Spectral Sparsification in Dynamic Graph Streams

Kook Jin Ahn[1⋆], Sudipto Guha[1⋆], and Andrew McGregor[2⋆⋆]

[1] University of Pennsylvania
{kookjin,sudipto}@seas.upenn.edu
[2] University of Massachusetts Amherst
mcgregor@cs.umass.edu

Abstract. We present a new bound relating edge connectivity in a simple, unweighted graph with effective resistance in the corresponding electrical network. The bound is tight. While we believe the bound is of independent interest, our work is motivated by the problem of constructing combinatorial and spectral sparsifiers of a graph, i.e., sparse, weighted sub-graphs that preserve cut information (in the case of combinatorial sparsifiers) and additional spectral information (in the case of spectral sparsifiers). Recent results by Fung et al. (STOC 2011) and Spielman and Srivastava (SICOMP 2011) show that sampling edges with probability based on edge-connectivity gives rise to a combinatorial sparsifier whereas sampling edges with probability based on effective resistance gives rise to a spectral sparsifier. Our result implies that by simply increasing the sampling probability by a $O(n^{2/3})$ factor in the combinatorial sparsifier construction, we also preserve the spectral properties of the graph. Combining this with the algorithms of Ahn et al. (SODA 2012, PODS 2012) gives rise to the first data stream algorithm for the construction of spectral sparsifiers in the dynamic setting where edges can be added or removed from the stream. This was posed as an open question by Kelner and Levin (STACS 2011).

1 Introduction

The main result of this paper is a bound between two basic graph quantities. First, let λ_e denote the edge-connectivity of edge $e = (s, t)$ in an unweighted graph G, i.e., the size of the minimum s-t cut. Second, consider the electrical network corresponding to G where every edge has unit resistance. Then, let r_e denote the effective resistance of edge $e = (s, t)$, i.e., the potential difference induced between s and t when a unit of current is injected at s and extracted at t. Then we show that $\lambda_e^{-1} \leq r_e \leq O(n^{2/3})\lambda_e^{-1}$.

Furthermore, there exist graphs where this inequality is tight. The first inequality is well known but the best existing upper bound for r_e in terms of λ_e

⋆ Research supported by NSF awards CCF-0644119, CCF-1117216 and a gift from Google.
⋆⋆ Research supported by NSF award CCF-0953754.

is $O(\sqrt{m})\lambda_e^{-1}$ where m is the number of edges in the graph [6]. Hence, our new bound is a strict improvement when $m = \Omega(n^{4/3})$. Indirectly related is work by Lyons et al. [17] that showed that for some classes of graphs, the effective resistance is within a constant factor of the corresponding edge connectivity if the graph is randomly weighted.

Graph Sparsification. The main idea in graph sparsification [4, 20] is to approximate a given graph $G = (V, E)$ by a sparse, weighted subgraph $H = (V, E', w)$. Throughout this paper, we will assume that G is unweighted and simple. A useful application of sparsifiers is in the design of faster algorithms for a range of problems including approximate max-flow [4] and sparsest cut [16]. The idea is that if H is a good approximation of G in an appropriate sense, then it suffices to solve the problem of interest on H rather than on G which would potentially have had many more edges. We say that H is a *combinatorial sparsifier* if

$$(1-\epsilon)\lambda_G(U) \leq \lambda_H(U) \leq (1+\epsilon)\lambda_G(U) \qquad \forall \text{ cuts } (U, V \setminus U) \tag{1}$$

where $\lambda_G(U)$ is the cardinality of the edges in E that cross the cut and $\lambda_H(U)$ is the total weight of the edges in E' that cross the cut. A more powerful sparsifier is a *spectral sparsifier* defined as follows. Let $L_G, L_H \in \mathbb{R}^{n\times n}$ by the Laplacian matrices of G and H respectively.[3] Then we say H is a spectral sparsifier of G if

$$(1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x \qquad \forall x \in \mathbb{R}^n \ . \tag{2}$$

It is not hard to show that condition (2) implies condition (1) by relaxing the condition to only hold for $x \in \{0,1\}^n$.

It is perhaps surprising that, given any graph G, there exist sparsifiers for G with only $O(\epsilon^{-2} n \operatorname{polylog} n)$ edges. It is also possible to construct these subgraphs in near-linear time. In fact, construction is possible via the following two simple and elegant sampling algorithms:

1. *Combinatorial Sparsification* [11]: Sample each edge $e \in E$ independently with probability $p_e = \rho/\lambda_e$ where $\rho = \Theta(\epsilon^{-2}\log^2 n)$. Add each sampled edge e to H with weight $1/p_e$.
2. *Spectral Sparsification* [19]: Sample $\Omega(\epsilon^{-2} n \log n)$ edges with replacement where the probability p_e of picking e is proportional to r_e. Add each sampled edge e to H with weight s_e/p_e where s_e is the number of times e was sampled.

Roughly speaking, sampling e with respect to $1/\lambda_e$ is sufficient for preserving cut sizes whereas sampling e with respect to r_e also preserves additional spectral properties. In a sense, r_e is a more "nuanced" quantity since it takes into account not just the number of edge-disjoint paths between the end points of e but also the lengths of these paths. However, a consequence of our result is that if we simply over-sample by a factor $O(n^{2/3})$ in the above construction of a combinatorial sparsifier, we automatically preserve spectral information.

[3] Recall that $L_G = D_G - A_G$ where A_G is the adjacency matrix of G and D_G is the diagonal matrix where $D_G(i,i)$ is the degree of the ith node. Similarly $L_H = D_H - A_H$ where A_H is the weighted adjacency matrix of H and D_H is the diagonal matrix where $D_H(i,i) = \sum_{j:(i,j)\in E'} w_{ij}$ is the weighted degree of the ith node.

Dynamic Graph Streams and Sketches. The motivation for this work was the design and analysis of *graph sketching algorithms*, i.e., algorithms that use a (random) linear projection to compress a graph in such a way that relevant properties of the graph can be estimated from the projection with high accuracy and confidence. This use of linear projections is well-studied in the context of processing numerical data, e.g., signal reconstruction in compressed sensing [5, 8], Johnson-Lindenstrauss style dimensionality reduction [1, 12], and estimating properties of the frequency vectors that arise in data stream applications [7, 13]. However, it is only recently that it has been shown that this technique can be applied to more structured data such as graphs [2, 3].

Specifically, a sketch of a graph is defined as follows:

Definition 1 (Graph Sketches). *A* linear measurement *of a graph on n nodes is defined by a set of coefficients $\{c_e : e \in [n] \times [n]\}$. Given a graph $G = (V, E)$, the evaluation of this measurement is defined as $M_\lambda(G) = \sum_{e \in E} c_e$. A* sketch *is a collection of (non-adaptive) linear measurements. The size of this collection is referred to as the dimensionality of the sketch*

It was recently shown that there exist $O(\epsilon^{-2} n \operatorname{polylog} n)$-dimensional sketches from which a combinatorial sparsifier can be constructed [2, 3]. This naturally gave rise to the first data stream algorithm for cut estimation when the stream is *fully-dynamic*, i.e., contains both edge insertions and deletions. The space-use of the algorithm is essentially the dimensionality of the sketch, i.e., $O(\epsilon^{-2} n \operatorname{polylog} n)$ and is therefore referred to as a *semi-streaming algorithm* [10]. This follows because each linear measurement can be evaluated on the stream using a single counter: on the insertion of e, we add c_e and on the deletion of e, we subtract c_e. In this paper, we develop the first data stream algorithm for constructing a spectral sparsifier. The algorithm uses $O(\epsilon^{-2} n^{5/3} \operatorname{polylog} n)$ space. In the case where there are no edge deletions, Kelner and Levin [15] designed an algorithm that used $O(\epsilon^{-2} n \operatorname{polylog} n)$ space. They posed the fully-dynamic case as an open problem.

2 Edge Connectivity vs. Effective Resistance

The proof is pleasantly simple and proceeds by considering the execution of the Edmonds-Karp algorithm [9], i.e., the Ford-Fulkerson algorithm that uses the shortest augmenting path first. Let d_e be the length of the $\lceil \lambda_e/2 \rceil$-th augmenting path when executing the algorithm.

We will use the following basic properties of effective resistances:

1. The resistance of a path is the sum of the resistances along the path.
2. The resistance of parallel paths is the harmonic mean of the resistances of the individual paths.
3. The resistance does not increase if edges are added to a graph.

Therefore the main idea would be to construct a suitable subgraph with the desired resistance and the result would follow.

Lemma 1. *The effective resistance on e is at most $2d_e\lambda_e^{-1}$.*

Proof. From the first $t = \lceil \lambda_e/2 \rceil$ augmenting paths, we can construct t edge-disjoint paths say $p_1, p_2, \ldots, p_t$. Note that the augmenting paths are directed and two augmenting paths can use the same original edge in both directions. Moreover the length of these directed flow augmenting paths is nondecreasing. As a consequence $\sum_i p_i \leq td_e$.

We now construct actual flow paths $\ell_1, \ell_2, \ldots, \ell_t$. In this process we eliminate cycles formed by using the edge in both directions. But as a result the length of a particular flow path can increase. However the cycle cancellation ensures that the total length is nonincreasing and $\sum_i \ell_i \leq \sum_i p_i$ which is sufficient for our proof here.

First assume these paths are vertex disjoint. If there were no other edges in the graph, r_e would be the harmonic mean of $\ell_1, \ldots, \ell_t$. However, adding extra edges will only decrease the effective resistance and hence

$$r_e \leq \frac{1}{\frac{1}{\ell_1} + \frac{1}{\ell_2} + \cdots + \frac{1}{\ell_t}} \leq \frac{\sum_i \ell_i}{t^2} \leq \frac{d_e}{t} \leq \frac{2d_e}{\lambda_e} \ .$$

where the second inequality is an application of the HM-AM inequality.

For the general case, we reduce to the vertex-disjoint case by removing all edges not in $\ell_1, \ldots, \ell_t$ to create circuit C_1 where any node v that is used in multiple paths is replaced by multiple copies (such that each path uses a distinct copy). Then, the effective resistance of e in the resulting circuit C_1 is at most the harmonic mean of $\ell_1, \ldots, \ell_t$ as before. Now consider adding 0 ohm resisters between each of the copies of an original node to give a new circuit C_2. Note that effective resistances in C_2 are less than in C_1 because C_2 was formed by adding edges. But then note that effective resistance in C_2 is the same as the effective resistance in the subgraph of G defined by these subset of edges because the voltage (potential) in each of the copies of an original node will be the same. Finally, adding additional edges to this subgraph does not increase the effective resistance. The lemma follows. □

Theorem 1. *In a simple, unweighted graph, the effective resistance on e is at most $O(n\lambda_e^{-3/2})$.*

Proof. Consider the residual graph after $\lceil \lambda_e/2 \rceil$ steps. The connectivity of e in the residual graph is $\lfloor \lambda_e/2 \rfloor$ and the shortest path length is at least d_e. It can then be shown[4] [14] that $d_e = O(n\lambda_e^{-1/2})$. Therefore, by appealing to Lemma 1, the effective resistance of e is at most $2d_e\lambda_e^{-1} \leq O(n\lambda_e^{-3/2})$. □

Corollary 1. *For simple, unweighted graphs, $\lambda_e^{-1} \leq r_e \leq O(n^{2/3})\lambda_e^{-1}$.*

[4] For completeness we include the argument here. Let $e = (s, t)$ and let $\Gamma_i(s)$ be the set of nodes with distance exactly i from s. Because there are still at least $\lfloor \lambda_e/2 \rfloor$ edge disjoint paths between s and t we know $(|\Gamma_i(s)| + |\Gamma_{i+1}(s)|)^2 \geq \lfloor \lambda_e/2 \rfloor$ for each i. Hence, $\Omega(d_e\sqrt{\lambda_e}) = \sum_{i=1}^{d_e} |\Gamma_i(s)| \leq n$ and therefore $d_e = O(n\lambda_e^{-1/2})$ as required.

Proof. First, note that $r_e \leq 1$ and so if $n\lambda_e^{-3/2} \geq 1$, we have

$$r_e \leq 1 \leq (n\lambda_e^{-3/2})^{2/3} = n^{2/3}\lambda_e^{-1} \ ,$$

as required. On the other hand, if $n\lambda_e^{-3/2} \leq 1$, then by Theorem 1 we also have

$$r_e \leq O\left(n\lambda_e^{-3/2}\right) \leq O\left(n^{2/3}\lambda_e^{-1}\right).$$

□

2.1 A Tight Example

We next show that the bound in Corollary 1 is tight. Consider a layered graph with layers $L_0 = \{s\}, L_1, \ldots, L_k, L_{k+1} = \{t\}$ with edges

$$E = \{(u,v) : u \in L_i, v \in L_{i+1} \text{ for some } i \in \{0,1,2,\ldots,k\}\} \cup \{(s,t)\} \ .$$

Let $e = (s,t)$. Let $k = n/\sqrt{\lambda_e}$. Setting $|L_1| = |L_k| = \lambda_e - 1$ and $|L_1| = |L_2| = \ldots = |L_{k-1}| = \sqrt{\lambda_e - 1}$ ensures that the minimum $s-t$ cut has size λ_e. The total number of nodes is $\Theta(n)$ on the assumption that $\lambda_e = O(n)$. The graph is illustrated in Figure 1.

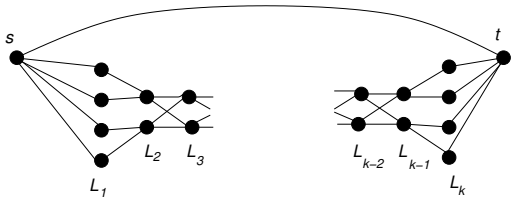

Fig. 1. Tight example for Corollary 1

The above network, when considered from the perspective of s and t is the same as the following network in Figure 2; the nodes in $L_1, \ldots, L_k$ all have the same potential and the capacity (parallel edges) between adjacent nodes is $\lambda_e - 1$. Therefore, using the fact that for parallel edges the effective resistance is the harmonic mean of the resistances, the effective resistance between L_i, L_{i+1} is $1/(\lambda_e - 1)$ when we ignore edge e. It follows that, if we ignore the edge (s,t), then

Fig. 2. Network Equivalent to Figure 1 (from the perspective of s,t)

the effective resistance between s and t is $\Theta(k\lambda_e^{-1}) = \Theta(n\lambda_e^{-1.5})$ since resistances connected in series are additive. If we set $\lambda_e = n^{2/3}$ then $n\lambda_e^{-1.5} = 1$ and consequently $r_e = \Theta(1)$. Therefore, the edge $e = (s,t)$ satisfies $r_e = \Theta(n^{2/3}/\lambda_e)$ and hence Corollary 1 is tight.

3 Spectral Sparsification

Our sparsification result uses the following theorem[5] due to Spielman and Srivastava [19].

Theorem 2. *Given a graph G, let $\{z_e\}_{e\in E}$ be a set of positive values that satisfy:*

1. *$z_e \geq r_e$ for all $e \in E$*
2. *$\sum_e z_e = \beta \sum_e r_e$ for some $\beta \geq 1$*

Sample $q \geq c_0\beta\epsilon^{-2}n\log n$ edges with replacement where edge e is chosen with probability $p_e = z_e/\sum_e z_e$ where c_0 is an absolute constant. Let H be the weighted graph where edge e has weight $s_e/(qp_e)$ where s_e is the number of times e was sampled. Then,

$$(1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x \qquad \forall x \in \mathbb{R}^n \ ,$$

with probability at least $1/2$.

In what follows z_e will take a known value between α/λ_e and $4\alpha/\lambda_e$ where $\alpha = \Theta(n^{2/3})$ is chosen according to Corollary 1 such that $\alpha/\lambda_e \geq r_e$. Since $r_e \geq 1/\lambda_e$, we have

$$\sum_e z_e \leq \sum_e 4\alpha/\lambda_e \leq 4\alpha \sum_e r_e$$

as required. Therefore, to construct a spectral sparsifier of a dynamic graph stream, it suffices to emulate that above sampling procedure. To do this, we divide the procedure into the following two steps:

1. *Independent Edge Sampling:* We first show how to sample edges such that each edge is sampled independently and e is sampled with probability $y_e = \min\{8c_0 z_e \epsilon^{-2}\log n, 1\}$. This will be performed in a single-pass over a dynamic graph stream using $O(\epsilon^{-2}n^{5/3}\operatorname{polylog} n)$ space. Let S be the set of samples returned.
2. *Emulate Sampling with Replacement:* Given the sampled edges S along with $\{z_e\}_{e\in S}$ and $Z = \sum_e z_e$, we show that it is possible to emulate the required sampling with replacement. This will be performed in post-processing without requiring an additional pass over the data stream.

In the next section we show how to perform the independent edge sampling with the required parameters. However, emulating sampling replacement is straight-forward and is described in the proof of the next lemma.

[5] The theorem follows from Corollary 6 in [19] after a re-parameterization of ϵ and by scaling z_e such that we ensure $z_e \geq r_e$.

Lemma 2. *Let $S = \{e_1, e_2, \ldots\}$ be the edges returned by the independent edge sampling and let $\{z_e\}_{e \in E}$ be the sampling parameters with $Z = \sum_{e \in E} z_e$. Then, given a positive integer $q \in [c_0 \beta \epsilon^{-2} n \log n, 2c_0 \beta \epsilon^{-2} n \log n]$ suppose that we sample each edge e with probability at least $y_e = \min\{8c_0 z_e \epsilon^{-2} \log n, 1\}$, then after the independent edge sampling is done, we can emulate the process of sampling q edges with replacement such that each edge is chosen with probability proportional to z_e.*

Proof. Consider a Poisson process with parameter z_e for each edge e and collect the first q edges. This sampling process is equivalent to sample q edges with replacement where the probability of sampling an edge e is z_e/Z. Recall $Z = \beta \sum_e r_e = \beta(n-1)$ since $\sum_e r_e = n - 1$ for any graph [6, 19].

We use the independent sampling to emulate this process with high probability. Then, the expected number of edges sampled in a unit time in the above Poisson process is Z and therefore, we sample q edges in $2q/Z$ time with high probability assuming that q is sufficiently large (note that $q \geq \Omega(n\epsilon^{-2} \log n)$).

Let t_e be the random variable which indicates the first time when e appears in the above Poisson process. If $t_e > 2q/r$, we can safely ignore e because then e is not going to be sampled with high probability. Otherwise, we want to store e to emulate the process. By the definition, $t_e \sim \mathrm{Exp}(z_e)$ and we have

$$\mathbb{P}\left[t_e \leq \frac{2q}{Z}\right] = 1 - \exp\left(-z_e \frac{2q}{Z}\right) \leq \frac{2qz_e}{Z} \leq \frac{4c_0 \beta n \epsilon^{-2} \log n}{\beta(n-1)} z_e \leq 8c_0 z_e \epsilon^{-2} \log n$$

Therefore, if we sample e with the stated probability or higher, we can emulate the Poisson process for e upto time $2q/Z$ which is equivalent to emulate the sampling with replacement with high probability. □

3.1 Independent Edge Sampling

Our sampling makes use of the following two existing algorithms for dynamic graph streams [2, 3]:

1. k-EDGE-CONNECT: Given a dynamic graph stream defining a graph G, this algorithm returns a subgraph $S = k\text{-EDGE-CONNECT}(G)$ such that with high probability

$$\lambda_S(U) \geq \min(k, \lambda_G(U)) \qquad \text{for any cut } (U, V \setminus U)$$

   The algorithm uses $O(kn \operatorname{polylog} n)$ bits of space.
2. CUT-SPARSIFIER: Given a dynamic graph stream defining a graph G, this algorithm returns a weighted subgraph $H = \text{CUT-SPARSIFIER}(G)$ such that with high probability

$$\lambda_G(U) \leq \lambda_H(U) \leq 2\lambda_G(U) \qquad \text{for any cut } (U, V \setminus U)$$

   The algorithm uses $O(n \operatorname{polylog} n)$ bits of space.

The idea behind our sampling algorithm is to subsample the edges of the graph at $O(\log n)$ different rates, $1/2, 1/4, 1/8, \ldots$. At each sampling rate, or level, we maintain a "skeleton" that ensures that we have at least a certain number of edges across each cut. We then return an edge e if it appears in the skeleton at a particular level where the level should be chosen proportional to $1/\lambda_e$. We use CUT-SPARSIFIER as an oracle that can provide estimates for every λ_e value in post-processing. Specifically, the new edge-sampling algorithm operates as follows:

1. During a single pass of the stream:
   - Construct $H = \text{CUT-SPARSIFIER}(G)$.
   - Construct $T_i = k\text{-EDGE-CONNECT}(G_i^z)$ for $i = 0, 1, 2, \ldots, 2\lg n$ where $G_i^z = (V, E_i^z)$ is the graph formed by sampling each edge in G with probability 2^{-i} and k is set to $16 \ln n$.
   - Construct $S_i = k\text{-EDGE-CONNECT}(G_i)$ for $i = 0, 1, 2, \ldots, 2\lg n$ where $G_i = (V, E_i)$ is the graph formed by sampling each edge in G with probability 2^{-i} and k is set to $64 c_0 \alpha \epsilon^{-2} \log n$.
2. Post-Processing:
   - From H, let $\tilde{\lambda}_e$ be an estimate of λ_e and note that $\lambda_e \leq \tilde{\lambda}_e \leq 2\lambda_e$ by the guarantee of the CUT-SPARSIFIER algorithm.
   - Using $F = \{e : e \in T_i \text{ where } i = \max(0, \left\lfloor \lg(\tilde{\lambda}_e / \ln n) \right\rfloor - 1)\}$, estimate $Z = \sum_e z_e$:
     (a) For $e \in F$, compute $f_e = 2^{\min(0, -\lfloor \lg(\tilde{\lambda}_e / \ln n) \rfloor + 1)}$ and $z_e = 2\alpha/\tilde{\lambda}_e$.
     (b) Return $\tilde{Z} = \sum_{e \in F} z_e / f_e$ as an estimation of Z (See Lemma 4).
   - Let $z_e = 2\alpha/\tilde{\lambda}_e$ and $y_e = \min\{8 c_0 z_e \epsilon^{-2} \log n, 1\}$. Note that,

$$z_e \leq \frac{2\alpha}{\lambda_e} \leq 2\alpha r_e \qquad \text{and} \qquad z_e = \frac{2\alpha}{\tilde{\lambda}_e} \geq \frac{2\alpha}{2\lambda_e} = \frac{\alpha}{\lambda_e} \geq r_e \ ,$$

     which satisfies the desiderata of Theorem 2.
   - Return $\{e : e \in S_i \text{ where } i = \lfloor \lg 1/y_e \rfloor\}$ as the required set of samples for Lemma 2. We now have a set of independent samples with replacement which satisfies Theorem 2 and the sparsifier can be constructed as stated therein. This is proved in Lemma 3.

Note that, for the sake of analysis, we assume each edge is included in G_i independently. However, to implement that algorithm in small space we would actually use Nisan's pseudo-random generator [18]. While this is not necessarily the most efficient way to generate the random variables (it adds additional logarithmic terms to the running time and space complexity), this approach leads to a simpler description of the algorithm.

Lemma 3. *For all $e \in E$ and $i = \lfloor \lg 1/y_e \rfloor$, the edge e is sampled in G_i with probability between y_e and $2y_e$. With high probability, $e \in S_i$ iff $e \in G_i$.*

Proof. Clearly $e \in S_i$ implies $e \in G_i$ since S_i is a subgraph of G_i. For the other direction, assume $e = (s,t) \in G_i$ and let $E_e \subseteq E$ be the edges across the minimum s-t cut in G. In particular, $|E_e| = \lambda_e$. But

$$\mathbb{E}\left[|E_e \cap E_i|\right] \leq 32 c_0 \alpha \epsilon^{-2} \log n$$

and by an application of the Chernoff bound, $|E_e \cap E_i| < 64 c_0 \alpha \epsilon^{-2} \log n$ with high probability. Hence $e \in S_i$ for $k = 64 c_0 \alpha \epsilon^{-2} \log n$ by appealing to the guarantees of the k-EDGE-CONNECT algorithm. By an application of the union bound, this is true for all $e \in E$ with high probability as well. □

The next lemma establishes that we also have a good estimate of $Z = \sum_e z_e$.

Lemma 4. *With high probability,* $(1-\epsilon)Z \leq \tilde{Z} \leq (1+\epsilon)Z$.

Proof. Using an argument almost identical to the proof of Lemma 3 we argue that the algorithm (with high probability) samples each edge e in F with probability f_e. Since f_e is the probability that $e \in F$,

$$\mathbb{E}\left[\tilde{Z}\right] = \mathbb{E}\left[\sum_{e \in F} z_e / f_e\right] = \sum_{e \in E} z_e \ .$$

In addition, $z_e / f_e \leq \epsilon^2 Z / \log n$. By an application of the Chernoff bound, $(1-\epsilon)Z \leq \tilde{Z} \leq (1+\epsilon)Z$ with high probability. □

4 Conclusions

While the bound we establish relating λ_e and r_e is tight up to constant factors, the resulting data stream algorithm need not be optimal. Specifically, we conjecture that spectral sparsification is possible in the semi-streaming model where the algorithm may use $O(n \operatorname{polylog} n)$ space.

References

1. D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.
2. K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467, 2012.
3. K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, pages 5–14, 2012.
4. A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time. In *STOC*, pages 47–55, 1996.
5. E. J. Candès, J. K. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

6. P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *STOC*, pages 273–282, 2011.
7. G. Cormode. Sketch techniques for approximate query processing. In G. Cormode, M. Garofalakis, P. Haas, and C. Jermaine, editors, *Synposes for Approximate Query Processing: Samples, Histograms, Wavelets and Sketches*, Foundations and Trends in Databases. NOW publishers, 2011.
8. D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
9. J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. In *Combinatorial Optimization*, pages 31–33, 2001.
10. J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.
11. W. S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *STOC*, pages 71–80, 2011.
12. W. B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert Space. *Contemporary Mathematics, Vol 26*, pages 189–206, May 1984.
13. D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52, 2010.
14. A. Karzanov. Determining a maximal flow in a network by the method of preflows. *Soviet Math. Dokl.*, 15(2), 1974.
15. J. A. Kelner and A. Levin. Spectral sparsification in the semi-streaming setting. In *STACS*, pages 440–451, 2011.
16. R. Khandekar, S. Rao, and U. V. Vazirani. Graph partitioning using single commodity flows. *J. ACM*, 56(4), 2009.
17. R. Lyons, R. Pemantle, and Y. Peres. Resistance bounds for first-passage percolation and maximum flow. *J. Comb. Theory, Ser. A*, 86(1):158–168, 1999.
18. N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
19. D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.
20. D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, 2004.