of Graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.