Constant Arboricity Spectral Sparsifiers

Timothy Chu
Carnegie Mellon University
tzchu@andrew.cmu.edu

August 17, 2018

Abstract

We show that every graph has a spectral sparsifier with a constant arboricity. Moreover, we show that a range of ε sparsifiers have nearly constant arboricity for constant ϵ. This includes the spectral sparsifiers of Spielman and Srivastava, the semi-streaming spectral sparsifiers of Kelner-Levin and Kyng-Pachocki-Peng-Sachdeva, and more. A wide range of algorithms that are otherwise time-consuming, are known to be efficient on graphs of low arboricity. This includes approximate vertex cover, approximate dominating set, cut-queries, and more. Previously, searching for classes of dynamic sparsifiers with low arboricity have allowed researchers to find fast algorithms for approximate undirected bipartite min-cut.

Further, we discuss how the low arboricity of the Spielman Srivastava spectral sparsifier lets us compute any cut in a graph in time proportional to the vertex set size of the smaller side of the cut, with nearly linear pre-processing time. We hope that the efficiency of certain algorithms on graphs of low arboricity will continue to see algorithmic use. At the heart of our result is a simple statement, with a simple proof. It states: for any graph G, the subgraph of edges with leverage score $> \alpha$ has arboricity $< O(1/\alpha)$. This is sufficient to imply the low arboricity of all the sparsifiers mentioned above.

1 Introduction

Graph Sparsifiers. Let G be a graph with vertices V, and edges E, where $|V| = n$ and $|E| = m$. If H is similar to G under some appropriate measure, then H can be used as a proxy for G in computations. Benczur and Karger [BK96] introduced the notion of a graph

[0]An earlier version of this result (including a proof that the Spielman Srivastava sparsifier is uniformly sparse, and that each graph had a constant number of trees) was worked out with Michael B. Cohen from MIT, Jakub Pachocki from OpenAI, and Richard Peng from Georgia Tech in 2014, under the direction of Gary Miller from Carnegie Mellon University. Those results can be found on arxiv, and will be uploaded shortly.

sparsifier (called a cut-sparsifier) where H is a cut-sparsifier of G (where H has the same vertex set) if every cut in H is within a $(1+\varepsilon)$ factor of the corresponding cut in G. Spielman and Teng [ST04] introduced the notion of graph sparsification in the spectral setting. The Laplacian of G is the unique symmetric matrix L_G such that for all vectors $x \in \mathbb{R}^n$, we have $x^T L_G x = \sum_{(u,v)\in E}(x_u - x_v)^2$. H is an ε-spectral-sparsifier of G if:

$$(1-\varepsilon)\boldsymbol{x}^T L_G \boldsymbol{x} \leq \boldsymbol{x}^T L_H \boldsymbol{x} \leq (1+\epsilon)\boldsymbol{x}^T L_G \boldsymbol{x}. \tag{1}$$

Spielman and Teng [ST04] showed that ε-spectral-sparsifiers can be constructed in $O(n \log^{O(1)} n/\varepsilon^2)$ time. They also demonstrated that a spectral sparsifier is a cut-sparsifier, by restricting $\boldsymbol{x} \in \{0,1\}^n$. Spielman and Srivastava later proved that ε-spectral sparsifiers with $O(n \log^2 n/\varepsilon^2)$ edges can be constructed in nearly linear time via leverage score sampling [SS08].

Arboricity. For a graph G, the arboricity $A(G)$ of the graph is the smallest integer k such that there exists forests $T_1, T_2, \ldots T_k$ which are subgraphs of G such that their union is G. This measure is equivalent, up to a factor of 2, to the maximum average degree of any subgraph. Hence, arboricity can be viewed as a measure of uniform sparsity.

Arboricity has been studied in the context of local algorithms [S18], fixed-parameter tractable algorithms [LSS12, S18], minor-closed graph families, and more. A considerable range of algorithms are known to run quickly on low-arboricity graphs, including approximate vertex cover, approximate dominating set, approximate independent set, approximate maximum matching, and more [S18, S12, KW05, LSS12, GG06]. Furthermore, low arboricity graphs imply a vertex with low degree, which allows for fast Schur complementing (which is analogous to vertex sparsification) [ADKKP16].

1.1 Contributions and Previous Work

We show that any graph G admits a subgraph H that is a ε-spectral-sparsifiers with arboricity $O(1/\varepsilon^2)$. We also show that a range of spectral sparsifiers, including the Spielman Srivastava sparsifier [SS08] have nearly constant arboricity. This implies that all algorithms that run quickly on low arboricity graphs run quickly on such sparsifiers.

Our paper's main lemma, with a simple proof, is showing that for *any* graph, the subset of edges with leverage scores higher than α form a subgraph with arboricity at most $O(1/\alpha)$.

Lemma 1.1. *For any graph G, the subgraph H of edges with leverage score $> \alpha$ has arboricity $O(1/\alpha)$.*

This directly implies the following theorem:

Theorem 1.2. *For any graph G whose leverage scores are all $> \alpha$, graph G has arboricity $O(1/\alpha)$.*

This theorem has the following immediately corollaries:

Corollary 1.2.1. *Any ε-spectral-sparsifier constructed using the procedure of Spielman and Srivastava [SS08] has arboricity at most $O(\log n/\varepsilon^2)$.*

Corollary 1.2.2. *There exists an ε-spectral-sparsifier with $O(1/\varepsilon^2)$ arboricity for any graph G.*

Corollary 1.2.3. *Any ε-spectral-sparsifier constructed using the procedure of Kelner and Levin [KL13] (rigorously proven in [P16, KPPS16]) has arboricity at most $O(\log n/\varepsilon^2)$.*

This is as tight a bound as we can hope to get for each class of sparsifiers.

Corollary 1.2.4. *For any graph G, there exists a subgraph H with arboricity $O(\log^{O(1)} n/\varepsilon)$ such that for all pairs of vertices u and v in $V(G)$, the effective resistance between them in graph H is a $(1+\varepsilon)$ approximation of the effective resistance between u and v in G.*

Corollary 1.2.4 reduces the ϵ dependency of the arboricity, compared to the sparsifiers in Corollaries 1.2.1, 1.2.2 1.2.3.

Proofs of our main theorem, and its corollaries, will be given in Section 3 .

One other new result in our paper is that we show cut queries can be calculated in nearly optimal time on low-arboricity graphs. As shown by Andoni, Krauthgamer, and Woodruff in [AKW14], any sketch of a graph that w.h.p. preserves all cuts in an n-vertex graph must be of size $\Omega(n/\epsilon^2)$ bits. Using Spielman-Srivastava sparsifiers, we can achieve the nearly optimal query time $O\left(|S|\frac{\log n}{\epsilon^2}\right)$ when estimating the boundary of $S \subseteq V$, compared to the trivial query time of $O\left(n\frac{\log n}{\epsilon^2}\right)$. This proof hinges on the low arboricity of Spielman Srivastava sparsifiers. This is formalized in Theorem 4.4.

Besides for cut queries, there is a long list of algorithms that run quickly on low arboricity graphs. These which includes approximate maximum matching, approximate dominating set, and approximate vertex cover [S18, S12, KW05, LSS12, GG06]. Our result immediately implies that fast algorithms for computing any of these, exist given any of the spectral sparsifiers specified above. Approximate vertex cover on low-arboricity spectral sparsifiers has been used by Abraham et.al. in [ADKKP16] to compute $(1-\varepsilon)$ undirected maximum bipartite flow. It remains an open question what the output of these algorithms on a spectral sparsifier H of G, imply about the original graph G.

The existence of $O(\log^2 n/\varepsilon^2)$ sparsifiers has been previously implied by Kyng and Song [KS18], which shows that a $O(\log^2 n/\varepsilon^2)$ random spanning trees with edges re-weighted by their effective conductance, form a spectral sparsifier with high probability. It was previously known that this many random spanning trees was a cut-sparsifier with high probability [FH10].

2 Preliminaries

2.1 Electrical Flows and Effective Resistance

Let graph G = (V, E, c) have edge weights c_e, where c_e is the conductance of each edge. Define the resistance r_e on each edge to be $\frac{1}{c_e}$.

Let L_G be the Laplacian of graph G. Let $\boldsymbol{v} \in \mathbb{R}^{|V|}$ be the vector of voltages on the vertices of V.

Let the vector $\boldsymbol{\chi}_v$ denote the vector of excess demand on each vertex. It's well known in the theory of electrical networks that

$$L_G \boldsymbol{v} = \boldsymbol{\chi}_v, \tag{2}$$

or equivalently,

$$\boldsymbol{v} = L_G^+ \boldsymbol{\chi}_v \tag{3}$$

where L_G^+ is the Moore-Penrose pseudo-inverse of L_G.

For edge $e = (i, j)$ with $i, j \in V$, the effective resistance $R_e(G)$ is defined as

$$R_e(G) = \frac{\boldsymbol{\chi}_v^T L_G^+ \boldsymbol{\chi}_v}{2} \tag{4}$$

where

$$\boldsymbol{\chi}_v := \begin{cases} 1 & x = i \\ -1 & x = j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The effective resistance of edge e can be interpreted as the voltage drop across that edge given an flow of 1 unit of current from i to j.

When the underlying choice of graph G is clear, $R_e(G)$ will be shortened to R_e.

Lemma 2.1. *For all graphs H that spectrally sparsify G,*

$$\left(\frac{1}{1+\epsilon}\right) R_e(H) < R_e(G) < (1+\epsilon) R_e(H) \tag{6}$$

Proof. This follows immediately from Equation 1, and substituting

$$R_e(G) = \frac{\boldsymbol{\chi}_v^T L_G^+ \boldsymbol{\chi}_v}{2}$$

and

$$R_e(H) = \frac{\boldsymbol{\chi}_v^T L_H^+ \boldsymbol{\chi}_v}{2}.$$

□

where $R_e(H)$ denotes the effective resistance of edge e in H and $R_e(G)$ denotes the effective resistance of e in G.

2.2 Leverage Scores

Leverage score sampling has been used in a number of settings [SS08, KL13, P16, KPPS16, CGPSSW18]. In the context of graph Laplacians, leverage scores of an edge are interchangeable with $c_e \cdot R_e$, where c_e represents the conductance of the edge and R_e represents the effective resistance of that edge.

2.3 The Spielman-Srivastava Sparsifier

Spielman and Srivastava showed in [SS08] that any graph can be sparsified with high probability using the following routine, for a large enough constant C:

- For each edge, assign it a probability $p_e := \frac{R_e c_e}{(n-1)}$, where R_e is the effective resistance of that edge and c_e is the conductance (the inverse of the actual resistance) of that edge. Create a distribution on edges where each edge occurs with probability equal to p_e.
- Weight each edge to have conductance $\frac{c_e \epsilon^2}{(Cn \log n) p_e}$, and sample $Cn \log n/\epsilon^2$ edges from this distribution.

We call such a scheme a Spielman-Srivastava sparsifying routine. Note that this scheme allows for multiple edges between any two vertices. The probability each edge is sampled is proportional the leverage score, and the leverage score for any edge sampled is at least $O(\varepsilon^2/\log n)$.

Remark 2.1.1. *Sampling by approximate effective resistances (as Spielman and Srivastava did in their original paper [SS08]) will work in place of using exact values for effective resistances. The results in our paper will still go through; an approximation will still ensure that every edge has a relatively large weighting, which is what the result in our paper depends on.*

2.4 The Marcus-Spielman-Srivastava Sparsifier

The following scheme from [S13] produces a sparsifier with non-zero probability, for sufficiently large constants C:

- For each edge, assign it a probability $p_e := \frac{R_e c_e}{(n-1)}$, where R_e is the effective resistance of that edge and c_e is the conductance (inverse of actual resistance) of that edge. Create a distribution on edges where each edge occurs with probability equal to p_e.
- Weight each edge to have conductance $\frac{c_e \epsilon^2}{p_e (Cn)}$, and sample Cn/ϵ^2 edges from this distribution.

We call such a scheme a Marcus-Spielman-Srivastava sparsifying routine. Note that this scheme allows for multiple edges between any two vertices.

Note that the probability this routine returns a sparsifier may be exponentially small, and there is no known efficient algorithm to actually find such a sparsifier, making the [SS08] result more algorithmically relevant.

Note that the leverage score for any edge sampled is at least $O(\varepsilon^2)$.

2.5 The Kelner-Levin Sparsifier

Kelner and Levin proposed a sparsifier in the semi-streaming setting, which is: one pass over a stream of edges is given, and the memory allotted is the output size. How can one construct a spectral sparsifier with $O(n \log n/\varepsilon^2)$ edges?

This sparsifier also ends using a leverage score sampling algorithm. Their algorithm is quite simple: include an edge, and if the edge count gets too high, re-sparsify in such a way that the leverage score of each edge kept is bounded above (assuming that a 2-approximation for Leverage score is used in Step 3(b) of the algorithm StreamSparsify in [KPPS16].

This scheme was first introduced in [KL13], and a rigorous proof was given in [P16, KPPS16]. Of note is that, if a resparsification step is performed at the end, every edge in the final graph has leverage score at least $O(\varepsilon^2/\log n)$. Call the output of such a procedure a Kelner-Levin sparsifier.

2.6 Resistance Sparsifiers

An ε-resistance-sparsifier H supported on the same vertex set of graph G, is a graph where effective resistance between two vertices in H is within a $(1+\varepsilon)$ multiplicative factor of the corresponding two vertices in G. This notion was first introduced in [DKW15].

The paper of Chu et.al. [CGPSSW18] showed that there exist resistance sparsifiers of size $O(n \log^{O(1)} n/\varepsilon)$. This is done via a tool called short-cycle decompositions (decomposing a graph into short cycles and a small number of residual edges).

In that paper, Chu et.al. gives a short cycle decomposition algorithm called NaiveCycleDecomposition. This algorithm breaks the graph into a collection of short cycles, and a nearly linear number of edges. If NaiveCycleDecomposition is plugged into the algorithm SpectralSketch in the same paper, then the resulting graph can be written as the union of a graph with a nearly linear number of edges, and a graph whose edges have leverage score (with respect to the entire graph) of $\geq \Omega(\varepsilon/log^{O(1)}n)$. Call such a sparsifier a Chu-Gao-Peng-Sachdeva-Sawlani-Wang sparsifier.

2.7 Uniform Sparsity and Low Arboricity

Definition 2.2. *The **arboricity** of a graph G is the equal to the minimum number of forests its edges can be decomposed into.*

Definition 2.3. *A graph $G = (V, E, c)$ is said to be c-**uniformly sparse** if, for all subsets $V' \subset V$, the subgraph induced on G by V' contains no more than $c \cdot |V'|$ edges.*

Lemma 2.4. *Uniform Sparsity implies Low Arboricity. That is, if G is c-uniformly sparse, then the arboricity of G is no greater than $2c$.*

This statement is known in the literature, and is not difficult to show.

3 Low Arboricity Spectral Sparsifiers

In this section, we prove Theorem 1.2 and its associated corollaries. First we establish some preliminary lemmas. Throughout, we assume graph G is connected, and $G = (V, E, c)$ where $c_e \forall e \in E$ represents the edge weights (conductances) of graph G.

Lemma 3.1. *(Foster's Resistance Theorem) Let $G = (V, E, c)$ be any graph. Then*

$$\sum_{e \in E} R_e c_e = n - 1. \tag{7}$$

where $n := |V|$. [F61]

This is equivalent to saying: the sum of leverage scores in any graph is exactly $n - 1$, which is the rank of L_G.

Lemma 3.2. *(Subgraph Effective Resistances are Higher than Graph Effective Resistances) Let H be a subgraph of $G = (V, E, c)$, where L_H is treated as a linear operator from $\mathbb{R}^{|V|}$ to $\mathbb{R}^{|V|}$. Then*

$$x^T L_H^+ x \geq x^T L_G^+ x \tag{8}$$

for all $x \in \mathbb{R}^{|V|}$ orthogonal to the nullspace of L_H.

Lemma 3.2 is known in the literature. Now we are ready to prove Lemma 1.1, which states that the subgraph H of graph G has arboricity $O(1/\alpha)$, where H consists of the edges with leverage score in G greater than α.

Proof. (of Lemma 1.1) We show that H is uniformly sparse. By Lemma 2.4, this would prove the arboricity is $O(1/\alpha)$. Let S be any vertex subset of $V(G)$. Then: we know that the sum of $c_e R_e(G)$ on S is at most $|S| - 1$, by Foster's Theorem and Lemma 3.2.

Therefore, the set of edges with $c_e R_e(G) > \alpha$ must have at most $O(|S|/\alpha)$ edges between two vertices in S, by Markov's inequality. This holds for all subsets S, and thus implies uniform sparsity of $O(1/\alpha)$ □

Theorem 1.2 follows directly from Lemma 1.1.

Theorem 3.3. *Marcus-Spielman-Srivastava sparsifiers are $O(1/\epsilon^2)$-uniformly sparse.*

Proof. For each edge included in the graph by the Marcus-Spielman-Srivastava sampling scheme, they're designed with the property that each edge is sampled with probability proportional to their leverage score, and each edge in the final graph has leverage score at least $O(\varepsilon^2)$ [S13, MSS13].

Therefore, by Theorem 1.2, they have a arboricity of at most $O(1/\varepsilon^2)$. □

This proves Corollary 1.2.2. Now we prove that Spielman Srivastava sparsifiers are uniformly sparse.

Theorem 3.4. *Spielman-Srivastava sparsifiers have arboricity* $O\left(\frac{\log n}{\epsilon^2}\right)$.

Proof. Spielman Srivastava sparsifiers are designed such that the leverage score of any edge in the final graph is at most $O(\varepsilon^2/\log n)$ [SS08]. This implies that they have arboricity at most $O(1/\varepsilon^2)$. □

This proves Corollary 1.2.1.

Theorem 3.5. *Kelner-Levin spectral sparsifiers, in the semi-streaming setting, have arboricity* $O(\log n/\varepsilon^2)$.

Proof. This follows on Theorem 1.2 and the bounds on the leverage score of edges in the Kelner-Levin sparsifier established in Section 2.5. □

Theorem 3.6. *Chu-Gao-Peng-Sachdeva-Sawlani-Wang resistance sparsifiers can be written as the union of a graph with nearly linear number of edges, and a graph with arboricity bounded by* $O(\log^{O(1)} n/\varepsilon)$.

Proof. This follows from Theorem 1.2, and the remarks in Chu-Gao-Peng-Sachdeva-Sawlani-Wang sparsifier established in Section 2.6. □

4 Applications to Approximating Cut Queries

Definition 4.1. *We say that a total ordering* $\dot{<}$ *of the vertices of a graph is* c-treelike *if every vertex* u *has at most* c *neighbors* v *such that* $u \dot{<} v$.

Lemma 4.2. *Every graph with arboricity* c *has a* $2c$*-treelike ordering. Moreover, this ordering can be computed in nearly linear time.*

Proof. Let G be a graph with arboricity c. Let v be the minimum degree vertex of G. Note that $d(v) \leq 2c$. We set v to be the smallest in the ordering $\dot{<}$ and then recursively construct the remainder of the ordering on $G' = G \setminus \{v\}$. The ordering can be computed in $O(n \log n + m)$ time by using a Fibonacci heap priority queue, and updating degrees as we remove the minimum degree vertex v. □

Lemma 4.3. *Let* $G = (V, E)$ *have arboricity* c. *After preprocessing in linear time and space, we can answer queries about the boundaries of subsets of vertices of* G *in* $O(ck)$ *time, where* k *is the size of the queried subset.*

Proof. Using Lemma 4.2 we first compute a $2c$-treelike ordering $\dot{<}$ of V. For every vertex $u \in V$, we store a list of edges $(u, v) \in E$ such that $u \dot{<} v$. We also compute and store the weighted degree $wd(v)$ for every vertex of G.

Assume we are given $S \subseteq V$. We first compute the total weight $s_{internal}$ of edges internal to S. To this end, for every vertex $u \in S$ we go through its neighbors that are larger in the

ordering $\dot{<}$ and sum up the weights of edges that lead to S. Note that every edge in $S \times S$ will be encountered exactly once. The boundary of S can be computed as

$$s_{cut} := \left(\sum_{v \in S} wd(v) \right) - 2s_{internal}.$$

□

Theorem 4.4. *Given a graph with n vertices, there exists a data structure that:*

- *achieves the construction time, storage space, and cut approximation guarantees of Spielman-Srivastava sparsifiers, and*
- *can compute approximate weights of cuts in $O(k\frac{\log n}{\epsilon^2})$ time, where k is the size of the smaller side of the cut.*

This is a simple consequence of Lemma 4.3, and our previous proof that Spielman Srivastava sparsifiers have $O(\log n/\varepsilon^2)$ arboricity.

5 Conclusions

There exist a plethora of algorithms that run quickly on low arboricity graphs. Leveraging low arboricity and the fast run time of these algorithms has been done by [ADKKP16], to approximate maximum bipartite undirected flow quickly.

It is an open question how these fast algorithms on low arboricity graphs can be used to speed up a variety of other classical algorithms relevant in the field, possibly including many algorithms that rely on Schur complementing (such as Effective Resistance estimation, Matrix determinant calculation, computing Sparse Cholesky factorization, and more).

It also remains an open question whether other classes of sparsifiers, such as those by Spielman and Teng [ST04], those by Batson, Spielman, and Srivastava [BSS09], dynamic sparsifiers, and others are necessarily low arboricity graphs. Lee et. al. recently showed how to find Batson-Spielman-Srivastava sparsifiers in nearly linear time. Ideas based on Spielman and Teng spectral sparsifiers have been used to create fast max-flow algorithms . It would be interesting to determine whether or not these sparsifiers have low arboricity, and if so, what kinds of fast algorithms can be run on them.

Thanks to Michael Cohen, Jakub Pachocki, Richard Peng, and Gary Miller for helpful conversations. As mentioned earlier, an earlier version of this work from 2014 with Michael Cohen, Jakub Pachocki, and Richard Peng (which was never published until August 2018) contained many of the main results, and was recently posted to ArXiv. This paper represents a modernized version of the same result, which was never submitted for publication outside of ArXiv, plus a few extra lemmas.

References

[ADKKP16] Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, Richard Peng On Fully Dynamic Graph Sparsifier. *57th Annual IEEE Symposium on Foundations of Computer Science 2016.*

[AKW14] Alexandr Andoni, Robert Krauthgamer, David Woodruff. The Sketching Complexity of Graph Cuts. *CoRR*, abs/1403.7058, 2014.

[BK96] Andras A. Benczur and David Karger. Approximating $s-t$ minimum cuts in $O(n^2)$ time. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, STOC '96.*. pages 47-55. New York, NY, USA, 1996. ACM.

[BSS09] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan Sparsifiers.
SIAM Review, Vol. 56*, Issue* 2*, pp* 315 − 334.2014.

[CGPSSW18] Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, Junxing Wang Graph Sparsification, Spectral Sketches, and Faster Resistance Computation, via Short Cycle Decompositions.
Foundations of Comupter Science, 2018. CoRR, abs/1805.12051, 2018.

[F61] R.M. Foster. An Extension of a Network Theorem. IRE Trans. Circuit Theory, 8 (1961), pp.75-76.

[DKW15] Michael Dinitz, Robert Krauthgamer, Tal Wagner Towards Resistance Sparsifiers. *Approximaion, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015),* volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 738-755. Dagstuhl, Germany, 2015. Scholss-Dagstuhl-Leibniz-Zentrum fuer Informatik. 3

[FH10] Wai Shing Fung, Nicholas J. A. Harvey Graph Sparsification by Edge-Connectivity and Random Spanning Trees. *CoRR*, abs/1005.0265, 2010.

[KL13] Jonathan A. Kelner and Alex Levin. Spectral Sparsification in the semi-streaming setting. *Theory of Computing Systems,* 53(2):243-262, 2013.

[GG06] Gaurav Goel and Jens Gustedt. Bounded Arboricity to Determine the Local Structure of Sparse Graphs Lecture Notes in Computer Science book series (LNCS, volume 4271), 2006.

[KS18] Rasmus Kyng, Zhao Song A Matrix Chernoff Bound for Strongly Rayleigh Distributions and Spectral Sparsifiers from a few Random Spanning Trees.
Foundations of Computer Science, 2018.

[KPPS16] Rasmus Kyng, Jakub Pachocki, Richard Peng, Sushant Sachdeva A Framework for Analyzing Resparsification Algorithms *Symposium on Discrete Algorithms, 2017.*

[KW05] Fabio Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Computing,* 17(4):303-310, 2005.

[LSS12] Min Chih Lin, Francisco J. Soulignac, Jayme L. Szwarcfiter. Arboricity, h-index, and dynamic algorithms. *Theoretical Computer Science,* v407 n.1-3, p.458-473, November 2008.

[MSS13] Adam Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing Families II: Mixed Characteristic Polynomials and the Kadison-Singer Problem. To appear in Annals of Mathematics.

[P16] Jakub Pachocki Analysis of Resparsification *CoRR*, abs/1605.08194, 2016.

[S12] Jukka Suomela. Survey of Local Algorithms. https://users.ics.aalto.fi/suomela/doc/local-survey.pdf

[S13] Nikhil Srivastava. Discrepancy, Graphs, and the Kadison-Singer Problem. http://windowsontheory.org/2013/07/11/discrepancy-graphs-and-the-kadison-singer-conjecture-2/

[S18] Shay Solomon. Local Algorithms for Bounded Degree Sparsifiers in Sparse Graphs. *Proceedings of 9th ITCS*. 52:1–52:19, 2018.

[SS08] Daniel A. Spielman and Nikhil Srivastava. Graph Sparsfiication by Effective Resistances. Conference version appeared in STOC '08.

[ST04] Daniel A. Spielman and Shan-Hua Teng. Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems *Proceedings of the* 36^{th} *Annual ACM Symposium on Theory of Computing, pp.* $81 - 90.2004$