Ultrasparse Ultrasparsifiers and Faster Laplacian System Solvers

ARUN JAMBULAPATI, University of Washington, USA
AARON SIDFORD, Stanford University, USA

In this paper we provide an $O(m\text{loglog}^{O(1)} n \log(1/\epsilon))$-expected time algorithm for solving Laplacian systems on n-node m-edge graphs, improving upon the previous best expected runtime of $O(m\sqrt{\log n}\text{loglog}^{O(1)} n \log(1/\epsilon))$ achieved by (Cohen, Kyng, Miller, Pachocki, Peng, Rao, Xu 2014). To obtain this result we provide efficient constructions of low spectral stretch graph approximations with improved stretch and sparsity bounds. As motivation for this work, we show that for every set of vectors in $\mathbb{R}^d$ (not just those induced by graphs) and all integer $k > 1$ there exist an ultra-sparsifier with $d - 1 + O(d/k)$ re-weighted vectors of relative condition number at most k^2. For small k, this improves upon the previous best known multiplicative factor of $k \cdot \tilde{O}(\log d)$, which is only known for the graph case. Additionally, in the graph case we employ our low-stretch subgraph construction to obtain $n - 1 + O(n/k)$-edge ultrasparsifiers of relative condition number $k^{1+o(1)}$ for $k = \omega(\log^{\delta} n)$ for any $\delta > 0$: this improves upon the previous work for $k = o(\exp(\log^{1/2-\delta} n))$.

CCS Concepts: • **Theory of computation → Sparsification and spanners**; **Streaming, sublinear and near linear time algorithms**.

Additional Key Words and Phrases: symmetric diagonally dominant (SDD) matrices, low-stretch trees

ACM Reference Format:
Arun Jambulapati and Aaron Sidford. 2023. Ultrasparse Ultrasparsifiers and Faster Laplacian System Solvers. 1, 1 (January 2023), 55 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

From the first proof of a nearly linear time Laplacian system solver [ST], to the current state-of-the-art running time for Laplacian system solving [CKM+], to advances in almost linear time approximate maximum flow [Sheb, KLOS, Pen] ultrasparsifiers have played a key role in the design of efficient algorithms. In [ST] ultrasparsifiers were used in the computation of sequences of graph preconditioners that enabled nearly linear time Laplacian system solvers. This initiated a long line of work on faster [KMP, KMP14, CKM+], simpler [LS, KOSA, KS], and more parallel [BGK+, PS, KLP+b] Laplacian system solvers, many of which leverage ultrasparsifiers or related sparse

Authors' addresses: Arun Jambulapati, University of Washington, CSE, Paul G. Allen Building, 185 E Stevens Way NE, Seattle, Washington, 98195, USA, jmblpati@uw.edu; Aaron Sidford, Stanford University, MS&E, Huang Engineering Center, 475 Via Ortega, Stanford, California, 94305, USA, sidford@stanford.edu.

XXXX-XXXX/2023/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

graph approximation, e.g. low stretch spanning trees. These results in turn fueled advances of graph decompositions for a range of problems including approximate maximum flow [Sheb, KLOS, Pen], directed Laplacian solving [CKP+a, CKP+b], and transshipment [Li].

The current fastest Laplacian solver [CKM+] computes expected ϵ-approximate solutions to Laplacians on n-node, m-edge graphs in time[1] $\tilde{O}(m\sqrt{\log n}\log(1/\epsilon))$. These solvers start from the graphs associated with Laplacians and compute randomized tree-based approximations with bounded expected ℓ_p-stretch: they use these to construct a sequence of preconditioners that efficiently decrease the error in expectation. The ultimate runtime achieved by this approach, $\tilde{O}(m\sqrt{\log n}\log(1/\epsilon))$, matches that of the runtime one would achieve if the best known ultrasparsfiers for graphs, due to [KMST], could be constructed in linear time and then preconditioning approaches related to [KMP] were applied. Though it is known that preconditioners exist that would enable an $\tilde{O}(m)\log(1/\epsilon)$ time solver, the current best construction of such preconditioners takes $O(m\text{poly}(\log(n)))$ time due to the need to compute linear sized sparsifiers [BSS14] of Schur complements.

Consequently, the best known bounds of ultrasparsifiers for graphs due to [KMST] constitute a fundamental barrier towards designing faster Laplacian system solvers. [KMST] showed that arbitrary n-vertex graphs possess $\tilde{O}(k\log n)$-spectral approximations with $n+\frac{n}{k}$ edges, and their proof is based on the existence of *low-stretch spanning trees* with average stretch $\tilde{O}(\log n)$. In turn [CKM+] achieves their running times by leveraging that ℓ_p^p stretch variants of these trees can be computed in $\tilde{O}(m)$ time. These methods all pay this $\log n$ factor due to the best known bounds combinatorial techniques for ball-growing and graph decomposition. It is known that the stretch bound of $\tilde{O}(\log n)$ is optimal up to an iterated logarithmic factor in a wide variety of graphs, such as the n-vertex grid or hypercube. In other words, if using just distance based combinatorial graph decomposition, one must pay a factor of $\Theta(\log n)$ in the worst case. This factor then appears in the best known ultrasparsifier bounds and as a $\sqrt{\log n}$ factor in the current best Laplacian solver runtimes, due to the nature of iterative methods for solving Laplacian systems.

The main conceptual contribution of this paper is that this barrier can be broken and this $\Theta(\log n)$ factor can, perhaps surprisingly, be avoided. As a quick, broad proof-of-concept, in Appendix A we show that [KMST] is not optimal in all parameter regimes. For arbitrary matrices and small target distortion we show that there exist sparser ultrasparsifiers that do not pay this $\Theta(\log n)$ factor. Interestingly, we give a simple [BSS14] based argument that arbitrary matrices have low-stretch subgraphs and then we apply the arguments of [KMST] to get our bounds.

Inspired by this proof of concept, the main technical contribution of this paper is to show that for the specific goal of constructing low-distortion spectral subgraphs, i.e. those that would suffice for Laplacian system solving, better bounds can be achieved and the $\Theta(\log n)$-factor can be avoided. We carefully combine both spectral methods and combinatorial decomposition techniques for this purpose. In particular, we show that traditional ball growing techniques can be augmented or patched by careful use of spectral sparsifiers to achieve lower distortion graph approximations.

[1]Here and throughout this paper, the notation $\tilde{O}(\cdot)$ hides loglog factors.

Interestingly, our procedure for augmenting a low-diameter decomposition requires us to efficiently compute stronger notions of spanners in sufficiently dense graphs. We provide an efficient procedure for computing a type of graph approximation related to fault-tolerant spanners, which we call *path spanners*. In turn, to compute path sparsifiers we show that there are many short vertex disjoint paths in a dense near-regular expander. This proofs builds upon the seminal work of [KR] which showed that every dense expander has many short edge-disjoint paths. We leverage this fact in an algorithm which combines a near-linear time expander decomposition procedure of [SW] with a new routine for approximately finding regular dense subgraphs. The resulting path sparsification algorithm serves as a type of vertex-based sparsification in our low-distortion subgraph computation algorithm. It is an interesting open problem if an alternative sparsification procedure can be used instead, however we think the tools developed for obtaining path sparsifiers may be of intrinsic interest.

Ultimately, we show that careful application of this routine for constructing low distortion spectral subgraphs yields an $\tilde{O}(m \log(1/\epsilon))$-time algorithm for computing expected ϵ-approximate solutions to Laplacian systems. This solver leverages heavily recursive preconditioning machinery of [CKP+a] and our efficiently computable spectral subgraphs. To simplify and clarify the derivation and analysis of this recursive solver we provide an analysis of a stochastic preconditioned variant of accelerated gradient descent (AGD) [Nes83].

We hope that this work may serve as the basis for further improvements in ultrasparsification and graph decomposition. Given the myriad of applications of these techniques and the simplicity and generality of our approach for overcoming the $\Theta(\log n)$-factor in previous combinatorial approaches we hope this work may find further applications.

Paper Organization. In the remainder of this introduction we provide preliminaries and notation we use throughout the paper (Section 1.1), our main results (Section 1.2), our approach for achieving them (Section 1.3), and provide a brief discussion of previous work (Section 1.4). In Section 2 we then give our main graph decomposition and use it to obtain low-distortion spectral subgraphs. The results of this section hinge on the efficient construction of a new combinatorial object known as *path sparsifiers*, which we compute efficiently in Section 3. In Section 4 we leverage our low-distortion spectral subgraph construction to obtain our Laplacian system solver results. Our existence proof for ultrasparsifiers is given briefly in Appendix A, our AGD analysis is given in Appendix C, and additional proofs are given in the other appendix sections.

1.1 Preliminaries

Here we provide notation and basic mathematical facts we use throughout the paper.

Graphs: Throughout this paper we let $G = (V, E, w)$ denote an undirected graph on vertices V, with edges $E \subseteq V \times V$, with integer positive edge weights $w \in \mathbb{Z}^E_{>0}$. Though many graphs in this paper are undirected, we typically use $(a, b) \in E$ notation to refer to an edge where we suppose without loss of generality that a canonical orientation of each edge has been chosen. Often in this paper we consider unweighted graphs $G = (V, E)$ where implicitly $w := \vec{1}$. Unless stated otherwise (e.g. much of

Section 1.3.2) we make no assumption about whether graphs are simple in this paper and often consider graphs with multi-edges and self-loops.

Degrees and Weights: For graph $G = (V, E, w)$ and $a \in V$ we let $\deg_G(a) \coloneqq \sum_{e \in E | a \in e} w_e$. Further, we let $d_{\min}(G) \coloneqq \min_{a \in V} \deg_G(a)$, $d_{\max}(G) \coloneqq \max_{a \in V} \deg_G(a)$, $d_{\text{ratio}}(G) \coloneqq d_{\max}(G)/d_{\min}(G)$ and $d_{\text{avg}}(G) \coloneqq \sum_{a \in V} \deg_G(a)/|V|$. For weighted graph $G = (V, E, w)$ we let $w_{\min}(G) \coloneqq \min_{e \in E} w_e$, $w_{\max}(G) \coloneqq \max_{e \in E}$, and $w_{\text{ratio}}(G) \coloneqq w_{\max}(G)/w_{\min}(G)$.

Volumes: For graph $G = (V, E, w)$ and $S \subseteq V$ we let $\text{Vol}_G(S) \coloneqq \sum_{a \in S} \deg_G(a)$ where $w \in \mathbb{R}^E$ are the edge weights of the graph. Note that when we allow self-loops, if a vertex has self-loops of total weight w this contributes w to the degree of that vertex and the volume of any set it is in. We further define the boundary volume $\text{Vol}_G(\partial S) \coloneqq \sum_{(u,v) \in E(G), u \in S, v \notin S} w_{(u,v)}$.

Distances and Shortest Path Balls: For graph $G = (V, E, w)$ and path $P \subseteq E$ between vertices a and b we let $\ell(P) \coloneqq \sum_{e \in P} w_e$ denote the length of the path and we let $d_G(a, b)$ denote the length of the shortest path between a and b. Further, we let $B_G(v, r) \coloneqq \{a \in V \mid d_G(v, a) \leq r\}$ denote the (shortest-path) ball of distance r from v.

Neighbors: For graph G and vertex set $S \subseteq V$ we let $N(S) \coloneqq \{a \in V \mid (b, a) \in E \text{ for some } b \in S\}$ denote the *neighbors of* S. Overloading notation we let $N(a) \coloneqq N(\{a\})$ for all $a \in V$.

Subgraphs and Contractions: For graph $G = (V, E, w)$ and $S \subseteq V$ we let $G[S]$ denote the subgraph induced by S, i.e. the graph with vertices S, edges $E \cap (S \times S)$, and edge weights the same as in G. We overload notation and, similarly, for $F \subseteq E$ we let $G[F]$ denote the subgraph of G induced by edge set F, i.e. the graph with vertices V, edges F, and edge weights the same as in G. For $S \subseteq V$, we let $G \backslash S$ denote the graph by contracting all vertices in S into a single supernode, while preserving multi-edges and possibly inducing self-loops.

Graph Matrices: For weighted graph $G = (V, E, w)$ we let $\mathcal{L}_G \in \mathbb{R}^{V \times V}$ denote its Laplacian matrix where for all $a, b \in V$ we have $\mathcal{L}_{a,b} = -w_{\{a,b\}}$ if $\{a, b\} \in E$ and $\mathcal{L}_{a,a} = \deg_G(a)$.

Effective Resistances: For graph $G = (V, E, w)$ and nodes u, v, we say the effective resistance between u and v is $\mathcal{R}_G^{eff}(u, v) = (\mathbf{e}_u - \mathbf{e}_v)^\top \mathcal{L}_G^\dagger (\mathbf{e}_u - \mathbf{e}_v)$ where $\mathcal{L}_G^\dagger$ denotes the Moore-Penrose pseudoinverse of $\mathcal{L}_G$. Throughout the paper, we make use of several standard facts about effective resistances stated in Appendix D.

Solver: We use the following notation for linear system solvers:

Definition 1.1 (Approximate Solver)**.** *We call a randomized procedure an* ϵ-solver for PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$ *for* $\epsilon \in [0, 1)$ *if given arbitrary* $b \in \mathbb{R}^n$ *it outputs random* $x \in \mathbb{R}^n$ *with*

$$\mathbb{E} \left\| x - \boldsymbol{A}^\dagger b \right\|_{\boldsymbol{A}}^2 \leq \epsilon \|b\|_{\boldsymbol{A}^\dagger}^2 \tag{1}$$

Definition 1.2 (Laplacian Solver)**.** *We call a randomized procedure an* ϵ-Laplacian (system) solver *for graph* $G = (V, E, w)$ *if it is an* ϵ*-solver for* $\mathcal{L}_G$*, i.e. given arbitrary* $b \in \mathbb{R}^n$ *it outputs random* $x \in \mathbb{R}^n$ *with* $\mathbb{E}[\|x - \mathcal{L}_G^\dagger\|_{\mathcal{L}_G}^2] \leq \epsilon \|\mathcal{L}_G^\dagger b\|_{\mathcal{L}_G}^2$.

Error guarantees in the $\mathbf{A}$ and $\mathcal{L}_G$ norm are standard to the literature; they corresponds to an ϵ-multiplicative decrease in the function error on the objective $f(x) = (1/2)x^\top \mathbf{A} x - b^\top x$ from initial point $\vec{0}$. However, that our solver error is defined with respect to the expected square norm of the matrices is less standard. By concavity of

$\sqrt{\cdot}$ for appropriate choice of ϵ this guarantee is stronger than defining error in terms of just the norm: for all PSD $\mathbf{A} \in \mathbb{R}^{n\times n}$ and vectors $x \in \mathbb{R}^n$ we have $[\mathbb{E}\,\|x\|_{\mathbf{A}}]^2 \leq \mathbb{E}\,\|x\|_{\mathbf{A}}^2$.

Asymptotics and Runtimes: Throughout we use $\tilde{O}(\cdot)$ to hide poly(log log) factors in n, the number of vertices in the largest graph considered.

Misc. All logarithms in this paper are in base e unless a base is explicitly specified.

1.2 Our Results

Here we present the main results of our paper. First, as discussed in the introduction we provide new bounds on existence of ultrasparsifiers for arbitrary matrices. Our construction is based on the spectral-sparsification results of [BSS14]. We prove this existence result briefly in Appendix A:

Theorem 1.3 (Ultrasparsifier Existence). *Let $v_1, \ldots v_m \in \mathbb{R}^n$ and $\boldsymbol{A} \coloneqq \sum_{i\in[m]} v_i v_i^\top$. For any integer $k \geq 2$, there exists $S \subseteq [m]$ with $|S| = n + O\left(\frac{n}{\sqrt{k}}\right)$ and $w \in \mathbb{R}^m_{\geq 0}$ where*

$$\boldsymbol{A} \preceq \sum_{i\in S} w_i v_i v_i^\top \preceq k\boldsymbol{A}.$$

This result when specialized to graphs immediately yields $n+O\left(\frac{n}{\sqrt{k}}\right)$-edge subgraphs with relative condition number k, and is a proof of concept towards the main results of this paper. We obtain it by a two-stage construction: we first find an ultrasparse subset of vectors satisfying a certain "on average" notion of spectral approximation, and then we correct this to a true ultrasparsifier with a procedure based on the spectral sparsification algorithm of [BSS14]. In the case of graphs, we give an improved guarantee for the first phase of our construction. We call the objects we compute low distortion spectral subgraphs, defined as follows.

Definition 1.4 (κ-Distortion Spectral Subgraph)**.** *Given a weighted graph $G = (V, E, w)$ we call $H = (V, E_H, w_H)$ a* κ-Distortion Spectral Subgraph *if $\mathcal{L}_H \preceq \mathcal{L}_G$ and*

$$\sum_{e\in E}(w_e \mathcal{R}_H^{\text{eff}}(e)) = \sum_{e\in E}\left(w_e \delta_e^\top \mathcal{L}_H^\dagger \delta_e\right) \leq \kappa\ .$$

If H is a subgraph of G in addition to the above, we call it a κ-distortion subgraph.

To give context for this definition, observe that $H = G$ is an n-distortion spectral subgraph of G, and any strict subgraph of G has spectral distortion strictly larger than n. In fact, we show something slightly stronger than what this definition encompasses. We show that there exist subgraphs with this guarantee that can be computed efficiently. Our main theorem for this construction (specialized for its application for Laplacian solvers) is as follows.

Theorem 1.5 (Efficient Construction of Ultrasparse κ-Distortion Subgraphs). *Let $G = (V, E, w)$ be a polynomially-bounded weighted graph, and let $c \geq 1$ be any fixed constant. Algorithm 3 equipped with Theorem 1.9 runs in $O(m)$ time and returns a κ-distortion subgraph H with $n + O\left(\frac{m}{(\log\log n)^c}\right)$ edges, for*

$$\kappa = O\left(m\,(\log\log n)^{\sqrt{8c}+1+o(1)}\right)\ .$$

It also returns a vector $\tau \in \mathbb{R}^E_{\geq 0}$ with $\|\tau\|_1 \leq \kappa$ where for any $e \in G$, $\tau_e \geq w_e \delta_e^\top \mathcal{L}_H^\dagger \delta_e$ is an overestimate of the leverage score of e measured through H.

We use this result to obtain preconditioners: employing a framework based on [CKM+], we thus obtain our main result on solving Laplacian linear systems.

Theorem 1.6 ($\tilde{O}(m)$-Laplacian System Solver). *There is a randomized algorithm which is an ϵ-approximate Laplacian system solver for any input n-vertex m-edge graph with polynomially-bounded edge weights (see Definition 1.2) and $\epsilon \in (0,1)$ and has the following runtime for any $\chi > 0$*

$$O(m(\log\log n)^{6+2\sqrt{10}+\chi}\log(1/\epsilon)) .$$

We assume polynomially-bounded weights primarily for simplicity of presentation: it can be removed via standard techniques (see for instance [CKP+a] for details) Further, in Appendix A show that techniques for constructing κ-distortion subgraphs yield ultrasparsifiers with the following improved guarantees over Theorem 1.3.

Theorem 1.7 (Improved Ultrasparsifiers). *There exists a polynomial time algorithm which given an input graph G with polynomially-bounded edge weights can compute a reweighted subgraph H with either of the following guarantees:*

- *For any constant c, H has $n + \frac{n}{(\log\log n)^c}$ edges and satisfies*

$$\mathcal{L}_G \preceq \mathcal{L}_H \preceq O((\log\log n)^{c+\sqrt{8c}+1+o(1)})\mathcal{L}_G.$$

- *For any constant $\delta > 0$ and $\alpha = \omega(\log^\delta n)$, H has $n + \frac{n}{\alpha}$ edges and satisfies*

$$\mathcal{L}_G \preceq \mathcal{L}_H \preceq \alpha^{1+o(1)}\mathcal{L}_G.$$

When compared to the previous state-of-the-art ultrasparsifier algorithm [KMST], our construction provides improved spectral approximation qualities for sparsities up to $n + \frac{n}{\alpha}$ for $\alpha = O\left(\exp\left(\log^{1/2-\delta} n\right)\right)$ for any $\delta > 0$, and improves upon Theorem A.2 for $\alpha = \omega\left(\text{poly}(\log\log n)\right)$. In particular, our method improves upon the previous-best methods in the important regime of $\alpha = \text{poly}(\log n)$ by a factor of $O(\log^{1-o(1)} n)$. Ultrasparsifiers of this quality form a critical part of the current best-known approximate max flow algorithms ([Sheb, Pen, Shea]), and thus we believe our techniques may be used to improve the running times of these methods.

To compute κ-distortion subgraphs efficiently we introduce a new combinatorial object we call a (α, β)-*Path Sparsifier*, defined as follows.

Definition 1.8 ((α,β)-Path Sparsifiers)**.** *Given unweighted graph $G = (V, E)$ and $F \subseteq E$ we call subgraph $H = G[F]$ an (α,β)-*path sparsifer *if for all edges $(u, v) \in E$, either $(u, v) \in F$ or there are α vertex-disjoint paths of length at most β from u to v in H, where we do not count u, v as part of the paths.*

Path sparsifiers provide a type of approximation for distance in unweighted graphs that is even stronger than that of fault-tolerant spanners [DK, BP] and crucial for obtaining our linear system solving runtimes. We prove the following theorem regarding path sparsifiers.

Theorem 1.9 (Efficient Path Sparsification). *Given any n-node, m-edge graph and parameter $k \geq 1$ the procedure,* PathSparsify(G, k) *(Algorithm 9) outputs w.h.p. $F \subseteq E$ with $|F| = O(nk \log^3(n))$ such that $G[F]$ is a $(k, O(\log^5 n))$-path-sparsifier of G in expected time $O(m + nk \log^{13}(n))$.*

When used to prove Theorem 1.6, we avoid the (large) polylogarithmic dependence of Theorem 1.9 by applying it to graphs with $O(m)$ edges and $O(\frac{m}{\text{poly} \log n})$ vertices. Thus the cost associated with computing path sparsifiers is $O(m)$: it does not affect our final $\tilde{O}(m)$ runtime claim for Laplacian solvers.

1.3 Overview of Approach

Here we provide a brief overview of our approach towards obtaining the results of Section 1.2.

1.3.1 Ultrasparse Low Distortion Subgraphs. Our techniques for computing ultrasparse low-distortion subgraphs are based on existing algorithms for computing low-stretch spanning trees: in particular we base our construction on a simple recursive procedure from [AKPW95] (with analysis insights from [CKM$^+$]). Given a graph G, our algorithm begins by partitioning its vertices into $V_1, V_2 \ldots$ such that the partition cuts few edges and each $G[V_i]$ has low diameter. With this, it then computes an ultrasparse subgraph inside each $G[V_i]$ such that each edge inside a $G[V_i]$ receives a small effective resistance overestimate when measured though the subgraph. It then recurses on a graph formed by appropriately contracting parts of each $G[V_i]$ and deleting all edges which lie inside a V_i. In the case where the graph computed inside each $G[V_i]$ is a shortest-path tree, the algorithm described is exactly the low-stretch spanning tree procedure of [AKPW95]. Our algorithm extends this classic result by adding a small number of extra edges within each $G[V_i]$ to improve the effective resistance overestimate: we use fast algorithms for path sparsifiers developed in Section 3 for this. Combining this with a more careful graph decomposition gives us our result.

1.3.2 Path Sparsifiers. Here we briefly outline our approach for proving Theorem 1.9, i.e. efficiently computing path sparsifiers. Recall that subgraph H is an (α, β)-path sparsifier of $G = (V, E)$ if every edge in G is either present in H or connected by α vertex-disjoint paths of length at most β. Consequently, constructing a path-sparsifier essentially involves replacing dense components of a graph with sparse subgraphs containing many short vertex disjoint paths.

One natural starting point for construct sparse subgraphs with vertex disjoint paths is to consider expanders, i.e. (informally) graphs where the number of edges leaving every small enough vertex subset is some bounded fraction of the number of edges contained inside the subset. It is known that by seminal work of [KR] that in every sufficiently dense expander every pair of vertices is connected by many short *edge-disjoint paths*. Consequently, our first step in constructing path sparsifiers is to show that in fact this result generalizes to graph that are good *vertex expanders*: graphs where the number of nodes neighboring every small enough vertex subset is some bounded fraction of the number of nodes contained inside the subset. By a standard connection between vertex-expansion and edge-expansion, this immediately yields

that every dense enough graph with nearly uniform degrees has many short paths between every pair of vertices.

Given this primitive, our task of computing a path sparsifiers reduces to the problem of decomposing an arbitrary dense graph into subsets where we can find sparser expanders of nearly uniform degree. To achieve this we provide a procedure for decomposing an arbitrary dense graph into nearly degree uniform dense subgraphs, sample these subgraphs uniformly, and apply known expander decompositions to the result. We show that these expanders with high probability contain enough of the volume of the original graph that by repeating on the edges not contained in these expanders we ultimately obtain a path sparsifier. Further, by careful sampling and use of known nearly linear time expander decompositions, i.e. [SW], our algorithm is time efficient as well and yields the desired Theorem 1.9.

1.3.3 Laplacian System Solvers. Finally, we leverage the contributions of Sections 2 and 3 to obtain our improved Laplacian solving algorithms. Our approach is a modification of the "preconditioning in expectation" framework used in [CKM+] to obtain $\tilde{O}(m\sqrt{\log n}\log(1/\epsilon))$ time Laplacian solvers. Given a graph G we first compute a low-distortion subgraph H, and then aim to solve linear systems in $G' = G + (\eta - 1)H$ for some appropriately chosen η. We use a modified version of accelerated gradient descent [Nes83] (given in Appendix C for completeness) to show that solving linear systems in $\mathcal{L}_G$ can be reduced to solving $O(\sqrt{\eta})$ linear systems in $\mathcal{L}_{G'}$. To solve linear systems in G', we form a series of preconditioners G'_i by sampling each edge e in G' with probability proportional to $w_e r_H(e)$, where $r_H(e)$ is the effective resistance overestimate of e given by the copy of ηH contained in G'. While asking for these G'_i to be true sparsifiers of G' would require paying a logarithmic oversampling factor (and hence appear as an $O(\sqrt{\log n})$ in the runtime guarantee), an insight of [CKM+] shows that sampling without this logarithmic factor still suffices to ensure that solving linear systems in a few randomly sampled $\mathcal{L}_{G'_i}$ enables one to solve linear systems in G'. Finally, via some parameter tradeoffs we can ensure that the G'_i consist of a tree plus a small number of edges: we then apply a combinatorial contraction procedure to eliminate the vertices and edges of the tree and recursively apply our solver to the remaining graphs.

We remark that our analysis of our recursion more closely resembles the original analysis of [KMP] which yields slower runtimes, and not the more sophisticated one given in [CKM+]. Although this tighter analysis was necessary in the previous work to reduce the logarithmic dependence, applying the same techniques here would only reduce our algorithm's poly$(\log\log n)$ dependence. Further, doing this introduces several technical issues which complicate the presentation of our algorithm. For the sake of clarity, we give the analysis which loses poly(loglog$n$) factors in this paper and make only limited attempt to control the polynomial dependence on loglog$n$ throughout the paper.

1.4 Previous Work

Sparsifiers: Spectral graph sparsification has been heavily studied since its invention by [ST] in the process of constructing the first near-linear time Laplacian solver. While the original procedure was somewhat involved, a dramatic simplification by [SS] shows

that sampling edges with probability proportional to their statistical leverage score gives a $(1+\epsilon)$-approximate sparsifier with $O(n \log n \epsilon^{-2})$ edges with high probability. [BSS14] showed that there is a more computationally expensive procedure that obtains sparsifiers with the same approximation quality and without the $\log n$ dependence in size. [KMP] extend this idea by showing that sampling edges with probability proportional to leverage score overestimates induced by a sparse subgraph also gives sparsifiers with high probability. They use this insight to construct *ultrasparsifiers*: given an input graph G they find H with $n + \tilde{O}(\frac{n \log^2 n}{k})$ edges where $\mathcal{L}_H \preceq \mathcal{L}_G \preceq k\mathcal{L}_H$. Extending this result, [KMST] improves the sparsity to $n + \tilde{O}(\frac{n \log n}{k})$ by replacing the random sampling of [KMP] with a procedure based on a sparsity-optimal algorithm by [BSS14]. However, progress on removing this final $\log n$ factor has stalled, partially due to a intrinsic barrier posed by the use of low-stretch spanning trees as a primitive. We bypass this barrier in two ways: we give a purely spectral argument which improves on the sparsity of [KMST] in certain parameter regimes, and we give a procedure which constructs ultrasparse subgraphs with better leverage score overestimates than trees can provide. While our results stop just short of obtaining truly comparable ultrasparsifiers (due to our graph decomposition approach), we provide the first methods to bypass the low-stretch tree barrier present in the previous work.

Graph Decomposition: While our specific notion of a κ-distortion spectral subgraph has (to our knowlege) not been studied before specifically, many related spectral primitives have been considered in the literature. We remark that standard sparsification routines [BSS14] trivially give $O(n)$-distortion spectral subgraphs with $O(n)$ edges, and that a low-stretch spanning tree of a sparsifier gives a tree which is an $\tilde{O}(n \log n)$-distortion spectral subgraph: these are the facts which inspire our definition. The previous fastest Laplacian solver [CKM$^+$] modifies this latter guarantee by providing a tree which is a spectral subgraph satisfying a certain ℓ_p notion of distortion. They provide a construction of a tree where the sum of the p^{th} powers of $w_e \mathcal{R}_H^{\text{eff}}(e)$ is bounded by $O\left(\frac{1}{(1-p)^2} n \log^p n\right)$, for any $0 < p < 1$.[2] Although this guarantee does not recover the standard low-stretch tree guarantee (which yields a bound for $p = 1$), [CKM$^+$] gives an algorithm which computes this tree in $\tilde{O}(m)$ time: this has not yet been achieved by more standard low-stretch spanning tree algorithms. Our work extends this result by giving $\tilde{O}(m)$ time algorithms which trade off the sparsity of the output subgraph with κ.

Laplacian System Solvers: Our result on Laplacian system solvers draws on a long series of work on time-optimal Laplacian system solvers. Our specific approach draws heavily from ultrasparsifier-based algorithms as pioneered by [ST] and refined by [KMP, KMP14, CKM$^+$]. These papers solve the Laplacian linear system $\mathcal{L}x = b$ by first recursively solving linear systems in $\mathcal{L}'x = b$, where $\mathcal{L}'$ is spectrally close to $\mathcal{L}$ but sparse. They then use this ability to solve linear systems in $\mathcal{L}'$ to precondition a conjugate gradient-type method. Although the specific way of constructing the

[2]We remark that [CKM$^+$] does not phrase their guarantee in this way. The tree they compute has steiner vertices, and their graph actually has $\kappa = O(\frac{1}{(1-p)^2} m \log^p n)$. These issues can be eliminated by standard vertex elimination techniques in trees and initially sparsifying the input, respectively. Further, the dependence on m in their claim has no impact on their final Laplacian solver algorithm.

ultrasparse $\mathcal{L}'$ has changed significantly over the previous line of work, all base their construction on *low-stretch trees* (or dominating trees) trees which "on average" contain a short path across the endpoints of a randomly chosen edge from their base graph. Our departure from this line of work is to base our ultrasparsifiers on graphs which are merely ultrasparse: we instead start with a subgraph consisting of a tree with $o(m)$ edges. While the presence of this small number of extra edges may seem inconsequential, we demonstrate that this small overhead allows us to bypass a $O(m\sqrt{\log n})$ barrier present in all of the previous work following the ultrasparsifier archetype.

As we remarked earlier, there is an alternative approach for solving Laplacian systems based on sparsification alone, e.g. [PS, KLP+b, KS], which is known to yield preconditioners that (once computed) yield $\tilde{O}(m)$ Laplacian system solvers. However, constructing such preconditioners currently requires $\Omega(m \log^c n)$ time, where $c \geq 1$ derives from the need to compute $\tilde{O}(n)$-edge sparsifiers. In our solver, we too need efficient strong sparsification-like results, however we show that it is possible to use the path sparsifiers we provide for this purpose.

Another approach proposed by [KLPa] gives Laplacian solvers running in $O(m + n \log^{O(1)} n)$ time: this is $O(m)$ for any slightly dense graph. Their approach is based on computing coarse $O(\log^{O(1)} n)$-quality $n + o(m)$-edge sparsifiers in $O(m)$ time. They then compute effective resistance overestimates in their computed sparsifier in $O(m + n \log^{O(1)} n)$ time by leveraging low-stretch spanning tree algorithms, and they finally leverage these estimates to obtain $o(m)$-edge $O(1)$-sparsifiers to their original graph. By finally using existing near-linear time Laplacian solvers to solve in this approximation to the input graph, they are able to use a standard preconditioning approach to obtain their claimed runtime. While their approach does not yield linear-time algorithms for graphs with $m < n \log^{O(1)} n$, we find it an interesting question to see if their techniques can be combined with ours to obtain $O(m + n(\log \log n)^{O(1)})$-time Laplacian solvers.

Fault Tolerant Spanners. To computing our κ-distortion subgraphs, we construct efficient algorithms for (α, β)-path sparsifiers. These are related to multipath spanners [GGV] and vertex fault-tolerant spanners [DK, BP] studied by the combinatorial graph algorithm community. A k-fault tolerant spanner of input G is a subgraph H such that for any "fault set" F with $|F| \leq k$, $H - F$ is a spanner of $G - F$. Intuitively, such subgraphs must contain many short disjoint paths across the endpoints of any edge not retained from the parent graph: if there was a small set of "bottleneck" nodes present in any short path between u and v for $(u, v) \in E(G)$, deleting these would mean $H - F$ no longer spanned $G - F$. Our definition extends this notion by additionally requiring these short paths to be vertex-disjoint. Our algorithm also departs from the previous work by using the spectral notion of expander decomposition to construct path sparsifiers, in a similar spirit to the independently developed ideas in [BvdBG+20] and in contrast to the random sampling approach of [DK] and the greedy approach of [BP]. While the use of expander decomposition comes with some significant drawbacks (most notably algorithm complexity), it enables us to obtain a linear dependence on the number of edges in our algorithm's runtime.

2 LOW DISTORTION SPECTRAL SUBGRAPHS

In this section we prove Theorem 1.5 showing that we can efficiently compute κ-distortion subgraphs in $O(m)$ time. First, we provide a single-level graph decomposition result in Section 2.1. Then, leveraging our efficient path-sparsification procedure of Section 3 we recursively apply our decomposition to provide our efficient construction of κ-distortion subgraphs in Section 2.2. Later, in Section 4 we use these in a solver framework related to that of [CKM+] to obtain our claimed $O(m(\log\log n)^{O(1)}\log(1/\epsilon))$ time Laplacian solver.

2.1 Graph Decomposition

In this section we give our core combinatorial graph-decomposition technique, which we use to compute low distortion subgraphs. This single-level graph decomposition can be interpreted as a significant modification of low-diameter decomposition as originally conceived by [Awe85]. Broadly, our algorithm chooses an arbitrary vertex and grows a shortest-path ball out from it. Whenever the cut defined by the ball is sufficiently small relative to its volume, we cut the edges defined by the cut, mark the vertices of the ball as a partition piece, and repeat on the remaining vertices in the graph. This basic procedure, known as *low-diameter decomposition*, has seen many applications in graph algorithms [CKM+, MPX, LSY, Bar]. We state the guarantee here:

Theorem 2.1. *Let $G = (V, E)$ be an unweighted graph and let $\beta > 0$ be a parameter. There is an algorithm which runs in $O(m)$ time and computes a partition of the vertices into $V_1, V_2 \ldots$ such that*

- *Each $G[V_i]$ has diameter $O(\beta \log n)$*
- *At most $\frac{m}{\beta}$ edges G cross different partition pieces.*

Unfortunately a significant limitation of the above procedure is the $O(\log n)$ factor in the diameter of the partition pieces. This is neccessary: for example a constant-degree unweighted expander graph has diameter $O(\log n)$ but any partition of G into balls of diameter $o(\log n)$ must necessarily cut at least a constant fraction of its edges.

We avoid this logarithmic factor by settling for a weaker guarantee that still suffices for Theorem 1.5. Our modification is this: after we have grown a ball $B_G(v, r)$ and made a cut, we "retract" the ball by distance δ and consider $B_G(v, r - \delta)$. The key insight is that although the vertices of $B_G(v, r)$ formed a low-conductance cut in G, the cut defined by $B_G(v, r')$ was sufficiently high conductance for any $r' < r$. With this we upper bound the size of $B_G(v, r - \delta)$, and consequently ensure that most vertices in $B_G(v, r)$ are close to a small number of nodes in $B_G(v, r - \delta)$.

Unfortunately, the presence of weights in a graph somewhat complicates this picture, due to the inherently unweighted nature of our expansion-based low-diameter decomposition algorithm. We circumvent this complication with a technique borrowed from [AKPW95]: we bucket the weights of the edges into a few classes $E_1, E_2, \ldots$, and decide to make a cut when $B_G(v, r)$ forms a low-conductance cut in the graph restricted to each E_i. Formally, we prove the following lemma.

Lemma 2.2. *Let $G = (V, E)$ be an unweighted m-edge (multi)graph, let $E_1, E_2, \ldots E_\ell$ be a partition of the edges, Let $r \geq 0$, and let $\beta \in [0, 1/6]$. Algorithm 1 computes in $O(m)$*

Algorithm 1: $\{V_i, U_i^j\}_{i,j\geq 1} = \text{Decompose}(G, \{E_1, E_2, ...\}, \beta, r)$

Input: Graph $G = (V, E)$, partition of E into $E_1, E_2, \cdots E_\ell$, and parameters $\beta, r \geq 0$

Output: $\{V_1, ...V_\alpha\}$ partition of V, $U_i^1, U_i^2, ...U_i$ partition of V_i

1 $t \leftarrow 1$
2 **while** $G \neq \emptyset$ **do**
3 $\quad v \leftarrow$ arbitrary vertex in G
4 $\quad R \leftarrow 0$
5 $\quad V_t \leftarrow B_G(v, R)$
6 $\quad$ **while** $e^{-\frac{r\beta}{\ell}}\text{Vol}_G(\partial V_t) + \text{Vol}_{G[E_j]}(\partial V_t) \geq 3\beta\left(e^{-\frac{r\beta}{\ell}}\text{Vol}_G(V_t) + \text{Vol}_{G[E_j]}(V_t)\right)$ *for any* E_j **do**
7 $\quad\quad R \leftarrow R + 1$
8 $\quad\quad V_t \leftarrow B_G(v, R)$
9 $\quad$ **end**
10 $\quad T \leftarrow$ shortest path tree from v in $B_G(v, R)$
11 $\quad$ **if** $R \geq r$ **then**
12 $\quad\quad E_t \leftarrow$ edges of T contained in $B_G(v, R - r)$
13 $\quad\quad U_t^1, U_t^2, ...U_t^{j_t} \leftarrow$ connected components of $T - E_t$
14 $\quad$ **else**
15 $\quad\quad U_t^1 \leftarrow V_t$
16 $\quad$ **end**
17 $\quad G \leftarrow G - B_G(v, R)$ and $t \leftarrow t + 1$
18 **end**
19 **return** $\{V_i, U_i^j\}_{i,j\geq 1}$

time a partition of G's vertices, $V_1, V_2, \cdots V_\alpha$, and trees, $U_i^1, U_i^2, \cdots, U_i^{j_i}$, whose vertices partition V_i such that

- *For all $i \in [\ell]$, at most $6\beta|E_i| + 6\beta m e^{-\frac{r\beta}{\ell}}$ edges of E_i cross the $V_1, V_2, \cdots, V_\alpha$ partition.*
- *Each U_i^j is a tree of radius r.*
- *The total number of U_i^j, i.e. $\sum_{i\in[\alpha]} j_i$, is at most $\alpha + 4me^{-\frac{r\beta}{\ell}}$.*

Proof. We first bound the running time of Algorithm 1. Observe that any time an edge (u, v) is traversed during the ball growing phase, one of its endpoints is deleted from G: thus we encounter each edge at most once during our traversal. Building the shortest path trees, the V_i, and the U_i^j can be done in $O(m)$ total work given this. Checking the condition in the while loop on Line 6 can be done in $O(m)$ total time by updating the relevant volumes whenever a new vertex is introduced to V_t.

We now prove the correctness of Algorithm 1's output. First, observe that we only cut a cluster V_t of G when for every E_j

$$e^{-\frac{r\beta}{\ell}}\text{Vol}_G(\partial V_t) + \text{Vol}_{G[E_j]}(\partial V_t) < 3\beta\left(e^{-\frac{r\beta}{\ell}}\text{Vol}_G(V_t) + \text{Vol}_{G[E_j]}(V_t)\right).$$

Further, by construction $V_t = B_G(s, R)$ for some choice $s \in V[G], R \geq 0$. We cut $\mathrm{Vol}_{G[E_j]}(\partial V_t)$ from E_j when we partition off V_t: this is therefore at most $3\beta e^{-\frac{r\beta}{\ell}}\mathrm{Vol}_G(V_t)+3\beta\mathrm{Vol}_{G[E_j]}(V_t)$ from E_j, for any j. As the total of all the $\mathrm{Vol}_G(V_t)$ terms for this bound on edges removed from E_j is m and the total of the $\mathrm{Vol}_{G[E_j]}(V_t)$ terms is $2|E_j|$ at the end of the partitioning procedure we cut at most $6\beta|E_j| + 6\beta m e^{-\frac{r\beta}{\ell}}$ edges from E_j.

We now show each generated partition piece V_t contains a forest $U_t^1, ..., U_t^{j_t}$ with the desired properties. Let R_t be the parameter such that $V_t = B_G(s, R_t)$ when cutting it out from the rest of the graph, and let T_t be the spanning tree rooted at s in V_t. If $R_t \leq r$, then only one U_t^i is created and it is equal to T_t. If instead $R_t > r$, the algorithm constructs a forest U_t^i where each subtree has diameter r: each U_t^i is the subtree of T_t from a node at distance $R_t - r$ from s while no node is further than R_t from s. Further, the forest is constructed by deleting at most $\mathrm{Vol}(B_G(s, R_t - r))$ edges from T_t. By the pigeonhole principle, at least one edge partition piece E_j must have passed the expansion condition on Line 7 of the algorithm at least r/ℓ times. Now as

$$\mathrm{Vol}_G(B_H(s, \alpha + 1)) \geq \mathrm{Vol}_G(B_H(s, \alpha)) + \mathrm{Vol}_G(\partial B_H(s, \alpha))$$

for any H a subgraph of G and since volume of balls is monotone increasing, for E_j we observe

$$\begin{aligned} &e^{-\frac{r\beta}{\ell}}\mathrm{Vol}_G(B_G(s, R_t)) + \mathrm{Vol}_{G[E_j]}(B_G(s, R_t)) \\ &\geq (1+3\beta)^{r/\ell}\left(e^{-\frac{r\beta}{\ell}}\mathrm{Vol}_G(B_G(s, R_t - r)) + \mathrm{Vol}_{G[E_j]}(B_G(s, R_t - r))\right) \end{aligned}$$

Now since $0 \leq \beta \leq 1/6$ implies $(1+3\beta) \geq e^{2\beta}$, $(1+3\beta)^{r/\ell} \geq \exp(2\frac{r\beta}{\ell})$. Further,

$$e^{-\frac{r\beta}{\ell}}\mathrm{Vol}_G(B_G(s, R_t)) + \mathrm{Vol}_{G[E_j]}(B_G(s, R_t)) \leq 2\mathrm{Vol}_G(B_G(s, R_t)) = 2\mathrm{Vol}_G(V_t).$$

Substituting in and rearranging we observe

$$2\exp\left(-\frac{r\beta}{\ell}\right)\mathrm{Vol}_G(V_t) \geq \mathrm{Vol}_G(B_G(s, R_t - r)).$$

Thus $U_t^1, ..., U_t^{j_t}$ consists of at most $2\exp(-\frac{r\beta}{\ell})\mathrm{Vol}(V_t)$ trees. Summing over all V_t gives the result. □

2.2 Obtaining Ultrasparse κ-Distortion Subgraphs

Here we leverage the graph decomposition primitive from the previous section to obtain κ-distortion subgraphs consisting of a tree plus a small number of edges. Our algorithm is a modification of a classic algorithm for construction low-stretch spanning trees due to [AKPW95]. Briefly, [AKPW95]'s algorithm on a graph G performs the following steps: it first computes a low-diameter decomposition of G into $V_1, V_2, \ldots$, it then forms a shortest-path tree within each $G[V_i]$, and finally it contracts each V_i to a single node and recurses on the remaining graph. Unfortunately due to aforementioned $\Omega(\log n)$ loss intrinsic to low-diameter decomposition mentioned, any algorithm based on standard low-diameter decomposition is insufficient for our purposes.

To improve, we leverage that the combinatorial stretch bounds given by low-stretch spanning trees are stronger than what is needed for our purposes. We show that adding a tree plus a small number of edges inside each partition piece enables us to obtain effective resistance overestimates that do not lose an $O(\log n)$ factor. For an

explicit demonstration, consider the case of a constant-degree expander G seen in the previous subsection. By growing a shortest-path tree from an arbitrary root and removing the edges in all but the last $O(\log k)$ levels (as is done in Algorithm 1), we see that G contains a forest consisting of $O(n/k)$ trees each of depth at most $O(\log k)$. We then show that by adding $O(\frac{n\text{poly}(\log n)}{k})$ edges to this forest we can ensure every edge in G is retained in the subgraph or possesses many sufficiently disjoint paths between its endpoints. These sufficiently disjoint paths then enable us to bound the effective resistance across every edge in G when measured through the subgraph. Although there could be different ways to add edges and form these disjoint paths, our specific approach will call a near-linear time algorithm for computing path sparsifiers on a graph obtained by contracting low-diameter clusters inside G. Hence, throughout this section we make reference to an abstract algorithm for path sparsification of unweighted graphs (we give a specific instantiation in Section 3).

Definition 2.3 (Path Sparsification Algorithm)**.** *We call an algorithm a $(\mathcal{S}_{PS}, \mathcal{T}_{PS}, \alpha)$-path sparsification algorithm if it takes in an unweighted graph G with n nodes and m edges and returns a $(10\alpha, \alpha)$-path sparsifier for it with $\mathcal{S}_{PS}(m, n)$ edges in $\mathcal{T}_{PS}(m, n)$ time. We assume the functions $\mathcal{S}_{PS}, \mathcal{T}_{PS}$ are supermodular[3] and non-decreasing in both arguments.*

We first give a procedure which returns an ultrasparse subgraph which generates small effective resistance overestimates whenever the input graph's vertices can be partitioned into a small number of low-diameter clusters.

Lemma 2.4. *Let $G = (V, E, w)$ be a weighted n-node m-edge graph with all edge weights at least $w_{\min}$. Let $T_1, T_2, \ldots T_v$ be a spanning forest of G such that each individual tree T_i has effective resistance diameter[4] at most δ. Let* AbstractPathSparsify *be a $(\mathcal{S}_{PS}, \mathcal{T}_{PS}, \alpha)$-path sparsification algorithm (Definition 2.3). Algorithm 2 computes a subgraph G'' such that $H = \bigcup_i T_i \cup G''$ has at most with at most $n + \mathcal{S}_{PS}(m, v)$ edges and for any edge $(u, v) \in E$, $\mathcal{R}_H^{eff}(u, v) \leq 3\delta + 1/w_{\min}$. Further Algorithm 2 runs in time $O(m) + \mathcal{T}_{PS}(m, v)$ if $v > 1$ and $O(m)$ time otherwise.*

Algorithm 2: $G'' =$ AugmentTree$(G, \{T_1, T_2, \ldots T_v\},$ AbstractPathSparsify$)$

Input: Graph $G = (V, E, w)$, $\{T_1, T_2, \ldots T_v\}$ forest in G, AbstractPathSparsify path-sparsification algorithm

Output: Subgraph G''

$G' \leftarrow (V, E, \mathbf{1})$; `// Unweighted copy of G without edge weights`

$\{V_1, V_2, \ldots V_v\}$ = connected components of forest $\{T_1, T_2, \ldots T_v\}$

$G' \leftarrow G' \backslash \{V_1, V_2, \ldots V_v\}$, deleting self-loops

if $G' = \emptyset$ **then return** $\emptyset$;

$G'' \leftarrow$ AbstractPathSparsify(G')

return G'' *with output edges mapped to the original (uncontracted) vertex set*

[3] A function $f(x, y)$ is supermodular if $f(a + b, c + d) \geq f(a, c) + f(b, d)$ for any a, b, c, d.

[4] The effective resistance diameter of a graph H is defined as $\max_{u,v \in H} \mathcal{R}_H^{eff}(u, v)$

Proof. We first bound the runtime of the algorithm. If $\nu = 1$, the algorithm clearly runs in $O(m)$ time: we may thus assume $\nu > 1$. We can compute the contracted graph G' directly in $O(m)$ time. Further, as G' has ν vertices and at most m edges, the cost of the call to AbstractPathSparsify is bounded by $\mathcal{T}_{PS}(m, \nu)$.

Next we bound the number of edges in the graph $H = \bigcup T_i \cup G''$. H consists of a forest $\bigcup_i T_i$ combined with the output of AbstractPathSparsify, with the edges mapped to the original (uncontracted) graph. As AbstractPathSparsify is called on a graph with ν vertices and at most m edges we add at most $\mathcal{S}_{PS}(m, \nu)$ edges to it yielding the claim.

Finally, we prove the bound on $\mathcal{R}_H^{eff}(u, v)$ for any edge $(u, v) \in E(G)$. We analyze this in cases. First, we consider the case when the edge (u, v) is fully contained inside a tree T_i. We observe that T_i has effective resistance diameter at most δ by assumption. Thus as H contains T_i we have by Claim D.1 that $\mathcal{R}_H^{eff}(u, v) \leq \mathcal{R}_{T_i}^{eff}(u, v) \leq \delta$. If instead (u, v) is not contained inside a tree T_i, it must be that u lies in some T_j and v lies in some T_k for $j \neq k$. In this case, we argue that the low resistance diameter of the trees in forest combined with the path sparsifier G'' enables us to certify a bound on $\mathcal{R}_H^{eff}(u, v)$. We observe that the graph G' obtained by contracting every tree T_i contains an edge from T_j to T_k corresponding to (u, v). By the guarantee of path sparsification either this edge is retained in G'', or G'' contains 10α vertex-disjoint paths of length at most α connecting T_j to T_k in G'. In the former case, the edge (u, v) is retained in G'' and hence the output graph H contains (u, v): thus $\mathcal{R}_H^{eff}(u, v) \leq 1$. In the latter case, G'' contains 10α vertex-disjoint paths of length at most α connecting T_j to T_k in G'. Now the edges in G'' correspond to weighted edges in the original input graph and therefore each have weight at least W. As each vertex in G' corresponds to a tree T_i and since each T_i has resistance diameter at most δ, we observe that these paths correspond to paths in H of effective resistance at most $\alpha(\delta + 1/W)$ connecting vertices in V_j to vertices in V_k. By Lemma D.2, bounding the effective resistance in such settings, and Claim D.1, this implies the effective resistance between u and v is bounded by $2\delta + \frac{\delta+1/W}{10} \leq 3\delta + 1/W$. □

We now recursively combine this result with the graph decomposition from Section 2.1 to prove the main result of this section.

Theorem 2.5. *Let $G = (V, E, w)$ be an n-node, m-edge graph with edge weights which are polynomially-bounded in n. Let $k, \gamma \geq 2$ be parameters. Let* AbstractPathSparsify *be an $(\mathcal{S}_{PS}, \mathcal{T}_{PS}, \alpha)$-path sparsification algorithm (in the sense of Definition 2.3) for any α. In $O(m + \mathcal{T}_{PS}\left(O(m), O\left(\frac{m}{k}\right)\right))$ time Algorithm 3 finds a subgraph H with at most*

$$n + O\left(\frac{m}{\gamma} + \frac{m \log \gamma}{k^2}\right) + \mathcal{S}_{PS}\left(O(m), O\left(\frac{m}{k}\right)\right)$$

edges which is a κ-distortion subgraph of G for

$$\kappa = O\left(m \exp\left(\sqrt{8 \log \gamma \cdot \log\left(48 \log k \sqrt{\log \gamma}\right)}\right) \log k \sqrt{\log \gamma}\right)$$

It also returns a vector $\tau \in \mathbb{R}^E$ which satsifies $\tau_{(u,v)} \geq w_{(u,v)} \mathcal{R}_H^{eff}(u, v)$ and $\|\tau\|_1 \leq \kappa$.

Algorithm 3:
$H = \text{SpectralSubgraph}(G = (V, E, w), \text{AbstractPathSparsify}, k, \gamma)$

Input: Graph $G = (V, E, w)$, $(\mathcal{S}_{PS}, \mathcal{T}_{PS}, \alpha)$-path sparsifier oracle AbstractPathSparsify, k, γ parameters
Output: Subgraph H with potentially smaller edge weights satisfying Theorem 2.5, τ leverage score overestimates of the edges in G

Set $r_e \leftarrow w_{\max}/w_e$ for all $e \in E$; // Edge lengths for AKPW
$F \leftarrow \emptyset, S \leftarrow \emptyset, \tau \leftarrow \mathbf{1} \in \mathbb{R}^E$
$\beta = \exp\left(-\sqrt{0.5 \log \gamma \cdot \log\left(48 \log k \sqrt{\log \gamma}\right)}\right), \sigma = \log_{1/\beta} \gamma, \delta = 48\sigma\beta^{-1} \log k$
$E_1, E_2, \cdots E_\ell \leftarrow$ partition of edges where E_i contains all edges with $r_e \in [\delta^{i-1}, \delta^i)$
$t \leftarrow 1$
while *F does not span G or $t \leq \ell$* **do**
  $E' \leftarrow \bigcup_{j=t-\sigma+1}^{t} E_j$
  $G_t \leftarrow G[E'] \backslash F_{t-1}$
  $G'_t \leftarrow (V(G_t), E(G_t), \mathbf{1})$; // Unweighted copy of G_t
  $V_{t,i}, T^j_{t,i} \leftarrow \text{Decompose}(G'_t, \{E_{t-\sigma}, E_{t-\sigma+1} \dots E_t\}, \beta/6, \delta/4)$
  for *each $V_{t,i}$* **do**
    $G_{t,i} \leftarrow \text{AugmentTree}(G_t[V_{t,i}], \{T^1_{t,i}, T^2_{t,i}, \dots\}, \text{AbstractPathSparsify})$
    $F_t \leftarrow F_{t-1} \cup \bigcup_{i,j} T^j_{t,i}$
    $S \leftarrow S \cup G_{t,i}$
    for $e \in G[V_{t,i}]$ **do**
      $\tau_e \leftarrow 4 w_e w_{\max}^{-1} \delta^{t+1}$
      $E_j \leftarrow E_j - \{e\}$
    end
  end
  for $j = t - \sigma + 1, \dots t$ **do**
    $Y_j \leftarrow$ arbitrary subset of $6m/k^2$ edges from each E_j
    $S \leftarrow S \cup Y_j$
    $E_j \leftarrow E_j - Y_j$
  end
  $S \leftarrow S \cup E_{t-\sigma}$
  $t \leftarrow t + 1$
end
$F = F_{t-1}$
for $e \in F \cup S$ **do**
  $\tau_e \leftarrow 1$ // Set stretch overestimate to 1 if in output subgraph
end
return $H = F \cup S, \tau$ // Edges in H are given the same weight they had in G

We remark that the sparsity and runtime guarantees in the above are independent of α. Before we prove this theorem, we state and prove some structural invariants about the algorithm:

Lemma 2.6. *Consider an execution of Algorithm 3, and consider an iteration t of the while loop on Line 6. Each time a weight bucket E_j is included in G_t on Line 8, we conclude that iteration of the while loop by decreasing the number of edges in E_j by a factor of β. In addition, each connected component of F_t is a tree of effective resistance diameter at most $w_{\max}^{-1}\delta^{t+1}$.*

Proof. We first prove that $|E_i|$ decreases by a factor of β each iteration. Let E_i be processed in iteration t, and note that the graph G_t's edges are partitioned into at most σ buckets. For our value $\delta = 48\sigma\beta\log k$, if edge set E_i is processed in iteration t Lemma 2.2 implies the call to Decompose forms a vertex partition $V_{t,i}$ which cuts at most

$$\beta|E_i| + 6|E(G_t)|e^{-\frac{(\delta/4)(\beta/6)}{\sigma}} = \beta|E_i| + 6|E(G_t)|e^{-\frac{\delta\beta}{24\sigma}} \leq \beta|E_j| + 6mk^{-2}$$

edges from E_i, where we used $|E(G_t)| \leq m$. Further, on Line 22 we move $6m/k^2$ edges from E_i to S during every iteration E_i is processed. Thus we see that E_i ends the iteration with at most $\beta|E_i|$ edges as desired.

For the second condition, we induct on t. For $t = 1$, observe that every edge seen in G_t has resistance between 1 and δ: this implies it has weight between $w_{\max}/\delta$ and $w_{\max}$. By the guarantee of Algorithm 1 (Lemma 2.2), the trees $T_{1,i}^j$ have unweighted radius at most $\frac{\delta}{4}$ and hence unweighted diameter at most $\frac{\delta}{2}$. As every edge in these trees has effective resistance at most $\delta w_{\max}^{-1}$ the claim follows. Now assume the claim for $t = \nu$: we will show it for $t = \nu + 1$. The edges in $G_{\nu+1}$ have effective resistance at most $w_{\max}^{-1}\delta^{\nu+1}$. Now, the edges added to $F_{\nu+1}$ belong to $T_{\nu+1,i}^j$: in the unweighted contracted graph $G'_{\nu+1}$ these are trees of (unweighted) diameter at most $\frac{\delta}{2}$ by Lemma 2.2. In G, the edges from $T_{\nu+1,i}^j$ connect together subsets of vertices which correspond to the forests in F_ν: by the induction hypothesis the trees of F_ν have effective resistance diameter at most $w_{\max}^{-1}\delta^{\nu+1}$. Combining these two observations, any path through a forest in $F_{\nu+1}$ travels through at most $\frac{\delta}{2}$ edges with resistance at most $w_{\max}^{-1}\delta^{\nu+1}$ (coming from the edges of $T_{\nu+1,i}^j$) and at most $\frac{\delta}{2}$ forests in F_ν with effective resistance diameter $w_{\max}^{-1}\delta^{\nu+1}$. Adding these together, we obtain an effective resistance overestimate of

$$\frac{\delta}{2}\left(w_{\max}^{-1}\delta^{\nu+1}\right) + \frac{\delta}{2}\left(w_{\max}^{-1}\delta^{\nu+1}\right) = w_{\max}^{-1}\delta^{\nu+2}.$$

□

With Lemma 2.6 established, we prove that Algorithm 3 outputs a low-distortion subgraph:

Lemma 2.7. *Let $G = (V, E, w)$ be a n-node m-edge graph and let H, τ be the output of Algorithm 3 in the setting of Theorem 2.5. Then H is a κ-distortion subgraph of G for*

$$\kappa = O\left(m\exp\left(\sqrt{8\log\gamma\cdot\log\left(48\log k\sqrt{\log\gamma}\right)}\right)\log k\sqrt{\log\gamma}\right)$$

Further, τ satisfies $\tau_{(u,v)} \geq w_{(u,v)}\mathcal{R}_H^{eff}(u,v)$ for any $(u,v) \in E$ and $\|\tau\|_1 \leq \kappa$.

Proof. Since H is clearly a subgraph of G it suffices to show that $\|\tau\|_1 \leq \kappa$ and $\tau_{(u,v)} \geq w_{(u,v)}\mathcal{R}_H^{eff}(u,v)$ for any $(u,v) \in E(G)$. We do this by using the effective resistance guarantee of Algorithm 2 to certify resistance bounds on edges contained within a partition piece $V_{t,i}$.

Since H is a subgraph of G, every edge of G that is added H receives $\tau_{(u,v)} = 1 \geq w_{(u,v)}\mathcal{R}_H^{eff}(u,v)$. Now let $e = (u,v) \in E(G)$ be contained in some $V_{t,i}$. We will show that the forest F_t combined with the path sparsifier $G_{t,i}$ computed with AugmentTree give e an effective resistance overestimate of $4w_{\max}^{-1}\delta^{t+1}$.

First, observe that each vertex in G_t corresponds to a tree in F_{t-1} and therefore $V_{t,i}$ corresponds to a subset of those trees. Since $T_{t,i}^1, T_{t,i}^2, \ldots$ are trees inside $G_t[V_{t,i}]$, we see that every tree in F_t is fully contained in some $V_{t,i}$. Let $F_{t,i} = F_t[V_{t,i}]$, and note that by Lemma 2.6 each tree in $F_{t,i}$ has effective resistance diameter $w_{\max}^{-1}\delta^{t+1}$ when the edges are given the edge weights they have in G. In addition, the edges in $G_t[V_{t,i}]$ which we pass into AugmentTree have weight at least $w_{\max}\delta^{-t}$ in G. Therefore, Lemma 2.4 ensures that any for any edge (u,v) in $V_{t,i}$ we have

$$\mathcal{R}_H^{eff}(u,v) \leq \mathcal{R}_{F_t \cup G_{t,i}}^{eff}(u,v) \leq 3w_{\max}^{-1}\delta^{t+1} + w_{\max}^{-1}\delta^t \leq 4w_{\max}^{-1}\delta^{t+1}.$$

We remark that this corresponds to the value given to $\tau_{(u,v)}$ in the algorithm.

The above shows that the effective resistance through H across the endpoints of any edge contained in a $V_{t,i}$ is bounded. We now fix a weight bucket E_j and bound E_j's contribution to H's spectral distortion. Observe that the number of edges in E_j which are not within a $V_{t,i}$ in iteration $t = j + x$ of the algorithm is at most $|E_j|\beta^x$ by Lemma 2.6. As the weight of any edge in E_j is at most $w_{\max}\delta^{1-j}$, we see that at most $|E_j|\beta^x$ end up receiving a $\tau_{(u,v)}$ value larger than

$$\left(w_{\max}\delta^{1-j}\right)\left(4w_{\max}^{-1}\delta^{j+x+1}\right) = 4\delta^{2+x}.$$

Therefore, the edges in E_j get effective resistance overestimates summing to at most

$$4|E_j|\sum_{x=0}^{\sigma-1}\beta^x\delta^{(2+x)}$$

since after σ iterations we add the remaining edges in E_j to H. This is at most

$$\begin{aligned}4|E_j|\sum_{x=0}^{\sigma-1}\beta^x\delta^{(2+x)} &= 4|E_j|\sum_{x=0}^{\sigma-1}\delta^2\left(48\sigma\log k\right)^x \\ &\leq 5|E_j|\delta^2\left(48\sigma\log k\right)^{\sigma-1} = 5|E_j|\beta^{-2}\left(48\sigma\log k\right)^{\sigma+1}\end{aligned}$$

where we used

$$\begin{aligned}48\sigma\log k = \frac{48\log k\log\gamma}{-\log\beta} &= \frac{48\log k\log\gamma}{\sqrt{0.5\log\gamma\cdot\log\left(48\log k\sqrt{\log\gamma}\right)}} \\ &\geq \frac{48\log k\sqrt{\log\gamma}}{\log\left(48\log k\sqrt{\log\gamma}\right)} \geq 5\end{aligned}$$

for $k, \gamma \geq 2$ and $\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1} \leq \frac{5}{4}c^n$ for $c \geq 5$. Our choice of β yields $\log 1/\beta = \sqrt{0.5 \log \gamma \cdot \log\left(48 \log k \sqrt{\log \gamma}\right)}$: this implies

$$\sigma = \frac{\log \gamma}{\log 1/\beta} = \frac{\log \gamma}{\sqrt{0.5 \log \gamma \cdot \log\left(48 \log k \sqrt{\log \gamma}\right)}} \leq \sqrt{\log \gamma}.$$

Thus we have

$$\begin{aligned}
\beta^{-2} \left(48\sigma \log k\right)^{\sigma+1} &\leq \beta^{-2} \left(48 \log k \sqrt{\log \gamma}\right) \left(48 \log k \sqrt{\log \gamma}\right)^{\frac{\log \gamma}{\log 1/\beta}} \\
&= \left(48 \log k \sqrt{\log \gamma}\right) \exp\left(2 \log 1/\beta + \frac{\log \gamma}{\log 1/\beta} \log\left(48 \log k \sqrt{\log \gamma}\right)\right) \\
&= 48 \exp\left(\sqrt{8 \log \gamma \cdot \log\left(48 \log k \sqrt{\log \gamma}\right)}\right) \sqrt{\log \gamma} \log k.
\end{aligned}$$

The last equality used the definition of β and the algebraic fact that $c_1 x + c_2/x = 2\sqrt{c_1 c_2}$ for $x = \sqrt{c_2/c_1}$. Substituting this in yields that E_j's contribution to the spectral distortion of H is bounded by

$$240|E_j| \exp\left(\sqrt{8 \log \gamma \cdot \log\left(48 \log k \sqrt{\log \gamma}\right)}\right) \log k \sqrt{\log \gamma}.$$

implying the claimed bound. □

Finally, we bound the runtime and sparsity guarantees of Algorithm 3:

Lemma 2.8. *Let G be a weighted graph with n nodes and m edges, and let H be the output of Algorithm 3 in the setting of Theorem 2.5. Then H has at at most*

$$n + O\left(\frac{m}{\gamma} + \frac{m \log \gamma}{k^2}\right) + \mathcal{S}_{PS}\left(O(m), O\left(\frac{m}{k}\right)\right)$$

edges. Further, Algorithm 3 runs in time $O(m + \mathcal{T}_{PS}\left(O(m), O\left(\frac{m}{k}\right)\right)$.

Proof. Observe that there are four different ways edges can be added to H: they can be added to the forest F_t on Line 13, to S through the path sparsifiers $G'_{t,i}$ on Line 14, to S via the extra edges from each E_j we keep on Line 22, or to S by the check on Line 25 (after a weight bucket has been processed in a G_t sufficiently many times). We bound these in order. Clearly, the returned forest F consists of at most n edges. By the guarantee of Algorithm 2, the calls to AugmentTree during iteration t are on graphs $G_t[V_{t,i}]$: let $m(t)$ denote the total number of edges contained in the $G_t[V_{t,i}]$. We observe that G_t's edge set is a subset of $\bigcup_{j=t-\sigma}^{t} E_j$, and moreover each edge in E_j is contained inside at most one $G_t[V_{t,i}]$ (as once this happens we delete it from our edge set): thus $\sum_t m(t) \leq m$. Next, each $G_t[V_{t,i}]$ contains a forest $T^l_{t,i}$: by the guarantee of Decompose we observe that the total number of $T^l_{t,i}$ is the number of $V_{t,i}$ plus at most $4m(t)e^{-\frac{(\delta/4)\cdot(\beta/6)}{\sigma}} = 4m(t)e^{-2\log k} \leq 4m(t)/k$. Now as AbstractPathSparsify is called inside AugmentTree only when $T^l_{t,i}$ consists of more than 1 tree, we may aggregate all the calls to AbstractPathSparsify in iteration t into a single call on a graph with $m(t)$

edges and $O(m(t)/k)$ nodes. Since $\mathcal{S}_{PS}$ is supermodular and non-decreasing in both arguments, we see that the number of edges added to H via path sparsifiers is at most

$$\sum_t \mathcal{S}_{PS}\left(m(t), O\left(\frac{m(t)}{k}\right)\right) \leq \mathcal{S}_{PS}\left(m, O\left(\frac{m}{k}\right)\right).$$

For the edges added on Line 22, we again see that each E_j is included in E' at most σ times– thus this collectively adds $O(\frac{m\sigma}{k^2}) \leq O(\frac{m\log\gamma}{k^2})$ edges. Finally, after an E_j has been processed σ times we observe that it decreases in size by a factor of γ: thus the addition on Line 25 adds $O(m/\gamma)$ edges to H. Combining gives our claimed size bound.

Finally, we prove the running time of our algorithm. We first bound the cost of all steps excluding the calls to AugmentTree. Each time a weight class is fed to Decompose on Line 10, the number of edges in that class falls by a factor of β. Thus, the total contribution of weight class E_j to the running time of all calls to Decompose is only $O(m)$: the number of edges left to consider in the recursively generated subproblems falls geometrically. The time it takes to sort the edges into weight buckets E_j is at most $O(m)$ via radix sort with base poly(n) (here we use our assumption of polynomially-bounded edge weights) and the running time of every other step in the algorithm can be implemented in the trivial fashion in $O(m)$ time. Finally, to bound the runtime of the calls to AugmentTree we again observe that during iteration t we can aggregate the nontrivial calls it makes to AbstractPathSparsify to a single one on a graph with at most $m(t)$ edges and $O(m(t)/k)$ vertices. As $\mathcal{T}_{PS}$ is also superlinear in both arguments, the runtime of these calls to AbstractPathSparsify can again be bounded by $\mathcal{T}_{PS}(O(m), O(m/k))$ as desired. □

Combining the last two lemmas gives Theorem 2.5. To conclude this section, we show that combining Theorem 2.5 with the path sparsification algorithm construction in Section 3 and choosing parameters appropriately yields an efficient construction of κ-distortion subgraphs.

Theorem 1.5 (Efficient Construction of Ultrasparse κ-Distortion Subgraphs). *Let $G = (V, E, w)$ be a polynomially-bounded weighted graph, and let $c \geq 1$ be any fixed constant. Algorithm 3 equipped with Theorem 1.9 runs in $O(m)$ time and returns a κ-distortion subgraph H with $n + O\left(\frac{m}{(\log\log n)^c}\right)$ edges, for*

$$\kappa = O\left(m\,(\log\log n)^{\sqrt{8c}+1+o(1)}\right).$$

It also returns a vector $\tau \in \mathbb{R}^E_{\geq 0}$ with $\|\tau\|_1 \leq \kappa$ where for any $e \in G$, $\tau_e \geq w_e \delta_e^\top \mathcal{L}_H^\dagger \delta_e$ is an overestimate of the leverage score of e measured through H.

Proof. Let PathSparsify$(G, O(\log^5 n))$ be the algorithm guaranteed by Theorem 1.9: note that this is a $(O(n\log^8 n), O(m + n\log^{18} n), O(\log^5 n))$-path sparsification algorithm. We apply SpectralSubgraph to G with parameters $k = \log^{18} n$ and $\gamma = (\log\log n)^c$. Combining this with the guarantee of Theorem 2.5, we therefore see that H contains

$$n + O\left(\frac{m\log^{10} n}{k} + \frac{m\log\gamma}{k^2} + \frac{m}{\gamma}\right) = n + O\left(\frac{m}{(\log\log n)^c}\right)$$

edges, and it is computed in time

$$O\left(m(\log\log n)^{1/2} + \frac{m\log^{18} n}{\log^{18} n}\right) = O\left(m(\log\log n)^{1/2}\right).$$

It remains to bound the spectral distortion of the computed subgraph. We use the notation $\log^{(i)}(n)$ to denote the result of applying the log function i times: hence $\log^{(1)}(n) = \log n$ and $\log^{(i+1)} n = \log\left(\log^{(i)} n\right)$. For our values of γ and k we see

$$\begin{aligned}\sqrt{8\log\gamma \cdot \log\left(48\log k\sqrt{\log\gamma}\right)} &= \sqrt{8c\log^{(3)} n \cdot \left(\log^{(3)} n + O(\log^{(4)} n)\right)} \\ &= \left(\sqrt{8c} + o(1)\right)\log^{(3)} n.\end{aligned}$$

Thus by Theorem 2.5 our computed subgraph is a κ-distortion subgraph for

$$\begin{aligned}\kappa &= O\left(m\exp\left((\sqrt{8c}+o(1))\log^{(3)} n\right)\log^{(2)} n\sqrt{\log^{(3)} n}\right) \\ &= O\left(m\left(\log\log n\right)^{\sqrt{8c}+1+o(1)}\right).\end{aligned}$$

Assembling these pieces yields the claim. □

3 EFFICIENT PATH SPARSIFICATION

In this section we prove Theorem 1.9 showing that path sparsifiers can be efficiently computed. To prove this result we provide several new algorithmic components of possible independent interest. First, in Section 3.1 we show that dense near-regular expanders have many short vertex disjoint paths. Then, in Section 3.2 we leverage this result with a new sampling scheme and previous expander partitioning results to show that we can efficient path-sparsify large amount of the volume of dense near-regular graphs. In Section 3.3 we then show that every dense graph can be efficiently decomposed into dense near-regular subgraphs. Carefully applying these tools yields our desired result in Section 3.4.

3.1 Short Vertex Disjoint Paths in Expanders

In this section we show that in every dense expander of balanced degrees, for every pair of vertices s and t there are many vertex disjoint paths between them (where here and throughout we ignore the necessary shared use of s and t). The number of paths and the length of these paths depend on the degree ratio and the conductance of the graph. Formally we define conductance, Definition 3.1, and present the main theorem of this section, Theorem 3.2 below.

Definition 3.1 ((Edge) Conductance)**.** *For undirected graph $G = (V, E)$ (possibly with self-loops) and $S \subseteq V$ we define the* (edge) conductance *of S and G by*

$$\phi_{\text{edge}}(S) \coloneqq \frac{|\partial(S)|}{\min\{\text{Vol}(S), \text{Vol}(V \setminus S)\}} \text{ and } \phi_{\text{edge}}(G) \coloneqq \min_{S\subseteq V, S\notin\{\emptyset,V\}} \phi_{\text{edge}}(S) \text{ respectively.}$$

We call any family of n-node graphs G with $\phi_{\text{edge}}(G) = \Omega(\log^c n)$ for some c, *expanders*. Our main result of this section is the following theorem regarding vertex disjoint paths in such expanders.

Theorem 3.2 (Short Vertex Disjoint Paths in Approximately Regular Expanders). *For all pairs of vertices s and t in an n-node undirected graph $G = (V, E)$ (possibly with self-loops) there is a set of at least $\phi_{\text{edge}}(G)d_{\min}(G)/(8d_{\text{ratio}}(G))$ vertex-disjoint paths from s to t of length at most $(4d_{\text{ratio}}(G)/\phi_{\text{edge}}(G)) \cdot \log(n/d_{\min}(G))$.*

Our main technical tool towards proving Theorem 3.2 is that graphs with large vertex conductance have many short vertex-disjoint paths. The formal definition of vertex conductance, Definition 3.3, and this tool, Lemma 3.4, are given below.

Definition 3.3 (Vertex Conductance). *For undirected graph $G = (V, E)$ and $S \subseteq V$ we define the* vertex conductance *of S and G by*

$$\phi_{\text{vert}}(S) := \frac{|N(S) \setminus S|}{\min\{|S|, |V \setminus S|\}} \quad \text{and} \quad \phi_{\text{vert}}(G) := \min_{S \subseteq V, S \notin \{\emptyset, V\}} \phi_{\text{vert}}(S) \quad \text{respectively.}$$

Note that for any set S with $|S| = \lceil |V|/2 \rceil$ we have $\phi_{\text{vert}}(S) \leq 1$. Consequently $\phi_{\text{vert}}(G) \in [0, 1]$ for all undirected G.

Lemma 3.4 (Short Vertex-Disjoint Paths in Vertex Expanders). *Let $G = (V, E)$ be an n-node undirected graph with $\phi_{\text{vert}}(G) \geq \phi$. Then for all nodes $s, t \in V$ with $s \neq t$ of degree at least d there is a set of at least $\phi d/8$-vertex disjoint paths from s to t of length at most $(4/\phi)\log(n/d)$.*

Lemma 3.4 implies Theorem 3.2 by a standard technique of relating edge and vertex expansion.

Proof of Theorem 3.2. By Lemma 3.4 it suffices to show that $\phi_{\text{vert}}(G) \geq \phi_{\text{edge}}(G)/d_{\text{ratio}}(G)$. To prove this, let $S \subseteq V$ be arbitrary and note that by assumption $\text{Vol}(S) \geq d_{\min}(G)|S|$, $\text{Vol}(V \setminus S) \geq d_{\min}(G)|V \setminus S|$, and $|\partial(S)| \leq d_{\max}(G)|N(S) \setminus S|$. Consequently,

$$\phi_{\text{vert}}(S) = \frac{|N(S) \setminus S|}{\min\{|S|, |V \setminus S|\}} \geq \frac{(d_{\max}(G))^{-1}|\partial(S)|}{(d_{\min}(G))^{-1}\min\{\text{Vol}(S), \text{Vol}(V \setminus S)\}} = \frac{\phi_{\text{edge}}(S)}{d_{\text{ratio}}(G)}.$$

The claim then follows by the definition of $\phi_{\text{vert}}(G)$ and $\phi_{\text{edge}}(G)$. □

Consequently, in the rest of this section, we prove Lemma 3.4. Our inspiration for this lemma is the seminal result of [KR] which proved an analogous result edge-disjoint paths in expanders. Their proof considered the minimum cost flow problem of routing flow of minimum total length between s and t and by reasoning about primal and dual solutions to this linear program, they obtained their result. We prove Lemma 3.4 similarly, by considering the minimum-length flow in the natural directed graph which encodes vertex-disjointness defined as follows.

Definition 3.5 (Directed Representation of Vertex Capacitated Graphs). *Given undirected graph $G = (V, E)$ we let $\overrightarrow{G} := (\overrightarrow{V}, \overrightarrow{E})$ denote the directed graph where for each $a \in V$ we have vertices a^{in}, and a^{out} and edge $(a^{\text{in}}, a^{\text{out}})$ and for each edge $\{a, b\} \in E$ we have edges $(b^{\text{out}}, a^{\text{in}})$ and $(a^{\text{out}}, b^{\text{in}})$.*

Note that any path of vertices $a, b, c, d, e \in V$ has an associated path $a_{\text{out}}, b_{\text{in}}, b_{\text{out}}, c_{\text{in}}, c_{\text{out}}, d_{\text{in}}, d_{\text{out}}, d_{\text{in}} \in \overrightarrow{V}$ path in $\overrightarrow{G}$. Further a set of a to b paths are vertex-disjoint in G if and only if their associated paths are edge-disjoint in $\overrightarrow{G}$. Also a simple path in G

has length k if and only if its associated path in $\overrightarrow{G}$ has length $2k-1$. Consequently, to reason about short vertex disjoint paths in G it suffices to reason about short edge-disjoint paths in $\overrightarrow{G}$.

To reason about the length of these paths, as in [KR] we consider the minimum cost flow problem corresponding to sending a given given amount of flow from s to t while using the fewest number of edges. However, unlike [KR] the graph we use is directed, i.e. $\overrightarrow{G}$, and thus we need to characterize the minimizers of a slightly different minimum cost problem. This optimality characterization of the minimum cost flow problem is given below and proven in Appendix B.

Lemma 3.6 (Dual Characterization of Shortest Flow)**.** *For directed graph $G = (V, E)$ and vertices $s, t \in V$ if there are at most F edge-disjoint paths from s to t then there is an integral s-t flow $f \in \{0,1\}^E$ corresponding to F edge-disjoint paths from s to t using a minimum number of edges and $v \in \mathbb{R}^V$ such that for every $(a,b) \in e$ if $f_e = 1$ then $v_a - v_b \geq 1$ and if $f_e = 0$ then $v_a - v_b \leq 1$.*

As in [KR], our approach to showing that the paths are short is to sweep over v and show that the associated sets increase rapidly. Here, we tailor the analysis to the structure of our directed (as opposed to undirected) minimum cost flow problem. Our main structural lemma is given below.

Lemma 3.7 (Characterization of Shortests Flow)**.** *Let $G = (V, E)$ be an undirected n-node graph for which there are F vertex disjoint paths from $s \in V$ to $t \in V$. Further, let $f \in \{0,1\}^{\overrightarrow{E}}$ be an integral flow corresponding to F disjoint paths from s_{out} to t_{in} in $\overrightarrow{G}$ using a minimum number of edges and let $v \in \mathbb{R}^{\overrightarrow{V}}$ be as described in Lemma 3.6. The following properties hold:*

1. *The values of v decrease monotonically along each path in f and decrease by at least 1 after each edge. Consequently, the length of each path is at most $y_{s^{\text{out}}} - y_{t^{\text{in}}}$.*
2. *For all $\alpha \in [y_{s^{\text{out}}}, y_{t^{\text{in}}}]$ if $S_\alpha^{\overrightarrow{V}} := \{a \in \overrightarrow{V} | y_a \leq \alpha\}$ and $S_\alpha^V := \{a \in V | a^{\text{in}} \in S_\alpha^{\overrightarrow{V}} \text{ or } a^{\text{out}} \in S_\alpha^{\overrightarrow{V}}\}$ then either $|S_{\alpha-2}^V| \geq n/2$ or $|S_{\alpha-2}^V| \geq (1+\phi_{\text{vert}}) \cdot (|S_\alpha^V| - F)$.*
3. *We have $|S_{y_{s^{\text{out}}}-1}^V| \geq \deg(s) + 1 - F$*

Proof. *Claim 1*: By Lemma 3.6, whenever $f_e = 1$ for $e = (a,b)$ then $v_a - v_b \geq 1$. Combined with the fact that f corresponds to disjoint s to t paths immediately yields the claim.

Claim 2: By claim 1, the set of edges leaving $S_\alpha^{\overrightarrow{V}}$ is of size exactly F. Consequently, leveraging that if $f_e = 0$ for $e = \{a,b\}$ we have $v_b \geq v_a - 1$ by Lemma 3.6 we have that if $a \in S_\alpha^V$, then either (1) $a^{\text{in}} \in S_\alpha^{\overrightarrow{V}}$, $f_{(a^{\text{in}}, a^{\text{out}})} = 1$, and $a^{\text{out}} \notin S_\alpha^{\overrightarrow{V}}$ or (2) $a^{\text{out}} \in S_{\alpha-1}^{\overrightarrow{V}}$. Further, if $a^{\text{out}} \in S_{\alpha-1}^V$ and $b \in N(a)$ and $f_{(a^{\text{out}}, b^{\text{in}})} = 0$ then we have $b^{\text{in}} \in S_{\alpha-2}^V$, again by Lemma 3.6. Consequently, if there are ℓ vertices for which case (1) holds then the neighbors of all the other $|S_\alpha^V|$ vertices are in $|S_{\alpha-2}^V|$ except for at most $F - \ell$ vertices. The claim then follows as $\ell \in [0, F]$ and

$$|S_{\alpha-2}^V| \geq (1+\phi_{\text{vert}}) \cdot (|S_\alpha^V| - \ell) - (F - \ell) \geq (1+\phi_{\text{vert}})(|S_\alpha^V| - F)$$

Claim 3: Again by Lemma 3.6 we have $b^{\text{in}} \in S_{y_{s^{\text{out}}}-1}^V$ for all b^{in} where $\{a,b\} \in E$ and $f_{(a^{\text{out}}, b^{\text{in}})} = 0$, which happens for all but F edges.

□

We now have everything we need to prove Lemma 3.4

Proof of Lemma 3.4. Let $T \coloneqq (N(s) \cap N(t)) \setminus \{s, t\}$. Note that there are $|T|$ vertex disjoint paths of length at most 2 from s to t (ignoring the shared use of s and t). Further, we see that $N(s) \setminus T$ and $N(t) \setminus T$ each have size at least $d - |T|$ and by assumption of vertex conductance this implies that every set S with $(N(s) \setminus T) \subset S$ and $S \cap (N(t) \setminus T) = \emptyset$ has $|N(S) \setminus S| \geq \phi(d - |T|)$ and therefore if we further constrain that $S \cap (T \setminus \{s, t\}) = \emptyset$ this implies that $|(N(S) \setminus (S \cup T))| \geq \phi(d - |T|) - |T|$. By maxflow minimum cut theorem on $\overrightarrow{G}$ this implies that there are at least $\phi d - (1+\phi)|T|$ disjoint paths from s to t, not using T. Consequently, the number of vertex disjoint paths in total from s to t is

$$\max\{\phi d - (1+\phi)|T|, 0\} + |T| = \max\{\phi(d - |T|), |T|\} \geq \min_{\alpha \geq 0} \max\{\phi(d - \alpha), \alpha\}$$
$$= \left(\frac{\phi}{1+\phi}\right) d \geq \frac{\phi d}{2} .$$

It simply remains to bound the length of a smaller set of paths.

To bound the length of these paths, let $F = \phi d/8$. Further, let $f \in \{0, 1\}^{\overrightarrow{E}}$ be an integral flow corresponding to F disjoint paths from s^{out} to t^{in} in $\overrightarrow{G}$ and let $v \in \mathbb{R}^{\overrightarrow{V}}$ be described as in Lemma 3.6. We now prove by induction that for all $t \geq 0$ it is the case that either

$$|S^V_{y_s - 1 - 2t}| \geq n/2 \quad \text{or} \quad |S^V_{y_s - 2t}| \geq (1 + (\phi/2))^t d/2 \tag{2}$$

Note that for the base case we have that

$$|S^V_{y_s - 2}| \geq d + 1 - F \geq d/2 .$$

For the inductive case note that if the claim holds for t and it is not the case that $|S^V_{y_s - 1 - 2(t+1)}| \geq n/2$ then $F \leq |S^V_{y_s - 1 - 2t}| \cdot \phi/4$ and consequently, by Lemma 3.7 and the inductive hypothesis we have

$$|S^V_{y_s - 1 - 2(t+1)}| \geq (1+\phi) \cdot (|S_{y_s - 1 - 2t}| - F) \geq (1+\phi)(1-\phi/4)|S_{y_s - 1 - 2t}| \geq (1 - (\phi/2))|S_{y_s - 1 - 2t}|$$

where we used $\phi \in (0, 1)$. Consequently by induction (2) holds for all $t > 0$ and for some $t \leq \frac{2}{\phi} \log(n/d)$ we have $|S^V_{y_s - 1 - 2t}| \geq n/2$. By symmetry this also implies that $y_t \leq y_s + 2 + \frac{4}{\phi} \log(n/d)$. Since a path of length k in G is length $2k - 1$ in $\overrightarrow{G}$ the result then follows again by Lemma 3.7. □

3.2 Path Sparsification on Dense Near-Regular Expanders

Here we provide an efficient procedure to compute path sparsifiers of a constant fraction of the edges in a dense degree regular graph. This procedure leverages Theorem 3.2, which shows that dense near-regular expanders have many short vertex disjoint paths. Coupled with a procedure for partitioning a graph into approximately regular subgraphs (Section 3.3) this yields our main theorem of this section, an efficient path sparsification procedure. Consequently, in the remainder of this subsection our goal is to prove the following theorem.

Theorem 3.8 (Partial Path Sparsification of Nearly Regular Graphs). *Given any n-node m-edge undirected unweighted graph $G = (V, E)$ and $k \geq 1$, the procedure* PartialPathSparsify(G, k) *(Algorithm 4) in time $O(m + nk \cdot d_{\text{ratio}}(G) \log^8(n))$, outputs $F, E_{\text{cut}} \subseteq E$ such that w.h.p. in n*

- (Size Bound)*: $|F| = O(nk \cdot d_{\text{ratio}}(G) \log(n))$, $|E_{\text{cut}}| \leq |E|/2$ and*
- (Path Sparsification)*: $G[F]$ is a $(\Omega(k/(d_{\text{ratio}}(G) \log^2(n))), O(d_{\text{ratio}}(G) \log^4(n)))$-path sparsifier of $(V, E \setminus E_{\text{cut}})$.*

Our partial path sparsification procedure, PartialPathSparsify(G, k) (Algorithm 9), works simply by randomly sampling the edges, partition the resulting graph into expanders, and output the edges of those expanders as a path sparsifier of the edges on those node induced subgraphs in the original graph. In the following lemma we give basic properties of this random sampling procedure, Lemma 3.9, which is reminiscent of the sublinear sparsification result of [Lee14]. After that we give the expander partitioning procedure we use, Theorem 3.10 from [SW] and our algorithm, Algorithm 9. We then prove Theorem 3.8 by showing that the expanders found have sufficiently many vertex disjoint paths by Theorem 3.2 and the right size properties by Theorem 3.10

Lemma 3.9. *Let $G = (V, E)$ be an unweighted n-node graph, $d \leq d_{\min}(G)$, and let H be a graph constructed by sampling every edge from G with probability $p = \min\{1, \Theta(d^{-1} \log n)\}$ for some parameter $d \leq d_{\min}(G)$. With high probability in n*

$$\frac{1}{2}\mathcal{L}_H - \frac{pd}{n}\mathcal{L}_{K_n} \preceq p\mathcal{L}_G \preceq \frac{3}{2}\mathcal{L}_H + \frac{pd}{n}\mathcal{L}_{K_n}.$$

and

$$\deg_H(a) \in \left[\frac{p}{2} \cdot \deg_G(a)\ ,\ 2p \cdot \deg_G(a)\right] \text{ for all } a \in V\ . \tag{3}$$

Proof. Define auxiliary graph G' with $\mathcal{L}_{G'} = \mathcal{L}_G + \frac{2d}{n}\mathcal{L}_{K_n}$. We observe that for any nodes u, v,

$$\mathcal{R}^{\text{eff}}_{G'}(u, v) \leq \frac{n}{2d}\mathcal{R}^{\text{eff}}_{K_n}(u, v) = \frac{1}{d}.$$

Thus, we interpret our sampling procedure to construct H as sampling edges from G' with the leverage score overestimates

$$\widetilde{\mathcal{R}^{\text{eff}}}(e) = \begin{cases} 1/d & \text{if } e \in G \\ 1 & \text{if } e \in \frac{2d}{n}K_n. \end{cases}$$

We observe that these values are valid leverage score overestimates in G': thus sampling and reweighting the edges in G with probability p and preserving the edges in $\frac{2d}{n}K_n$ produces a graph H' with $\mathcal{L}_{H'} = \frac{1}{p}\mathcal{L}_H + \frac{2d}{n}\mathcal{L}_{K_n}$ such that $\frac{1}{2}\mathcal{L}_{H'} \preceq \mathcal{L}_{G'} \preceq \frac{3}{2}\mathcal{L}_{H'}$ (see, e.g. Lemma 4 [CLM+14]). Rearranging yields the desired

$$\frac{1}{2}\mathcal{L}_H - \frac{pd}{n}\mathcal{L}_{K_n} \preceq p\mathcal{L}_G \preceq \frac{3}{2}\mathcal{L}_H + \frac{pd}{n}\mathcal{L}_{K_n}$$

Finally, (3) follows by an application of the Chernoff bound to the number of edges picked incident to each node in G. Since $d \leq d_{\min}(G)$ this number concentrates around its expected value with high probability in n for appropriate choice of constant in the assumption of p. □

Theorem 3.10 (Expander Decomposition [SW]). *There is a procedure* ExpanderDecomp(G) *which given any m-edge graph $G = (V, E)$, in time $O(m \log^7 m)$ with high probability outputs a partition of V into $V_1, V_2, \cdots V_k$ such that*

- $\phi_{\text{edge}}(G\{V_i\}) \geq \Omega(1/\log^3(m))$ *for all* $i \in [k]$,
- $\sum_i |\partial_G(V_i)| \leq m/8$,

where $G\{V_i\}$ denotes the induced subgraph of G with self-loops added to vertices such that their degrees match their original degrees in G.

Proof. This is a specialization of Theorem 1.2 from [SW] where ϕ in that theorem is chosen to be $\Theta(1/\log^3(m))$. □

Algorithm 4: $F =$ PartialPathSparsify$(G, k \geq 1)$

Input: $G = (V, E)$ simple input graph with degrees between $d_{\min}$ and $d_{\max}$
Output: (F, E_{cut}) such that F is a path sparsifier for $(V, E \setminus E_{\text{cut}})$ and $|E_{\text{cut}}| \leq |E|/2$

`// Sample edges uniformly at random to apply Lemma 3.9`
1 $d := \frac{d_{\min}(G)}{10k}$ and $p = \min\{1, \Theta(d^{-1} \log n)\}$ (where the constant in Θ is as in Lemma 3.9);
2 **if** $p = 1$ **then return** $(E, \emptyset)$;
3 $E' = \emptyset$;
4 **for** $e \in E$ **do** Add e to E' with probability p. ;
5 $G' = (V, E')$;
`// Find expanders and use their edges as a path sparsifier`
6 $(V_1, V_2, \cdots V_r) \leftarrow$ ExpanderDecomp(G') ; `// Computed via Theorem 3.10`
7 For all $i \in [r]$ let $G_i = (V_i, E_i) := G'[V_i]$;
8 Let $E_{\text{cut}} := \cup_{i \in [r]} \partial_G(V_i)$;
9 Let $F := \cup_{i \in [r]} E_i$;
10 **return** (F, E_{cut});

Proof of Theorem 3.8. First, suppose that $p = 1$. In this case, by Line 2 the algorithm outputs $F = E$ and $E_{\text{cut}} = \emptyset$ in linear time. Consequently, F is a path sparsifier of desired quality with m edges and $|E_{\text{cut}}| \leq |E|/2$. Since, in this case $d^{-1} \log n = \Omega(1)$ we have $d_{\min}(G) = O(k \log n)$ and the theorem follows as

$$2pm \leq O(n d_{\max}(g)) = O(n d_{\text{ratio}}(G) d_{\min}(G)) = O(nk \cdot d_{\text{ratio}}(G) \log(n)) ,$$

Consequently, in the remainder of the proof we assume $p < 1$.

Next, note that by design, G' was constructed so Lemma 3.9 applies. Consequently, with high probability in n the following hold:

- G' is an edge-subgraph of G satisfying $p\mathcal{L}_G \preceq \frac{3}{2}\mathcal{L}_{G'} + \frac{pd}{n}\mathcal{L}_{K_n}$.
- $\deg_H(a) \in \left[\frac{p}{2} \cdot \deg_G(a) \, , \; 2p \cdot \deg_G(a)\right]$ for all $a \in V$.
- G' contains at most $2pm$ edges.

Leveraging the bounds we prove the path sparsification property and size bound for F. By Theorem 3.10 we have that for all $i \in [k]$, the V_i output by ExpanderDecomp satisfy that $\Phi_{G'\{V_i\}} = \Omega(1/\log^3(m))$ for all $i \in [k]$. Further, by the above properties of G' we have that $d_{\min}(G'\{V_i\}) \geq \frac{p}{2} \cdot d_{\min}(G)$ and $d_{\max}(G'\{V_i\}) \leq 2p \cdot d_{\max}(G)$. Consequently, by Theorem 3.2 and the fact that $p < 1$ we have that $G'\{V_i\}$ has at least

$$\Omega\left(\frac{d_{\min}(G)p}{d_{\text{ratio}}(G)\log^3(n)}\right) = \Omega\left(\frac{k}{d_{\text{ratio}}(G)\log^2(n)}\right)$$

vertex disjoint paths from s to t of length at most $O(d_{\text{ratio}}(G)\log^4(n))$ for any $s, t \in V_i$. Further, since every edge in $(F, E \setminus E_{\text{cut}})$ has both endpoints in some V_i we see that F is a path sparsifier as desired. Further, F has at most

$$2pm = O(nd_{\max}(G)\log(n)k/d_{\min}) = O(nk\log(n)d_{\text{ratio}}(G))$$

edges by the properties above.

Next, we bound the size of E_{cut}. Note that

$$|E_{\text{cut}}| = \frac{1}{2}\sum_{i\in[r]}|\partial_G(V_i)| = \frac{1}{2}\sum_{i\in[r]} v_i^\top \mathcal{L}_G(V_i)v_i \tag{4}$$

where v_i is the indicator vector for V_i, i.e. $v_i \in \mathbb{R}^V$ with $[v_i]_a = 1$ if $a \in V_i$ and $[v_i]_a = 0$ if $a \notin V_i$. Since, $p\mathcal{L}_G \preceq \frac{3}{2}\mathcal{L}_{G'} + \frac{pd}{n}\mathcal{L}_{K_n}$ we have that for all $i \in [r]$ that

$$p|\partial_G(V_i)| = pv_i^\top \mathcal{L}_G v_i \leq \frac{3}{2}v_i^\top \mathcal{L}_{G'}v_i + \frac{pd}{n} = \frac{3}{2}|\partial_{G'}(V_i)| + \frac{pd}{n}|V_i||V \setminus V_i| \,. \tag{5}$$

Combining (4) and (5) yields that

$$|E_{\text{cut}}| \leq \frac{1}{2}\sum_{i\in[r]}\left(\frac{3}{2p}|\partial_{G'}(V_i)| + d|V_i|\right) \leq \frac{3}{32p}|E'| + dn \leq \frac{3m}{16} + \frac{d_{\min}(G)n}{10k} \leq \frac{|E|}{2}$$

where in the third to last step we used that $\sum_{i\in[r]}|\partial_{G'}(V_i)| \leq |E'|/8$ by Theorem 3.10, in the second to last step we used that $|E'| \leq 2pm$ and that $d = d_{\min}(G)/(10k)$, and in the last step we used that $d_{\min}(G)n \leq |E|$.

Finally, we bound the running time of the algorithm. Every line except for Line 6 in our algorithm is a standard graph operation that can be implemented in linear total time. The call to ExpanderDecomp on Line 6 is on a graph with $O(mp)$ edges. Consequently, by Theorem 3.10 the call takes time

$$O(mp\log^7(m)) = O(m(\log(n)/d)\log^7(n)) = O(nk \cdot d_{\text{ratio}}(G)\log^8(n))$$

where the first equality used the simplicity of G and that $p = \Theta(d^{-1}\log n) \leq 1$ and the second equality used that $m \leq nd_{\max}(G)$ and the definition of d. □

3.3 Approximately Degree Regular Graph Decompositon

Here we provide a linear time procedure to decompose a constant fraction of a dense graph into a nearly-regular dense pieces supported on a bounded number of vertices. We use degree regularity to turn edge expansion bounds on a graph into vertex expansion bounds and finding vertex disjoint paths in Section 3.1 which in turn we use to efficiently compute path sparsifiers.

The main result of this section is the following theorem on computing such a decomposition.

Theorem 3.11 (Regular Decomposition). *Given n-vertex m-edge simple undirected graph $G = (V, E)$ with $d_{\text{avg}}(G) \geq 2000(\log(2n))^2$, RegularDecomp$(G)$ (Algorithm 8) in expected $O(m)$ time outputs graphs $H_1 = (V_1, E_1), ..., H_\ell = (V_\ell, E_\ell)$ which are edge disjoint subsets of G such that*

(1) *(Vertex Size Bound): $\sum_{i \in \ell} |V_i| \leq 4n \log n$.*
(2) *(Volume Lower Bound): $\sum_{i \in \ell} \text{Vol}(E_i) \geq \text{Vol}(G)/100$*
(3) *(Degree Regularity Bound): $d_{\text{ratio}}(H_i) \leq 1000(\log(2n))$ for all $i \in [\ell]$*
(4) *(Minimum Degree Bound): $d_{\min}(H_i) \geq d_{\text{avg}}(G)/(250 \log n)$ for all $i \in [\ell]$.*[5]

We build RegularDecomp and prove Theorem 3.11 in several steps. First we provide DegreeLowerbound (Algorithm 5) which simply removes vertices of degree less than a multiple of the average. It is easy to show (Lemma 3.12) that this procedure runs in linear time, doesn't remove too many edges, and ensures that the minimum degree is a multiple of the average.

Leveraging DegreeLowerbound we provide two procedures, BipartiteSplit (Algorithm 6) and BipartiteDecomp (Algorithm 7) which together, show how to prove a variant of Theorem 3.11 on bipartite graphs where the max degree is not too much larger than the average degree for each side of the bipartition. We show that provided these average degrees are sufficiently large, BipartiteSplit (Algorithm 6) splits the graph into pieces of roughly the same size where the degrees on the larger side are preserved up to a multiplicative factor (See Lemma 3.13). This procedure simply randomly partitions one of the sides and the result follows by Chernoff bound. The procedure BipartiteDecomp (Algorithm 7) then carefully applies BipartiteSplit and DegreeLowerbound.

Our main algorithm, RegularDecomp (Algorithm 8) operates by simply bucketing the vertices into to groups with similar degree and considering the subgraphs of edges that only go between pairs of these buckets. The algorithm then applies either BipartiteDecomp or DegreeLowerbound to subgraphs of sufficiently high volume and analyzing this procedure proves Theorem 3.11.

Algorithm 5: $\{G_i\}_{i \in [k]} = \text{DegreeLowerbound}(G, c)$

Input: Graph $G = (V, E)$, parameter $c \in (0, 1)$
Let $d_{\text{avg}} := 2m/n$, $S = V$, $R = \emptyset$;
while $d_{\min}(G[S]) < c \cdot d_{\text{avg}}$ **do**
  Pick $a \in S$ with $\deg_{G[S]}(a) < c \cdot d_{\text{avg}}$;
  $S := S \setminus \{a\}$;
end
return $G[S]$;

Lemma 3.12 (Degree Lower Bounding)**.** *For any n-node m-edge graph $G = (V, E)$ and $c \in (0, 1)$, DegreeLowerbound(G, c) (Algorithm 5) outputs $G[S]$ for $S \subseteq V$ such that*

$$\text{Vol}(G[S]) \geq (1 - c)\text{Vol}(G) \text{ and } d_{\min}(G[S]) \geq c \cdot d_{\text{avg}}(G)$$

[5]This condition is not use for our path sparsification construction, but is included due to its possible utility.

Proof. This procedure can be implemented in linear time by storing $\deg_{G[S]}(a)$ for all $a \in V$ and updating it in $O(1)$ per edge when a vertex is removed. Further, $d_{\min}(G[S]) \geq cd_{\text{avg}}(G)$ by design of the algorithm. Finally, every time we remove a vertex a from S we remove at most cd_{avg} edges from $G[S]$. Consequently, the final S satisfies $|V \setminus S| \cdot cd_{\text{avg}} \leq n \cdot c \cdot d_{\text{avg}} = 2cm = c \cdot \text{Vol}(G)$ and $\text{Vol}(G[S]) \geq (1-c)\text{Vol}(G)$. □

Algorithm 6: $\{G_i\}_{i\in[k]} = \text{BipartiteSplit}(G, (L, R), k)$

Input: Bipartite graph $G = (V, E)$, bipartition (L, R) of V, $k \in [1, |L|]$

// Assume for all $e = (a, b) \in E$, $a \in L$, $b \in R$

// Assume $\deg_G(b)/k \geq 20\log(2n)$ for all $b \in R$ and $|L|/k \geq 20\log(2n)$

do

  Let $L_i = \emptyset \subseteq L$ for all $i \in [k]$;

  For each $a \in L$ add a to L_i for $i \in [k]$ uniformly, independently at random ;

  Let $G_i = G[L_i \cup R]$ for all $i \in [k]$;

while $\deg_{G_i}(b) \notin [\frac{\deg_G(b)}{2k}, \frac{3\deg_G(b)}{2k}]$ *or* $|L_i| \notin [\frac{|L|}{2k}, \frac{3|L|}{2k}]$ *for some* $b \in R$, $i \in [k]$;

return $\{G_i\}_{i\in[k]}$;

Lemma 3.13 (Bipartite Graph Splitting)**.** *Given n-node m-edge simple bipartite graph $G = (V, E)$ with bipartition into $(L, R) \subseteq V$ and $k \in [1, |L|]$ where*

$$\frac{\deg_G(b)}{k} \geq 20\log(4n) \text{ for all } b \in R \text{ and } \frac{|L|}{k} \geq 20\log(4n)$$

$\text{BipartiteSplit}(G, (L, R), k)$ *(Algorithm 6) in expected $O(m)$ time outputs $\{G_i\}_{i\in[k]}$ that partition the edges such that for all $i \in [k]$ and $b \in R$*

$$\deg_{G_i}(b) \in \left[\frac{\deg_G(b)}{2k}, \frac{3\deg_G(b)}{2k}\right] \text{ and } |L_i| \in \left[\frac{|L||}{2k}, \frac{3|L|}{2k}\right] \tag{6}$$

Proof. Note that each loop of the algorithm clearly takes linear time and if the algorithm terminates its output is as desired. Consequently, it suffices to show that the probability the loop repeats is bounded by some fixed constant probability.

Let $\mu = \min\{\min_{b\in R} \deg_G(b)/k, |L|/k\}$. Further, for each $a \in L$ and $i \in [k]$ let $x_{i,a}$ be a random variable set to 1 if $a \in L_i$ and 0 otherwise. Now note that for all $b \in R$ we have by the fact that $\mathbb{E}[x_{i,a}] = 1/k$ we have

$$\deg_{G_i}(b) = \sum_{a\in N_G(b)} x_{i,a} \text{ and } \mathbb{E}\left[\deg_{G_i}(b)\right] = \frac{\deg_G(b)}{k} \geq \mu \cdot \ .$$

Consequently, by Chernoff bound we have that for all $b \in R$ and $i \in [k]$

$$\Pr\left[\deg_{G_i}(b) \leq \frac{\deg_G(b)}{2k}\right] \leq \exp\left(-\frac{\mu}{8}\right) \text{ and } \Pr\left[\deg_{G_i}(b) \geq \frac{3\deg_G(b)}{2k}\right] \leq \exp\left(-\frac{\mu}{10}\right)$$

Further, for all $i \in [k]$, by the same reasoning (e.g. suppose there was a vertex in R of degree $|L|$),

$$\Pr\left[|L_i| \leq \frac{|L|}{2k}\right] \leq \exp\left(-\frac{\mu}{8}\right) \text{ and } \Pr\left[|L_i| \geq \frac{3|L|}{2k}\right] \leq \exp\left(-\frac{\mu}{10}\right)$$

Now since by assumption $\mu \geq 20\log(4n)$, by applying union bound to the $2(|L|+1)\cdot k$ different reasons the loop in BipartiteSplit might repeat, the loop repeats with probability at most

$$2(|L|+1)\cdot k\cdot \exp(-\mu/10) \leq 2(|L|+1)\cdot k\cdot \frac{1}{8n^2} \leq \frac{1}{4}$$

where we used that since the graph is non-empty $|L|+1 \leq n$. □

Lemma 3.14 (BipartiteDecomp (Algorithm 7))**.** *Let $G=(V,E)$ be an arbitrary simple bipartite graph with bipartition (L,R) such that $d^L_{\text{avg}} \coloneqq \text{Vol}_G(L)/|L|$ and $d^R_{\text{avg}} \coloneqq \text{Vol}_G(R)/|R|$ satisfy $d^L_{\text{avg}} \geq 40\log(2n)$ and $d^R_{\text{avg}} \geq 40\log(2n)$. In expected $O(m)$ time,* BipartiteDecomp$(G,(L,R))$ *outputs graphs $H_1=(V_1,E_1),\ldots,H_k=(V_k,E_k)$ which are edge disjoint subsets of G with*

(1) *(Vertex Size Bound): $\sum_{i\in\ell}|V_i| \leq 4n$.*
(2) *(Volume Lower Bound): $\sum_{i\in\ell}\text{Vol}(E_i) \geq \text{Vol}(G)/8$*
(3) *(Degree Regularity Bound): $d_{\text{ratio}}(H_i) \leq 16c$ for all $i\in[k]$ where*

$$c \coloneqq \max\{d^L_{\max}/d^L_{\text{avg}}, d^R_{\max}/d^R_{\text{avg}}\}$$

for $d^L_{\max} \coloneqq \max_{a\in L}\deg_G(a)$ and $d^R_{\max} \coloneqq \max_{a\in R}\deg_G(a)$.
(4) *(Minimum Degree Bound): $d_{\min}(H_i) \geq \min\{d^L_{\text{avg}}, d^R_{\text{avg}}\}/16$.*

Algorithm 7: $\{H_i\}_{i\in[k]} =$ BipartiteDecomp$(G,(L,R))$

Input: Bipartite graph $G=(V,E)$ and bipartition (L,R) of V
// Assume $\text{Vol}(L)/|L| \geq 40\log(2n)$ and $\text{Vol}(R)/|R| \geq 40\log(2n)$
1 Let $d^L_{\text{avg}} \coloneqq \text{Vol}(L)/|L|$ and $d^R_{\text{avg}} \coloneqq \text{Vol}(R)/|R|$;
2 Swap L and R if needed so that $|R| \leq |L|$;
3 Let $R' = \{a\in R \mid \deg_G(a) \geq (1/2)d^R_{\text{avg}}\}$ and $G' = G(L\cup R')$;
4 **if** $|R'| \geq |L|/2$ **then return** DegreeLowerbound$(G,1/2)$;
5 $\{G_i\}_{i\in[k]} =$ BipartiteSplit$(G',(L,R'),k)$ for $k=\lfloor |L|/|R'|\rfloor$;
6 **return** $\{H_i\}_{i\in[k]}$ where $H_i =$ DegreeLowerbound$(G_i,1/2)$;

Proof. First, note that G' is a vertex induced subgraph of G where only vertices in R with degree less than half the average in R are removed. Since $\sum_{a\in R\setminus R'}\deg_G(A) \leq |R|d^R_{\text{avg}}/2 \leq \text{Vol}_G(R)$ it follows that $\text{Vol}_{G'}(R') \geq \frac{1}{2}\text{Vol}_G(R)$. Further, since G is bipartite this implies that $\text{Vol}(G') \geq \text{Vol}(G)/2$.

Next, suppose the algorithm returns on Line 4 (i.e. $|R| < |L|/2$). In this case, Lemma 3.12 (which analyzes DegreeLowerbound) implies that the returned graph, which we denote H, is a vertex induced subgraph of G' with $\text{Vol}(H) \geq \text{Vol}(G')/2 \geq \text{Vol}(G)/4$ and $d_{\min}(H) \geq d_{\text{avg}}(G')/2 \geq d_{\text{avg}}(G)/4$. Since $d_{\text{avg}}(G) \geq \min\{d^L_{\text{avg}}, d^R_{\text{avg}}\}$, this immediately yields the desired vertex size bound, vertex lower bound, and the minimum degree bound. Further, since the graph is bipartite we have $d^L_{\text{avg}}/d^R_{\text{avg}} = |R|/|L|$. Therefore, since $|L|/2 \leq |R| \leq L$ we have $d^R_{\text{avg}}/2 \leq d^L_{\text{avg}} \leq d^R_{\text{avg}}$ and $d_{\text{avg}}(G) \geq$

$d^L_{\text{avg}} \geq d^R_{\text{avg}}/2$. Consequently

$$d_{\text{ratio}}(H) \leq \frac{d_{\max}(G)}{d_{\text{avg}}(G)/4} \leq 4 \cdot \max\left\{\frac{d^L_{\max}}{d_{\text{avg}}(G)}, \frac{d^R_{\max}}{d_{\text{avg}}(G)}\right\} \leq 4 \max\left\{\frac{d^L_{\max}}{d^L_{\text{avg}}}, \frac{d^R_{\max}}{(1/2)d^R_{\text{avg}}}\right\} \leq 8c$$

and the result holds in this case.

Therefore, in the remainder of the proof we assume instead that $|R'| \leq |L|/2$. Further, since G is bipartite we know that $d^L_{\text{avg}} \geq 2d^R_{\text{avg}}$. Note that this implies that $|L|/|R| \geq 2$ and therefore $k \in [|L|/(2|R'|), |L|/|R'|]$. Note that the average degree of a vertex in L in G' is at least $d^L_{\text{avg}}/2$, by our reasoning regarding G' and consequently, since the graph is simple $|R'| \geq d^L_{\text{avg}}/2$. This implies

$$\frac{|L|}{k} \geq |R'| \geq \frac{|R|}{2} \geq \frac{d^L_{\text{avg}}}{2} \geq 20 \log(2n)$$

where the last inequality follows from the assumption on the input. Further, by design we have that for all $b \in R'$

$$\deg_{G'}(b) \geq d^R_{\text{avg}}/2 \geq 20 \log(2n) \ .$$

Consequently, Lemma 3.13 applies to $\text{BipartiteSplit}(G', (L, R'), k)$ and for all $i \in [k]$ and $b \in R'$

$$\deg_{G_i}(b) \in \left[\frac{\deg_{G'}(b)}{2k}, \frac{3\deg_{G'}(b)}{2k}\right] \text{ and } |L_i| \in \left[\frac{|L|}{2k}, \frac{3|L|}{2k}\right] . \tag{7}$$

Now let $d^{L'}_{\text{avg}} \coloneqq \text{Vol}_{G'(L)}/|L|$ and $d^{R'}_{\text{avg}} \coloneqq \text{Vol}_{G'}(R')/|R'|$. Note that $|L_i| \leq 3|L|/(2k) \leq 3|R'|$ and $k \leq |L|/|R'| = d^{R'}_{\text{avg}}/d^{L'}_{\text{avg}}$. This implies

$$d_{\text{avg}}(G_i) = \frac{2}{|R'| + |L_i|} \sum_{b \in R'} \deg_{G_i}(b) \geq \frac{1}{2|R'|} \sum_{b \in R'} \frac{\deg_{G'}(b)}{2k} \geq \frac{d^{R'}_{\text{avg}}}{4k} \ .$$

Since the average degree of a vertex in R' in G' is at least the average degree of a vertex in R in G, we have $d^{R'}_{\text{avg}} \geq d^R_{\text{avg}}$. Further, we have $d^L_{\text{avg}} \leq d^{L'}_{\text{avg}} \leq d^{R'}_{\text{avg}}/k$ by the construction of G'. This implies

$$d_{\max}(G_i) \leq \max\{(3/(2k))d^R_{\max}, d^L_{\max}\} \leq c \cdot \max\{(3/(2k))d^{R'}_{\text{avg}}, d^{R'}_{\text{avg}}/k\} \leq 2c \cdot d^{R'}_{\text{avg}}/k$$

and $d_{\text{ratio}}(G_i) \leq 8c$. Further, since $d^{R'}_{\text{avg}} \geq d^{L'}_{\text{avg}}k$ and $d^{L'}_{\text{avg}} \geq d^L_{\text{avg}}/2$ by the construction of G, we have that $d_{\text{avg}}(G_i) \geq d^L_{\text{avg}}/8$. Consequently, the volume lower bound, degree regularity bound, and minimum degree bound follow from the fact that invoking $\text{DegreeLowerbound}(G_i, 1/2)$ only removes edges and decreases the volume by at most a factor of 2 and decreases the min degree to at most $(1/2)$ the average by Lemma 3.12. Finally, the vertex size bound followed from the bound on k and that the only vertices repeated are R' which are repeated at most k times. □

We now have everything we need to present RegularDecomp (Algorithm 8), our graph decomposition algorithm, and analyze it to prove Theorem 3.11, the main result of this section.

Algorithm 8: $\{G_i\}_{i\in[k]} = \text{RegularDecomposition}(G)$

Input: Undirected, unweighted, connected graph $G = (V, E)$ with n-vertices and m-edges

$G' = (V', E')$ for $G' := \text{DegreeLowerbound}(G, 1/2)$;
Let[6] $S_i := \{a \in V' \mid \deg'_G(a) \in [e^{i-1}, e^i)\}$ for all $i \in [1, k]$ where $k := \lfloor \log(n) \rfloor$;
$\mathcal{H}_{out} := \emptyset$;
for $i, j \in [k]$ *with* $i \le j$ *(including* $i = j$*)* **do**
  Let $V_{i,j} := S_i \cup S_j$ and $E_{i,j} := \{\{a, b\} \in E' \mid a \in S_i, b \in S_j\}$;
  Let $G_{i,j} := (S_i \cup S_j, E_{i,j})$;
  if $\text{Vol}(G_{i,j}) \ge \text{Vol}_{G'}(S_i)/(2\log n)$ *and* $\text{Vol}(G_{i,j}) \ge \text{Vol}_{G'}(S_j)/(2\log(n))$ **then**
    if $i = j$ **then** $\mathcal{H}_{out} := \mathcal{H}_{out} \cup \{\text{DegreeLowerbound}(G_{i,j}), 1/2\}$;
    else $\mathcal{H}_{out} := \mathcal{H}_{out} \cup \text{BipartiteDecomp}(G_{i,j}, (S_i, S_j))$;
  end
end
return all output graphs computed

Proof of Theorem 3.11. First we show that whenever BipartiteDecomp(G) is invoked by the algorithm in Line 9 on $G_{i,j}$ then Lemma 3.14 applies with $c \le 4e \log n$, $d^L_{\text{avg}} \ge 40\log(2n)$, $d^R_{\text{avg}} \ge 40\log(2n)$. Fix an invocation of BipartiteDecomp(G) on Line 9 for $i \ne j$ and let $L = S_i$, $R = S_j$, and $d^L_{\text{avg}} := \text{Vol}_{G_{i,j}}(S_i)/|S_i|$ and $d^R_{\text{avg}} := \text{Vol}_{G_{i,j}}(S_j)/|S_j|$. By design, for all $a \in S_i$ and $b \in S_j$

$$\deg_{G'}(a) \in [e^{i-1}, e^i] \text{ and } \deg_{G'}(b) \in [e^{j-1}, e^j] .$$

Therefore, by the guarantees of Lemma 3.12 for $G' := \text{DegreeLowerbound}(G, 1/2)$

$$e^{i-1} \ge e^{-1} d_{\min}(G') \ge (1/(2e)) d_{\text{avg}}(G) \ge (1000/e)(\log(2n))^2 .$$

Further, since $\text{Vol}(G_{i,j}) \ge \text{Vol}_{G'}(S_i)/(2\log n)$ and $\text{Vol}_{G'}(S_i) \ge |S_i| e^{i-1}/2$ by the degree bounds of $a \in S_i$ we see that

$$d^L_{\text{avg}} := \frac{\text{Vol}_{G_{i,j}}(S_i)}{|S_i|} \ge \frac{\text{Vol}_{G'}(S_i)}{|S_i| \cdot 2\log n} \ge \frac{e^{i-1}}{4\log n} \ge \frac{1000\log(2n)^2}{4e\log n} \ge 40\log(2n) .$$

By the same reasoning $d^R_{\text{avg}} \ge 40\log(2n)$. To bound c, note that deleting edges can only decrease degree and therefore

$$\max_{a\in L} \frac{\deg_{G_{i,j}}(a)}{d^L_{\text{avg}}} \le \max_{a\in L} \frac{\deg_{G'}(a)}{(e^{i-1}/(4\log(n)))} \le \frac{e^i}{(e^{i-1}/(4 \cdot \log n))} = 4e\log n .$$

Since by symmetry the same bound holds for R, the desired bound for c holds.

(Vertex Bound): Note that a vertex can appear in at most k different $G_{i,j}$. Consequently, the result follows by Lemma 3.12 and Lemma 3.14.

First note that by Lemma 3.12 we have $\text{Vol}(G') \ge (1/2)\text{Vol}(G)$ and $d_{\min}(G') \ge (1/2) d_{\text{avg}}(G)$.

(Volume Bound): Note that by Lemma 3.12 we have $\mathrm{Vol}(G') \geq (1/2)\mathrm{Vol}(G)$. Further, define the set $P := \{(i,j) \in [k] \times [k] \mid i \leq j\}$ and let

$$P_{\geq} := \{(i,j) \in P \mid \mathrm{Vol}(G_{i,j}) \geq \mathrm{Vol}_{G'}(S_i)/(2\log n) \text{ and } \mathrm{Vol}(G_{i,j}) \geq \mathrm{Vol}_{G'}(S_j)/(2\log n)\}\,.$$

Note that P contains the indices for every $G_{i,j}$ considered and that $P_{\geq}$ denotes the subset of them for which the volume of $G_{i,j}$ is large enough that Line 7 is true. By design,

$$\begin{aligned}\sum_{(i,j)\in P\setminus P_{\geq}} \mathrm{Vol}(G_{i,j}) &< \sum_{(i,j)\in P}\left(\max\left\{\frac{\mathrm{Vol}(G'(S_j))}{2\log n}, \frac{\mathrm{Vol}(G'(S_i))}{2\log n}\right\}\right)\\ &\leq \frac{k}{2\log n}\sum_{i\in[k]}\mathrm{Vol}(G'(S_j)) \leq \frac{1}{2}\mathrm{Vol}(G')\,.\end{aligned}$$

Since every edge $e \in E'$ is in some $G_{i,j}$ this then implies that

$$\sum_{(i,j)\in P_{\geq}}\mathrm{Vol}(G_{i,j}) = \sum_{(i,j)\in P}\mathrm{Vol}(G_{i,j}) - \sum_{(i,j)\in P\setminus P_{\geq}}\mathrm{Vol}(G_{i,j}) \geq \mathrm{Vol}(G') - \frac{1}{2}\mathrm{Vol}(G') \geq \frac{1}{4}\mathrm{Vol}(G)\,.$$

Since our output is simply the result of invoking DegreeLowerbound and BipartiteDecomp on these graphs and by Lemma 3.12 and Lemma 3.14 and these procedures decrease the volume by at most a factor of 8, the result follows.

(Degree Regularity Bound): For graph $G_{i,j}$ with $i \neq j$ this follows from the bound on $c \leq 4e\log n$ given in the first paragraph of this proof, Lemma 3.14, and that $16 \cdot 4e\log n \leq 1000\log(2n)$. For graph $G_{i,j}$ with $i = j$, the same reasoning implies the ratio of the maximum degree to the average degree is at most c and the result follows by Lemma 3.12, which shows that DegreeLowerbound$(G_{i,j}, 1/2)$ only decreases the maximum degree and makes the minimum degree is at least half the average degree.

(Minimum Degree Bound): By the reasoning of the first paragraph of this section we know that whenever BipartiteDecomp(G) is invoked by the algorithm in Line 9 on $G_{i,j}$ then

$$d^L_{\mathrm{avg}} \geq \frac{\mathrm{Vol}_{G'}(S_i)}{|S_i|\cdot 2\log n} \geq \frac{d_{\min}(G')}{4\log n} \geq \frac{d_{\mathrm{avg}}(G)}{8\log n}$$

where in the last step we used that the average degree in G' is at least the average degree in G by Lemma 3.12. Consequently, the result follows again by Lemma 3.12, Lemma 3.14, and the fact that $16 \cdot 8 \leq 250$. □

3.4 Putting it All Together

Here we show how to put together all the results of the previous subsection to prove Theorem 1.9. Our algorithm, PathSparsify (Algorithm 9) simply performs a regular decomposition of the input graph by Algorithm 8 (Theorem 3.11) of Section 3.3 and then performs of partial path sparsification of each of these graphs by Algorithm 4 (Theorem 3.8) of Section 3.2. In the remainder of this section we provide and analyze PathSparsify (Algorithm 9) to prove Theorem 1.9 (restated below for convenience).

Theorem 1.9 (Efficient Path Sparsification). *Given any n-node, m-edge graph and parameter $k \geq 1$ the procedure,* PathSparsify(G,k) *(Algorithm 9) outputs w.h.p. $F \subseteq E$ with $|F| = O(nk\log^3(n))$ such that $G[F]$ is a $(k, O(\log^5 n))$-path-sparsifier of G in expected time $O(m + nk\log^{13}(n))$.*

Algorithm 9: $F = \text{PathSparsify}(G, k \geq 1)$

Input: $G = (V, E)$ simple input graph with degrees between $d_{\min}$ and $d_{\max}$
Output: $F \subseteq E$ with $|F| = O(nk \log^3(n))$ such that $G[F]$ is a $(k, O(\log^5 n))$-path-sparsifier of G

1 $k_{\text{partial}} = \Theta(\log^3 n)$; // For constants in k_{partial} see Theorem 1.9 proof
2 $E_{\text{remain}} \leftarrow E$, $F \leftarrow \emptyset$;
3 **while** $d_{\text{avg}}(G(E_{\text{remain}})) \geq 2000(\log(2n))^2$ **do**
4 $\quad \{H_i = (V_i, E_i)\}_{i \in [\ell]} \leftarrow \text{RegularDecomposition}(G)$; // Algorithm 8 (Theorem 3.11)
5 $\quad$ **for** $i \in [\ell]$ **do**
6 $\quad\quad (F^{(i)}, E^{(i)}_{\text{cut}}) \leftarrow \text{PartialPathSparsify}(H_i, k_{\text{partial}})$; // Algorithm 4 (Theorem 3.8)
7 $\quad$ **end**
8 $\quad F \leftarrow \cup_{i \in [\ell]} F^{(i)}$ and $E_{\text{remain}} \leftarrow \cup_{i \in [\ell]} E^{(i)}_{\text{cut}}$;
9 **end**
10 $F \leftarrow F \cup E_{\text{remain}}$;
11 **return** F;

Proof of Theorem 1.9. We first consider the execution of a single loop of Algorithm 9, i.e. Line 3 to Line 9. Since $d_{\text{avg}}(G(E_{\text{remain}})) \geq 2000(\log(2n))^2$ we can apply Lemma 3.12 to analyze the execution of RegularDecomp. Lemma 3.12 implies that this line takes expected $O(|E_{\text{remain}}|)$ time and outputs $\{H_i = (V_i, E_i)\}_{i \in [\ell]}$ such that at least constant fraction of the edges of E_{remain} are in the E_i, $\sum_{i \in \ell} |V(H_i)| = O(n \log^2 n)$, $d_{\text{ratio}}(H_i)) = \Omega(\log(2n))$, and $d_{\min}(H_i) = \Omega(d_{\text{avg}}(G)/\log n)$ for all $i \in [\ell]$ (where we used that the number of vertices in $|E_{\text{remain}}|$ is at most n). Consequently, in a single execution of the while loop, Theorem 3.8 shows that Line 6 takes time

$$O\left(\sum_{i \in [\ell]} |E_i| + |V_i| k_{\text{partial}} d_{\text{ratio}}(H_i) \log^8(n)\right) = O\left(|E_{\text{remain}}| + nk_{\text{partial}} \log^9(n)\right) \quad (8)$$

and w.h.p. in n outputs $\{(F^{(i)}, E^{(i)}_{\text{cut}})\}_{i \in [\ell]}$ such that

$$\sum_{i \in [\ell]} |F^{(i)}| = O\left(\sum_{i \in [\ell]} |V_i| k_{\text{partial}} \cdot d_{\text{ratio}}(H_i) \log(n)\right) = O\left(nk_{\text{partial}} \log^2(n)\right)$$

each $H_i(F)$ is a $(\Omega(k_{\text{partial}}/\log^3(n)), O(\log^5 n))$-path sparsifier of $(V_i, E_i \setminus E^{(i)}_{\text{cut}})$ and $\sum_{i \in [\ell]} |E^{(i)}_{\text{cut}}| \leq c|E_{\text{remain}}|$ for some constant $c \in (0, 1)$.

The preceding paragraph ultimately shows that w.h.p. each iteration of the loop, i.e. Line 3 to Line 9, takes expected time $O(|E_{\text{remain}}| + nk_{\text{partial}} \log^9(n))$ to output a $(\Omega(k_{\text{partial}}/\log^3(n)), O(\log^5 n))$-path sparsifier on a constant fraction of the edges. Consequently, the loop terminates in $O(\log |E|) = O(\log n)$ iterations. Note that when the loop terminates $d_{\text{avg}}(G(E_{\text{remain}})) < 2000(\log(2n))^2$ and therefore $|E_{\text{remain}}| = O(n \log^2(n)) = O(nk_{\text{partial}} \log^2(n))$. Consequently, F returned by the algorithm has size

Algorithm 10: $\mathbf{Z} = \text{Sample}(\{\mathbf{Y}_1, ..., \mathbf{Y}_m\}, \mathbf{X}, \tau, \delta)$ (from [CKM+])

Input: $\mathbf{Y}_i = v_i v_i^\top$ are rank one matrices, τ_i are upper bounds of leverage scores, i.e. $\tau_i \geq \text{tr}[\mathbf{Y}_i \mathbf{X}^\dagger]$ for all i, and $\delta < 1$ is an arbitrary parameter.

Output: Matrix $\mathbf{X}$ satisfying conditions of Lemma 4.1.

$\mathbf{Z} \leftarrow \mathbf{X}$, $s = \sum_{i \in [m]} \tau_i$, $t = \delta^{-1} s$

$r \leftarrow$ randomly chosen integer in $[t, 2t-1]$

for $j = 1, 2 \ldots r$ **do**

  Pick index i with probability proportional to τ_i

  $\mathbf{Z} \leftarrow \mathbf{Z} + \frac{\delta}{\tau_i} \mathbf{Y}_i$

end

return $\mathbf{Z}$

at most $O(nk_{\text{partial}} \log^3 n)$. Further, note that if for any edge-disjoint graphs $\bar{G}_i = (V, \bar{E}_i)$ for $i \in [k]$ and $\bar{F}_i \subseteq \bar{E}_i$ it is the case that each $\bar{G}[\bar{F}_i]$ is an (α, β)-path sparsifier of $\bar{G}_i$ then $\bar{G}[\cup_{i \in [k]} \bar{F}_i]$ is an (α, β)-path sparsifier of $\bar{G} = (V, \cup_{i \in [k]} \bar{E}_i)$. Consequently, $G[F]$ is an $(\Omega(k_{\text{partial}}/\log^3(n)), O(\log^5 n))$-path-sparsifier of G. By choice of constants in the setting of $k_{\text{partial}} = \Theta(k \log^3 n)$ we have that the output of the algorithm is as desired. Further, the run time follows from the reasoning in the preceding paragraph about the runtime of a single loop (i.e. (8)) and that the number of edges in E_{remain} decrease by a constant in each iteration.

□

4 LAPLACIAN SOLVERS WITH LOW-STRETCH SUBGRAPHS

In this section, we prove our main theorem regarding algorithms with improved running times for solving Laplacian linear systems. We show how to use the low distortion spectral subgraphs developed in Section 2.2 to prove the following theorem.

Theorem 1.6 ($\tilde{O}(m)$-Laplacian System Solver). *There is a randomized algorithm which is an ϵ-approximate Laplacian system solver for any input n-vertex m-edge graph with polynomially-bounded edge weights (see Definition 1.2) and $\epsilon \in (0, 1)$ and has the following runtime for any $\chi > 0$*

$$O(m(\log \log n)^{6+2\sqrt{10}+\chi} \log(1/\epsilon)) .$$

Our proof is based on an analogous claim in [CKM+] regarding different spectral subgraph guarantees. Several proofs in this section are adaptations of lemmas from [CKM+] to our setting. We provide the proof in full here both for completeness and because the specific guarantees of [CKM+] do not tolerate the extra edges in our preconditioners coming from our path sparsifiers. In addition our analysis based on noisy accelerated gradient descent is slightly tighter than that of [CKM+], enabling us to obtain an improved guarantee.

A key matrix fact we apply in this section is a slight extension of a claim from [CKM+] regarding Sample (Algorithm 10), a matrix sampling procedure from [CKM+].

Lemma 4.1 (Adaptation of Lemma 2.3 from [CKP+14]). *Suppose $\mathbf{X}$ and $\mathbf{Y} = \sum_{i \in [m]} \boldsymbol{Y}_i$ are symmetric matrices with the same null space such that $\mathbf{X} \preceq \mathbf{Y}$, $b = \mathbf{Y}\bar{x}$, and x is*

an arbitrary vector. Let the $\mathbf{Y}_i$ *matrices be rank-one, and let* $\tau \in \mathbb{R}^E_{\geq 0}$ *be leverage score overestimates in that they satisfy* $\tau_i \geq \mathrm{tr}[\mathbf{Y}_i \mathbf{X}^\dagger]$. *Let* $\mathbf{Z} = \mathsf{Sample}(\{\mathbf{Y}_1, \ldots \mathbf{Y}_m\}, \mathbf{X}, \tau, \frac{1}{10})$, *and define* x' *as*

$$x' = x - \frac{1}{10}\mathbf{Z}^\dagger(\mathbf{Y}x - b).$$

Then

$$\mathbb{E}_{r, i_1, i_2, \ldots i_r}\left[\|x' - \bar{x}\|_{\mathbf{Y}}^2\right] \leq \left(1 - \frac{1}{40}\right)\|x - \bar{x}\|_{\mathbf{Y}}^2. \tag{9}$$

Further, $\mathbf{Z}$ *can be computed in* $O(m + \|\tau\|_1)$ *time, and each matrix* $\mathbf{Y}_i$ *is added at least once to* $\mathbf{Z}$ *with probability at most* $\min\{1, 20\tau_i\}$. *Finally, for some fixed constant* c_s *with high probability in* n *we have*

$$\frac{1}{c_s \log n}\mathbf{Y} \preceq \mathbf{Z} \preceq c_s \log n \mathbf{Y}.$$

Proof. Equation 9 and the bound on the algorithm's runtime are directly copied from Lemma 2.3 from [CKP+14]. For the bound on the probability $\mathbf{Y}_i$ is added to $\mathbf{Z}$, we look at each execution of line 5 of the algorithm. For iteration j, we pick $\mathbf{Y}_i$ with probability $\frac{\tau_i}{s}$. Thus, we pick $\mathbf{Y}_i$ at most $r\frac{\tau_i}{s} \leq \frac{2s}{\delta}\frac{\tau_i}{s} = 20\tau_i$ times in expectation. The conclusion follows by Markov's inequality. The final claim follows immediately from Lemma C.2 from [CKP+14]. □

We additionally employ a partial Cholesky factorization lemma from [CKM+] which enables us to reduce solving ultrasparse graph Laplacians to solving Laplacians with a much smaller number of edges.

Lemma 4.2. *Let* G *be a weighted graph on* n *vertices and* $n + m'$ *edges. There is a routine* $\mathsf{EliminateAndSolve}(G, \mathsf{Solve}, b)$ *which computes a vector* x *satisfying*

$$\left\|x - \mathcal{L}_G^\dagger b\right\|_{\mathcal{L}_G}^2 \leq \epsilon \left\|\mathcal{L}_G^\dagger b\right\|_{\mathcal{L}_G}^2$$

using $O(n + m')$ *time plus one call to* Solve, *which is an* ϵ*-approximate solve for graphs with at most* $O(m')$ *nodes and edges.*

Variants of this result are used in many prior Laplacian system solvers, e.g. [ST, KMP, KMP14]. For a detailed proof of this lemma with floating-point error analysis see Appendix C of [Pen13].

We now assemble these pieces to give an algorithm for solving Laplacians in graphs which contain ultrasparse low-stretch subgraphs:

Lemma 4.3. *Let* G *be a weighted* n*-node* m*-edge graph and let* G' *be a subgraph of* G *with* $n + m'$ *edges. Let* $\tau_1, \ldots \tau_m \mathbb{R}^E$ *be values satisfying* $\tau_e \geq w_e \mathcal{R}_{G'}^{eff}(u, v)$ *for any* $e = (u, v)$. *Let* b *be a vector, and let* $\bar{x} = \mathcal{L}_G^\dagger b$. *Then if* Solve *is a* $(1600 c_s^2 \log^2 n)^{-1}$*-Laplacian solver (Definition 1.2)', Algorithm* $\mathsf{PreconRichardson}(G, G', \tau, b, \epsilon)$ *computes a vector* x *satisfying*

$$\mathbb{E}\left[\|x - \bar{x}\|_{\mathcal{L}_G}^2\right] \leq \epsilon \|\bar{x}\|_{\mathcal{L}_G}^2$$

using $O(\log 1/\epsilon)$ *iterations. Each iteration consists of* $O(m + \|\tau\|_1)$ *work plus one call to* Solve *on a graph with* $O(\|\tau\|_1 + m')$ *edges.*

Algorithm 11: $\mathbf{Z}$ = PreconRichardson(G, G', τ, b, ϵ, Solve)

Input: G graph, G' subgraph, τ_i are upper bounds of leverage scores through G', b vector, ϵ error tolerance.

Output: x approximately satisfying $\mathcal{L}_G x = b$.

1 $x_1 \leftarrow 0$
2 **for** $i \leftarrow 1$ **to** $200 \log 1/\epsilon$ **do**
3 $H_i \leftarrow \text{Sample}(G, G', \tau, \frac{1}{10})$
4 **if** $|E(H_i)| \leq 1600 \|\tau\|_1 + |E(G')|$ **then**
5 $r_i \leftarrow \mathcal{L}_G x_i - b$
6 $y_i \leftarrow \text{EliminateAndSolve}(H_i, \text{Solve}, r_i)$
7 $x_{i+1} \leftarrow x_i - \frac{1}{10} y_i$
8 **end**
9 **else**
10 $x_{i+1} \leftarrow x_i$
11 **end**
12 **end**
13 **return** x_i

Proof. We bound the expected decrease of $\|x_i - \bar{x}\|^2_{\mathcal{L}_G}$ in an iteration. We first show the condition on Line 4 holds with large probability. By Lemma 4.1 the expected number of edges added to G' when forming H is at most $20 \|\tau\|_1$. Thus by Markov's inequality H contains fewer than $|E(G')| + 1600 \|\tau\|_1$ edges with probability at least $1 - \frac{1}{80}$. If we call this event ρ, by Markov's inequality we have

$$\mathbb{E}\left[\left\|\bar{x} - \left(x_i - \frac{1}{10}\mathcal{L}_{H_i}^{\dagger}(\mathcal{L}_G x_i - b)\right)\right\|^2_{\mathcal{L}_G} \Big| \rho\right] \leq \left(1 - \frac{1}{80}\right) \|\bar{x} - x_i\|^2_{\mathcal{L}_G}$$

where the expectation is over the randomness within a single iteration of the while loop. Now by the guarantees of Solve and Lemma 4.1 we know that in each iteration with high probability

$$\begin{aligned}
\mathbb{E}\left[\left\|y_i - \mathcal{L}_{H_i}^{\dagger} r_i\right\|^2_{\mathcal{L}_G}\right] &\leq (c_s \log n)\mathbb{E}\left[\left\|y_i - \mathcal{L}_{H_i}^{\dagger} r_i\right\|^2_{\mathcal{L}_{H_i}}\right] \leq \frac{c_s \log n}{1600 c_s^2 \log^2 n}\left\|\mathcal{L}_{H_i}^{\dagger} r_i\right\|^2_{\mathcal{L}_{H_i}} \\
&\leq \frac{1}{1600 c_s \log n}\|\mathcal{L}_G(\bar{x} - x_i)\|^2_{\mathcal{L}_{H_i}^{\dagger}} \leq \frac{c_s \log n}{1600 c_s \log n}\|\mathcal{L}_G(\bar{x} - x_i)\|^2_{\mathcal{L}_G^{\dagger}} \\
&= \frac{1}{1600}\|\bar{x} - x_i\|^2_{\mathcal{L}_G}.
\end{aligned}$$

Now, note that for any two vectors u, v and any Euclidean norm we have

$$\|u + v\|^2 = \|u\|^2 + \|v\|^2 + 2u^{\top} v \leq (1 + \alpha)\|u\|^2 + (1 + \alpha^{-1})\|v\|^2$$

for any $\alpha > 0$ by the Cauchy-Schwarz inequality and the AM-GM inequality. With this, we obtain for any $\alpha > 0$

$$\begin{aligned}\|\bar{x} - x_{i+1}\|_{\mathcal{L}_G}^2 &= \left\|\bar{x} - \left(x_i - \frac{1}{10}y_i + \frac{1}{10}\mathcal{L}_{H_i}^{\dagger} r_i - \frac{1}{10}\mathcal{L}_{H_i}^{\dagger} r_i\right)\right\|_{\mathcal{L}_G}^2 \\ &\leq (1+\alpha)\left\|\bar{x} - \left(x_i - \frac{1}{10}\mathcal{L}_{H_i}^{\dagger} r_i\right)\right\|_{\mathcal{L}_G}^2 + \frac{1}{100}\left(1+\alpha^{-1}\right)\left\|y_i - \mathcal{L}_{H_i}^{\dagger} r_i\right\|_{\mathcal{L}_G}^2.\end{aligned}$$

Choosing $\alpha = \frac{1}{400}$, we thus obtain

$$\begin{aligned}\mathbb{E}\left[\|\bar{x} - x_{i+1}\|_{\mathcal{L}_G}^2 \,|\rho\right] &\leq \left(1 + \frac{1}{400}\right)\left(1 - \frac{1}{80}\right)\|\bar{x} - x_i\|_{\mathcal{L}_G}^2 + \frac{1}{100}\left(\frac{401}{1600}\right)\|\bar{x} - x_i\|_{\mathcal{L}_G}^2 \\ &\leq \left(1 - \frac{1}{160}\right)\|\bar{x} - x_i\|_{\mathcal{L}_G}^2.\end{aligned}$$

Therefore we obtain

$$\begin{aligned}\mathbb{E}\left[\|\bar{x} - x_{i+1}\|_{\mathcal{L}_G}^2\right] &\leq \Pr(\rho)\mathbb{E}\left[\|\bar{x} - x_{i+1}\|_{\mathcal{L}_G}^2 \,|\rho\right] + Pr(\neg\rho)\mathbb{E}\left[\|\bar{x} - x_{i+1}\|_{\mathcal{L}_G}^2 \,|\neg\rho\right] \\ &\leq \left(1 - \frac{1}{80}\right)\left(1 - \frac{1}{160}\right)\|\bar{x} - x_i\|_{\mathcal{L}_G}^2 + \frac{1}{80}\|\bar{x} - x_i\|_{\mathcal{L}_G}^2 \\ &\leq \left(1 - \frac{1}{200}\right)\|\bar{x} - x_i\|_{\mathcal{L}_G}^2.\end{aligned}$$

As we perform $200\log 1/\epsilon$ iterations, this error decrease guarantee implies the output x satisfies the desired bound of

$$\mathbb{E}\left[\|\bar{x} - x\|_{\mathcal{L}_G}^2\right] \leq \left(1 - \frac{1}{200}\right)^{200\log 1/\epsilon}\|\bar{x} - x_1\|_{\mathcal{L}_G}^2 \leq \epsilon\|\bar{x}\|_{\mathcal{L}_G}^2 = \epsilon\left\|\mathcal{L}_G^{\dagger} b\right\|_{\mathcal{L}_G}^2.$$

We now bound the runtime per iteration. In each iteration we make 1 call to Sample, which by Lemma 4.1 requires $O(m + \|\tau\|_1)$ time. Now if the sampled subgraph H_i does not satisfy the condition on line 4 of the algorithm, we conclude the iteration. If we instead have $|E(H_i)| \leq 3200\|\tau\|_1 + |E(G')|$, the call to EliminateAndSolve on line 6 requires $O(m)$ work plus a single call to Solve on a graph with $O(\|\tau\|_1 + m)$ edges. The claim follows. □

Finally, we apply this primitive recursively to precondition an accelerated gradient descent algorithm with guarantees given by the following theorem.

Theorem 4.4 (Randomized Preconditioned AGD). *Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ be symmetric PSD matrices with $\boldsymbol{A} \preceq \boldsymbol{B} \preceq \kappa\boldsymbol{A}$ for $\kappa \geq 1$, let $b \in \mathrm{im}(\boldsymbol{A})$, let $\epsilon \in (0,1)$ and let $\mathrm{Solve}_{\boldsymbol{B}}$ be a $\frac{1}{10\kappa}$-approximate solver for $\boldsymbol{B}$. Then $\mathrm{PreconditionedNoisyAGD}(\boldsymbol{A}, b, \epsilon, \kappa, \mathrm{Solve}_{\boldsymbol{B}})$ (Algorithm 13) is an ϵ-approximate solver for $\boldsymbol{A}$ and its runtime is the runtime of $O(\sqrt{\kappa}\log(1/\epsilon))$ iterations each of which consist of applying $\boldsymbol{A}$ to a vector, invoking $\mathrm{Solve}_{\boldsymbol{B}}$, and additional $O(n)$ time operations.*

While related theorems are standard to the literature and a deterministic variant analyzing Chebyshev iteration appears in [CKM+], we provide this theorem both for completeness and to simplify and improve our analysis. The theorem is discused in greater detail and proved in Appendix C. With this, we have the pieces to give our final algorithm for solving Laplacian linear systems:

Algorithm 12: $\mathbf{Z}$ = RecursiveSolver$(G, b, \epsilon, \delta,$ LowStretch$)$

Input: G graph,b vector, LowStretch oracle that returns low-stretch subgraphs, $\epsilon \in (0, 1/2]$ error tolerance, $\delta \in (0, 1)$

Output: x approximately satisfying $\mathcal{L}_G x = b$.

1 Let z denote the solution of $f(z) + 2 + \delta = z$, where f is defined in Theorem 4.5
2 $\gamma = C(\log\log n)^z$, for sufficiently large constant C
3 $\{H, \tau\} \leftarrow$ LowStretch(G, z)
4 $\kappa \leftarrow$ an overestimate of $\|\tau\|_1$ output by LowStretch, $\eta \leftarrow \frac{\gamma\kappa}{m}$
5 $G' = G + (\eta - 1)H$, $\tau' \leftarrow \frac{\tau}{\eta}$
6 RecSolver $\leftarrow$ RecursiveSolver$(\cdot, \cdot, \frac{1}{1600c_s^2 \log^2 n}$, LowStretch$)$
7 RichardsonSolver$_{G'}$ = PreconRichardson$(G', \eta H, \tau', \cdot, \frac{1}{10\eta}$, RecSolver$)$
8 $x \leftarrow$ PreconNoisyAGD$(G, b, \epsilon, \eta,$ RichardsonSolver$_{G'})$
9 **return** x

Theorem 4.5. *Let $G = (V, E, w)$ be a n-node m-edge graph and let* LowStretch *by an algorithm which takes as input an n'-node m'-edge graph G' and parameters $C, c > 0$ and returns a κ-distortion subgraph of G with at most $n' + \frac{m'}{(C\log\log n')^c}$ edges and a corresponding vector of leverage score overestimates in $O(m \log\log n)$ time, where*

$$\kappa = O\left(m\,(\log\log n')^{f(c)}\right)$$

for some concave, monotone increasing function f with $f(0) > 0$. Let $\delta > 0$ be a parameter, and let z denote the (unique) solution to the equation $f(z) + 2 + \delta = z$. Then for all sufficiently large n RecursiveSolver *is an ϵ-approximate Laplacian solver for G with running time*

$$O\left(m(\log\log n)^{z-1}\log(1/\epsilon)\right).$$

Proof. We first prove the algorithm is an ϵ-approximate Laplacian solver. We proceed by strong induction on m, the number of edges in the input graph G. Assume that RecursiveSolver's output is correct for all graphs with fewer than m edges. In one level of recursion, we construct a subgraph H with associated stretch overestimates τ with $n + \gamma^{-1}m$ edges that achieves κ-spectral distortion for

$$\kappa \leq \zeta m(\log\log n)^{f(z)},$$

where ζ is an absolute constant. We use this to form a graph G' with stretch overestimates $\tau' = \frac{\tau}{\eta}$. We observe that

$$\eta H \preceq G' \quad \text{and} \quad G \preceq G' = (\eta - 1)H + G \preceq \eta G$$

since H is a subgraph of G: thus we conclude that τ' are valid stretch overestimates of the edges in G'. Further, the choice of parameters η, κ, γ implies that PreconRichardson$_{G'}$ on line 7 applies RecursiveSolver to graphs with $O(\gamma^{-1}m + \eta^{-1}\kappa) = O(\gamma^{-1}m)$ edges: for sufficiently large constant C we see that this is less than m. Thus the calls to RecursiveSolver are correct by induction, and by Lemma 4.3 RichardsonSolver$_{G'}$ is a $\frac{1}{10\eta}$-solver for G'. Finally since $G' \approx_\eta G$ we conclude by Theorem 4.4 that PreconNoisyAGD is an ϵ-appoximate Laplacian solver: this completes the induction.

We now bound the running time of the algorithm. Fix a constant value m_0, and note that RecursiveSolver runs in $O(1)$ time for all graphs with fewer than m_0 edges since the proof of correctness implies that the algorithm runs in finite time. Let $\mathcal{T}(m, \epsilon)$ denote the running time of our algorithm on a graph with $m \geq m_0$ edges with error parameter ϵ. In one level of recursion, we first perform one call to LowStretch– this requires $O(m \log\log n)$ time. All remaining steps in our recursive algorithm are trivially $O(m)$ time except for the call to PreconNoisyAGD on line 8. We recall $\mathcal{L}_G \preceq \mathcal{L}_{G'} \preceq \eta \mathcal{L}_G$: by Theorem 4.4 the call to PreconNoisyAGD performs $O(\sqrt{\eta}\log(1/\epsilon))$ iterations, each of which performs $O(m)$ work plus one call to $\text{RichardsonSolver}_{G'}$ with error parameter $\frac{1}{10\eta}$. Observing that $\|\tau'\|_1 \leq \eta^{-1}\kappa = \gamma^{-1}m \leq m$, by Lemma 4.3 each of these calls to PreconRichardson runs in time $O(m\log(\eta))$ plus the time needed for $O(\log(\eta))$ calls to RecursiveSolver on graphs with $O(\gamma^{-1}m)$ edges and error parameter $\varepsilon_{solve} = \frac{1}{1600c_s^2\log^2 n}$. Thus, one recursive loop of the algorithm performs

$$O(m\log\log n) + O\left(m\sqrt{\eta}\log(1/\epsilon)\log\eta\right)$$

work plus the cost of $O\left(\sqrt{\eta}\log(1/\epsilon)\log\eta\right)$ calls to RecursiveSolver on graphs with at most $\beta\gamma^{-1}m$ edges (where β is the constant hidden in the big-O notation) and error parameter ε_{solve}. This implies the recurrence

$$\mathcal{T}(m,\epsilon) \leq \psi\left(\log\log n + \sqrt{\eta}\log(1/\epsilon)\log\eta\right)\left(m + \mathcal{T}\left(\beta\gamma^{-1}m, \varepsilon_{solve}\right)\right)$$

for some explicit constant ψ. Define $\widetilde{\mathcal{T}}(m) = \mathcal{T}(m, \varepsilon_{solve})$. We first establish and solve a recurrence for $\widetilde{\mathcal{T}}(m)$: we will use this to prove the full runtime claim. Since $\beta\gamma^{-1}m < m$ we may apply our induction hypothesis: further as $\eta \geq 1$ and $\log(1/\varepsilon_{solve}) \leq 2\log\log n + O(1) \leq 3\log\log n$ for n sufficiently large we have

$$\widetilde{\mathcal{T}}(m) \leq 4\psi\left(\sqrt{\eta}\log\eta\log\log n\right)\left(m + \widetilde{\mathcal{T}}(\beta\gamma^{-1}m)\right).$$

We will show that

$$4\psi\left(\sqrt{\eta}\log\eta\log\log n\right)\beta\gamma^{-1} \leq \frac{3}{4}$$

for the appropriate choice of constant C: this will allow us to apply the master theorem for this recurrence. Observe

$$\eta = \frac{\gamma\kappa}{m} \leq \zeta\left(\log\log n\right)^{f(z)}\gamma = \zeta\left(\log\log n\right)^{z-2-\delta}\gamma = \zeta C^{-1}\gamma^2\left(\log\log n\right)^{-2-\delta}. \quad (10)$$

Thus

$$4\psi\left(\sqrt{\eta}\log\eta\log\log n\right)\beta\gamma^{-1} \leq 4\psi\zeta^{1/2}C^{-1/2}\beta\log\eta\left(\log\log n\right)^{-\delta/2}.$$

As $\log\eta \leq 2\log\gamma + O(1) = O(\log\log\log n)$ and $m \geq m_0$, we choose m_0 sufficiently large and conclude

$$4\psi\left(\sqrt{\eta}\log\eta\log\log n\right)\beta\gamma^{-1} \leq 4\psi\zeta^{1/2}C^{-1/2} \leq \frac{3}{4}$$

for a sufficiently large choice of C. Thus for any $m \geq m_0$ we have

$$\widetilde{\mathcal{T}}(m) \leq \frac{3}{4}\alpha^{-1}\left(m + \widetilde{\mathcal{T}}(\alpha m)\right)$$

for $\alpha = \beta\gamma^{-1}$. The master theorem thus implies

$$\widetilde{\mathcal{T}}(m) \leq O(\alpha m) = O\left(m\left(\log\log n\right)^z\right)$$

for all m. Applying this to the recurrence for $\mathcal{T}(m,\epsilon)$, we obtain

$$\begin{aligned}\mathcal{T}(m,\epsilon) &\leq \psi\left(\log\log n + \sqrt{\eta}\log(1/\epsilon)\log\eta\right)\left(m + \widetilde{\mathcal{T}}\left(\beta\gamma^{-1}m\right)\right)\\ &\leq O\left(m\log\log n + m\sqrt{\eta}\log(1/\epsilon)\log\eta\right)\end{aligned}$$

since $\widetilde{\mathcal{T}}(\beta\gamma^{-1}m) = O(m)$. We note by (10) that

$$\sqrt{\eta} \leq O\left(\gamma\left(\log\log n\right)^{-1-\delta/2}\right) = O\left(\left(\log\log n\right)^{z-1-\delta/2}\right).$$

We now observe $\log\eta \leq (\log\log n)^{\delta/2}$ for large enough n and $z = 2+\delta+f(z) > 2$: these together imply

$$\mathcal{T}(m,\epsilon) \leq O\left(m\log\log n + m\left(\log\log n\right)^{z-1}\log(1/\epsilon)\right) = O\left(m\left(\log\log n\right)^{z-1}\log(1/\epsilon)\right)$$

as desired.

□

Finally, we apply the κ-distortion subgraphs computed in Theorem 1.5 to obtain our main result:

Theorem 1.6 ($\tilde{O}(m)$-Laplacian System Solver). *There is a randomized algorithm which is an ϵ-approximate Laplacian system solver for any input n-vertex m-edge graph with polynomially-bounded edge weights (see Definition 1.2) and $\epsilon \in (0,1)$ and has the following runtime for any $\chi > 0$*

$$O(m(\log\log n)^{6+2\sqrt{10}+\chi}\log(1/\epsilon)).$$

Proof. We observe that the procedure given in Theorem 1.5 yields κ-distortion subgraphs with $n + \frac{m}{C(\log\log n)^c}$ edges, for

$$\kappa = O\left(m\left(\log\log n\right)^{1+\sqrt{8c}+o(1)}\right).$$

For sufficiently large n and constant in the big-O notation, this satisfies the conditions of Theorem 4.5 for function $f(x) = 1+\sqrt{8x}+\delta$ for any $\delta > 0$. Theorem 1.5 constructs such subgraphs in $O(m)$ time which is sufficient for our guarantee. Substituting these into the guarantee of Theorem 12 gives an ϵ-approximate solver running in time

$$O\left(m(\log\log n)^{z-1}\log(1/\epsilon)\right),$$

where z is the solution to $\sqrt{8z}+3+2\delta = z$ for any $\delta \geq 0$. Solving this equation reveals $z = 7+\sqrt{40+8\delta}+2\delta$: by choosing δ sufficiently small this gives an exponent of $6+\sqrt{40}+\chi$ for any $\chi \geq 0$. We remark that $6+\sqrt{40} \approx 12.324$. □

We made only limited attempts to optimize the loglog dependence of this algorithm. We believe this dependence may be improved and here describe possible improvements. First, our method of analyzing the recursion is in some sense, weaker than that of [CKM+]. In their setting the low stretch subgraph is a tree: in that case the recursively generated subproblems natively contain a low-distortion subgraph with $\kappa = O(m)$. This observation enables them to make different choices for the recursion parameter η. Although this observation is key to ensure their algorithm runs in $\tilde{O}(m\sqrt{\log n})$ time and not $\tilde{O}(m\log n)$, in our case the only effect is to reduce the polynomial dependence

on $\log \log n$. By carefully applying this technique to our setting we believe our runtime can be improved.

A larger obstruction in obtaining a better runtime is the use of a "bottom-up" recursion based on [AKPW95] in our construction of κ-distortion subgraphs. While this suffices to obtain our claim, our running time would be much improved by using a "top-down" graph decomposition more closely resembling [CKM+]. We leave it as an interesting problem for future work.

As an additional remark we observe that we can convert the expected-decrease guarantee of Theorem 1.6 to a high-probability bound, provided that we allow our runtime bound to hold in expectation and with an extra $O(\log \log n)$ factor. To obtain this, we use Lemmas 4.5 and 4.9 from [CKP+14], which allow us to construct a linear operator $\mathbf{Z}$ satisfying $\mathcal{L}^\dagger \preceq \mathbf{Z} \preceq \log^4 n \mathcal{L}^\dagger$ which can be computed and applied in $O(m \log \log n)$ time.[7] With this, we simply apply Theorem 1.6 with $\epsilon \leftarrow \frac{\epsilon}{2 \log^4 n}$: Markov's inequality implies the output x has $\|\mathcal{L}x - b\|^2_{\mathcal{L}^\dagger} = \left\|x - \mathcal{L}^\dagger b\right\|^2_{\mathcal{L}} \leq \frac{\epsilon}{\log^4 n} \|b\|^2_{\mathcal{L}}$ with probability $1/2$. If this holds, we may use $\mathbf{Z}$ to verify in $O(m \log \log m)$ time whether $\left\|x - \mathcal{L}^\dagger b\right\|^2_{\mathcal{L}} \leq \epsilon \|b\|^2_{\mathcal{L}}$. The claim follows by repeating this procedure until a desired solution is found.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their many helpful comments and feedback on earlier versions (and in particular inquiring about whp. error bounds). We thank Yair Carmon, Yang P. Liu, Michael Kapralov, Jonathan Kelner, Navid Nouri, Richard Peng, and Jakab Tardos for helpful discussions. The research was supported by NSF CAREER Award CCF-1844855 and a PayPal research gift award.

REFERENCES

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

[Awe85] Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[Bar] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proccedings of the 30th Annual ACM SIGACT Symposium on Theory of Computing, STOC 1998*, pages 161–168.

[BGK+] Guy E. Blelloch, Anupam Gupta, Ioannis Koutis, Gary L. Miller, Richard Peng, and Kanat Tangwongsan. Nearly-linear work parallel SDD solvers, low-diameter decomposition, and low-stretch subgraphs. *Theory Comput. Syst.*, 55(3):521–554.

[BJL+] Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. In *Advances in Neural Information Processing Systems 32: NeurIPS 2019*, pages 13900–13909.

[BP] Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019*, pages 541–543.

[BSS14] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Review*, 56(2):315–334, 2014.

[7] Lemma 4.5 from [CKP+14] implies an $O(m \log \log n)$-time algorithm to construct a 'graph-tree tuple' with $m + n$ edges and $\|\tau\|_p^p \leq O(m \log^p n)$ in $O(m)$ time for any $p \in (1/2, 1)$. Lemma 4.9 can be modified to use the solver of [CKM+] instead of [KMP] to apply $\mathbf{Z}$: doing so enables us to apply it in $O(m + \log^{-2p} \|\tau\|_p^p \sqrt{\log n} (\log \log n)^4)$ expected time. Plugging in Lemma 4.5 and choosing $p = 2/3$ yields an expected running time bound of $O(m)$.

[BvdBG+20] Aaron Bernstein, Jan van den Brand, Maximilian Probst Gutenberg, Danupon Nanongkai, Thatchaphol Saranurak, Aaron Sidford, and He Sun. Fully-dynamic graph sparsifiers against an adaptive adversary. *arxiv:2004.08432*, 2020.

[CKM+] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m\log^{1/2} n$ time. In *Proccedings of the 46th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2014*, pages 343–352.

[CKP+a] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proccedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 410–419.

[CKP+b] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 583–592.

[CKP+14] Michael B. Cohen, Rasmus Kyng, Jakub W. Pachocki, Richard Peng, and Anup B. Rao. Preconditioning in expectation. *arxiv:1401.6236*, 2014.

[CLM+14] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. *arxiv:1408.5099*, 2014.

[DGN14] Olivier Devolder, François Glineur, and Yurii E. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Math. Program.*, 146(1-2):37–75, 2014.

[DK] Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011*, pages 169–178.

[DS] Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proccedings of the 40th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2008*, pages 451–460.

[GGV] Cyril Gavoille, Quentin Godfroy, and Laurent Viennot. Node-disjoint multipath spanners and their relationship with fault-tolerant spanners. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011*, volume 7109 of *Lecture Notes in Computer Science*, pages 143–158.

[HSS] Oliver Hinder, Aaron Sidford, and Nimit Sharad Sohoni. Near-optimal methods for minimizing star-convex functions and beyond. In *Conference on Learning Theory, COLT 2020*, pages 1894–1938.

[KLOS] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 217–226.

[KLPa] Ioannis Koutis, Alex Levin, and Richard Peng. Improved spectral sparsification and numerical algorithms for SDD matrices. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*, pages 266–277.

[KLP+b] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proccedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 842–850.

[KMP] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-m log n time solver for SDD linear systems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 590–598.

[KMP14] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. *SIAM J. Comput.*, 43(1):337–354, 2014.

[KMST] Alexandra Kolla, Yury Makarychev, Amin Saberi, and Shang-Hua Teng. Subgraph sparsification and nearly optimal ultrasparsifiers. In *Proceedings of the 42nd ACM SIGACT Symposium on Theory of Computing, STOC 2010*, pages 57–66.

[KOSA] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proccedings of the 45th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2013*, pages 911–920.

[KR] Jon M. Kleinberg and Ronitt Rubinfeld. Short paths in expander graphs. In *IEEE 37th Annual Symposium on Foundations of Computer Science, FOCS 1996*, pages 86–95.

[KS] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians - fast, sparse, and simple. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 573–582.

[Lee14] Yin Tat Lee. Probabilistic spectral sparsification in sublinear time. *arxiv:1401.0085*, 2014.

[Li] Jason Li. Faster parallel algorithm for approximate shortest path. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 308–321.

[LS] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS 2013*, pages 147–156.

[LSY] Yang P. Liu, Sushant Sachdeva, and Zejun Yu. Short cycles via low-diameter decompositions. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2602–2615.

[MPX] Gary L. Miller, Richard Peng, and Shen Chen Xu. Parallel graph decompositions using random shifts. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2013*, pages 196–203.

[MS13] Renato D. C. Monteiro and Benar Fux Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM J. Optim.*, 23(2):1092–1125, 2013.

[Nes83] Y. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

[Pen] Richard Peng. Approximate undirected maximum flows in $O(mpolylog(n))$ time. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 1862–1867.

[Pen13] Richard Peng. Algorithm design using spectral graph theory. *Ph.D Thesis*, 2013.

[PS] Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In *Proccedings of the 46th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2014*, pages 333–342.

[Sch03] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Number v. 1 in Algorithms and Combinatorics. Springer, 2003.

[Shea] Jonah Sherman. Area-convexity, l_∞ regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 452–460.

[Sheb] Jonah Sherman. Nearly maximum flows in nearly linear time. In *IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS 2013*, pages 263–269.

[SS] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proccedings of the 40th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2008*, pages 563–568.

[ST] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th ACM SIGACT Symposium on Theory of Computing, STOC 2004*, pages 81–90.

[SW] Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2616–2635.

A ULTRASPARSIFIERS BY SPECTRAL GRAPH THEORY

In this section, we prove our two main results regarding ultrasparsifiers. We begin by proving the existence of ultrasparsifiers for sums of arbitrary rank-1 matrices.

Theorem 1.3 (Ultrasparsifier Existence). *Let $v_1, \ldots v_m \in \mathbb{R}^n$ and $\boldsymbol{A} \coloneqq \sum_{i\in[m]} v_i v_i^\top$. For any integer $k \geq 2$, there exists $S \subseteq [m]$ with $|S| = n + O\left(\frac{n}{\sqrt{k}}\right)$ and $w \in \mathbb{R}^m_{\geq 0}$ where*

$$\boldsymbol{A} \preceq \sum_{i\in S} w_i v_i v_i^\top \preceq k\boldsymbol{A}.$$

To obtain this result we first give the following Theorem A.1 regarding spectral properties of subsets of sums of rank one matrices and then we use it to prove Theorem 1.3.

Theorem A.1. *Let $v_1 \ldots v_m \in \mathbb{R}^n$ and let $\boldsymbol{A} = \sum_{i\in[m]} v_i v_i^\top$ be full rank. For any $k \geq 1$ there exists $S \subset [m]$ with $|S| \leq n + \frac{n}{k}$ such that $\boldsymbol{B} \coloneqq \sum_{i\in S} v_i v_i^\top$ has $\operatorname{tr}[\boldsymbol{B}^{-1}\boldsymbol{A}] \leq mk$.*

Proof. We start with $S = [m]$ and greedily remove elements from S to minimize the increase in $\operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}]$. Observe that the initial value of this trace is n since $\mathbf{B} = \mathbf{A}$. By the Sherman-Morrison formula for rank 1 matrix updates, for any invertible matrix $\mathbf{M}$ and vector v with $v^\top \mathbf{M}^{-1} v \neq 1$

$$\left(\mathbf{M} - vv^\top\right)^{-1} = \mathbf{M}^{-1} + \frac{\mathbf{M}^{-1}vv^\top\mathbf{M}^{-1}}{1 - v^\top\mathbf{M}^{-1}v} \text{ and is invertible .}$$

With this, we analyze the change to $\operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}]$ after one element is removed from S. For any $i \in S$,

$$\operatorname{tr}\left[\left(\mathbf{B} - v_i v_i^\top\right)^{-1}\mathbf{A}\right] = \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}] + \frac{v_i^\top\mathbf{B}^{-1}\mathbf{A}\mathbf{B}^{-1}v_i}{1 - v_i^\top\mathbf{B}^{-1}v_i}$$

We consider randomly sampling i to remove from S with probability $p_i \propto 1 - v_i^\top\mathbf{B}^{-1}v_i$. We will show the increase to the trace is bounded in expectation. Since for $i \in S$, $v_i v_i^\top \preceq \mathbf{B}$, all the p_i are all nonnegative and non-zero only when $1 - v_i^\top\mathbf{B}^{-1}v_i > 0$. Selecting i in this way yields

$$\begin{aligned}\mathbb{E}\left[\operatorname{tr}\left[\left(\mathbf{B} - v_i v_i^\top\right)^{-1}\mathbf{A}\right]\right] &= \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}] + \sum_{i\in S} p_i \frac{v_i^\top\mathbf{B}^{-1}\mathbf{A}\mathbf{B}^{-1}v_i}{1 - v_i^\top\mathbf{B}^{-1}v_i} \\ &= \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}] + \frac{\sum_{i\in S} v_i^\top\mathbf{B}^{-1}\mathbf{A}\mathbf{B}^{-1}v_i}{\sum_{i\in S} 1 - v_i^\top\mathbf{B}^{-1}v_i}.\end{aligned}$$

By the cyclic property of trace, we observe $\sum_{i\in S} v_i^\top\mathbf{B}^{-1}\mathbf{A}\mathbf{B}^{-1}v_i = \operatorname{tr}[\mathbf{B}\mathbf{B}^{-1}\mathbf{A}\mathbf{B}^{-1}] = \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}]$ and $\sum_{i\in S} 1 - v_i^\top\mathbf{B}^{-1}v_i = |S| - n$. Applying these equations yields

$$\mathbb{E}\left[\operatorname{tr}\left[\left(\mathbf{B} - v_i v_i^\top\right)^{-1}\mathbf{A}\right]\right] = \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}]\left(1 + \frac{1}{|S| - n}\right)$$

and therefore whenever $|S| > n$, there exists some $i \in S$ satisfying

$$\operatorname{tr}\left[\left(\mathbf{B} - v_i v_i^\top\right)^{-1}\mathbf{A}\right] \leq \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}]\left(1 + \frac{1}{|S| - n}\right) = \operatorname{tr}[\mathbf{B}^{-1}\mathbf{A}]\left(\frac{|S| - n + 1}{|S| - n}\right).$$

By repeatedly applying this bound from $|S| = m$ to $|S| = n + \lceil\frac{n}{k}\rceil$, we remove all but $n + \frac{n}{k}$ elements from S and end up with $\mathbf{B}$ satisfying the desired bound of

$$\operatorname{tr}\left[\mathbf{B}^{-1}\mathbf{A}\right] \leq n \prod_{s=n+\lceil\frac{n}{k}\rceil}^{m}\left(\frac{s - n + 1}{s - n}\right) = n\left(\frac{m - n + 1}{n/k}\right) \leq mk\ .$$

□

We now show a modification of [BSS14] which allows us to convert the output of Theorem A.1 into an ultrasparsifier.

Theorem A.2. *Let $v_1, v_2, ..., v_m \in \mathbb{R}^n$ be vectors such that $\sum_i v_i v_i^\top = \boldsymbol{I}$. Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be a matrix satisfying $\boldsymbol{A} \preceq \boldsymbol{I}$ and $\mathrm{tr}[\boldsymbol{A}^{-1}] = \kappa$. Then for any $q \geq 0$ with $\lceil (\kappa + 2n)q \rceil \leq n$, there exists $S \subseteq [m]$ with $|S| = \lceil (\kappa + 2n)q \rceil$ with corresponding weights $w_i > 0$ such that*

$$q\boldsymbol{I} \preceq \boldsymbol{A} + \sum_{i \in S} w_i v_i v_i^\top \preceq 3\boldsymbol{I}.$$

Our proof of this result is as a consequence of technical lemmas from [BSS14] restated below.

Lemma A.3 (Combination of Lemmas 3.3, 3.4, 3.5 from [BSS14])**.** *Let $v_1, v_2, ...v_m \in \mathbb{R}^n$ be vectors such that $\sum_i v_i v_i^\top = \boldsymbol{I}$. Define the functions $\Phi^u(\boldsymbol{M}) \coloneqq \mathrm{tr}[(u\boldsymbol{I} - \boldsymbol{M})^{-1}]$ and $\Phi_l(\boldsymbol{M}) \coloneqq \mathrm{tr}[(\boldsymbol{M} - l\boldsymbol{I})^{-1}]$. Let $\boldsymbol{A}$ be a matrix satisfying $l\boldsymbol{I} \preceq \boldsymbol{A} \preceq u\boldsymbol{I}$ as well as*

$$\Phi^u(\boldsymbol{A}) \leq \gamma_U \quad \text{and} \quad \Phi_l(\boldsymbol{A}) \leq \gamma_L.$$

Then for δ_U, δ_L satisfying $1/\delta_U + \gamma_U \leq 1/\delta_L - \gamma_L$ there exists $i \in [m]$ and $t > 0$ such that $(l + \delta_L)\boldsymbol{I} \preceq \boldsymbol{A} + tv_i v_i^\top \preceq (u + \delta_U)\boldsymbol{I}$ as well as

$$\Phi^{u+\delta_U}\left(\boldsymbol{A} + tv_i v_i^\top\right) \leq \gamma_U \quad \text{and} \quad \Phi_{l+\delta_L}\left(\boldsymbol{A} + tv_i v_i^\top\right) \leq \gamma_L.$$

Our use of Lemma A.3 mirrors its use in [KMST]: we iteratively add vectors to the output of Theorem A.1 to increase the spectral upper and lower bounds on $\mathbf{A}$ appropriately. After a small number of iterations, we certify that $\mathbf{A}$ is appropriately spectrally bounded and terminate.

Proof of Theorem A.2. We let $\mathbf{A}^{(0)} \coloneqq \mathbf{A}$ and for $j \geq 0$ iteratively define $\mathbf{A}^{(j+1)} \coloneqq \mathbf{A}^{(j)} + tv_i v_i^\top$ for some $t \geq 0$ and $i \in [m]$ (each depending on j). Further, we set $\gamma_U = n$, $\gamma_L = \kappa$, $u_0 = 2$, $l_0 = 0$, $\delta_U = \frac{1}{n}$, $\delta_L = \frac{1}{2n+k}$. Observe that

$$\Phi^{u_0}\left(\mathbf{A}^{(0)}\right) = \mathrm{tr}\left(2\mathbf{I} - \mathbf{A}\right)^{-1} \leq \mathrm{tr}\left(\mathbf{I}\right) \leq n = \gamma_U$$

and

$$\Phi_{l_0}\left(\mathbf{A}^{(0)}\right) = \mathrm{tr}\left(\mathbf{A}^{-1}\right) = \kappa = \gamma_L.$$

Further, the choice of parameters ensures $1/\delta_U + \gamma_U \leq 1/\delta_L - \gamma_L$. Thus, inductively applying Lemma A.3 yields that for each $j > 1$ there exists $t > 0$, and $i \in [S]$ where $\mathbf{A}^{(j)} = \mathbf{A}^{(j-1)} + tv_i v_i^\top$ satisfies

$$\Phi^{u_0 + j\delta_U}\left(\mathbf{A}^{(j)}\right) \leq \gamma_U \quad \text{and} \quad \Phi_{l_0 + j\delta_L}\left(\mathbf{A}^{(j)}\right) \leq \gamma_L.$$

Therefore $\mathbf{A}^{(s)}$ for $s = \lceil (\kappa + 2n)q \rceil$ satisfies the desired bound of

$$q\mathbf{I} \preceq \left(l_0 + s\delta_L\right)\mathbf{I} \preceq \mathbf{A}^{(s)} \preceq \left(u_0 + s\delta_U\right)\mathbf{I} \preceq 3\mathbf{I}$$

where in the last inequality we used the upper bound on s. □

Finally, we combine Theorem A.1 and Theorem A.2 and give the proof of Theorem 1.3.

Proof of Theorem 1.3. We note that we may assume $\mathbf{A} = \sum_{i\in[m]} v_i v_i^\top$ is full-rank: otherwise we may add $u_j \in \ker(\mathbf{A})$ to make the result full rank, run the rest of the proof, and remove the added u_j before returning the output. Since the u_j are orthogonal to the v_i, removing them cannot affect the space spanned by $\mathbf{A}$'s eigenvectors.

By applying [BSS14], we can find a collection of $34n$ vectors $v'_1, v'_2, \ldots, v'_{34n}$ which are reweighted copies of the v_i and satisfy $\mathbf{A} \preceq \sum_{i\in[16n]} v'_i \left(v'_i\right)^\top \preceq 2\mathbf{A}$. We assume $k > 1000$ in the rest of the argument: the v'_i immediately satisfy our requirements otherwise as k was assumed to be at least 2.

Given the vectors $v'_1, v'_2, \ldots, v'_{34n}$, define $\mathbf{C} = \sum_{i\in[34n]} v'_i \left(v'_i\right)^\top$: note that $\frac{\mathbf{A}}{\preceq}\mathbf{C} \preceq 2\mathbf{A}$. As $\mathbf{C}$ is therefore full-rank, we apply Theorem A.1 and thus obtain a set S of $n + \frac{n}{k}$ vectors such that

$$\operatorname{tr}\left(\mathbf{C}\left(\sum_{i\in S} v_i v_i^\top\right)^{-1}\right) \leq 34nk.$$

Thus, $\mathbf{B} = \mathbf{C}^{-1/2}\left(\sum_{i\in S} v_i v_i^\top\right)\mathbf{C}^{-1/2}$ has $\mathbf{B} \preceq \mathbf{I}$ (as it is formed from an unweighted subset of the vectors that form $\mathbf{C}$) and $\operatorname{tr}(\mathbf{B}^{-1}) \leq 34nk$. Applying Theorem A.2 with $\kappa = 34nk$ and $q = \frac{1}{36k^2}$, we observe that there exists a set T of $(\kappa + 2n)q \leq \frac{n}{k}$ vectors with corresponding weights $w_i \geq 0$ such that

$$\frac{1}{36k^2}\mathbf{C} \preceq \mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2} + \sum_{i\in T} w_i v_i v_i^\top \preceq 3\mathbf{C}.$$

Using the definition of $\mathbf{B}$ and rearranging, we obtain

$$\frac{1}{36k^2}\mathbf{A} \preceq \frac{1}{36k^2}\mathbf{C} \preceq \sum_{i\in S\cup T} w_i v_i v_i^\top \preceq 3\mathbf{C} \preceq 6\mathbf{A}$$

Finally, $|S \cup T| \leq |S| + |T| \leq n + \frac{n}{k} + \frac{n}{k} = n + \frac{2n}{k}$: the output is a sum of outer products of at most $n + \frac{2n}{k}$ vectors. The claim follows by choosing $k \leftarrow 216k^2$ and scaling the output. □

We made no attempt to optimize the constants in the above proof. We additionally remark that Theorem 1.3 immediately implies the existence of k-ultrasparsifiers with $n + O\left(\frac{n}{\sqrt{k-1}}\right)$ edges: we simply apply it to a graph Laplacian written in the form $\sum_{e\in E(G)}\left(\sqrt{w_e}b_e\right)\left(\sqrt{w_e}b_e\right)^\top$. We now construct ultrasparsifiers for graphs with improved guarantees by combining our κ-distortion subgraph construction with this BSS-derived framework. We begin with the general claim of our construction: we specialize it to several interesting parameter regimes as a corollary.

Theorem A.4 (Polynomial-Time Ultrasparsifier Construction in Graphs). *Let G be any n-vertex graph with polynomially-bounded edge weights. There exists a polynomial time algorithm which constructs a reweighted subgraph H with $n + O\left(\frac{n}{\gamma}\right)$*

edges such that $\mathcal{L}_H \preceq \mathcal{L}_G \preceq \alpha\mathcal{L}_H$ *for*

$$\alpha = O\left(\gamma \exp\left(\sqrt{8\log\gamma \cdot \log\left(48\log\left(\gamma\log^{10} n\right)\sqrt{\log\gamma}\right)}\right)\log\left(\gamma\log^{10} n\right)\sqrt{\log\gamma}\right)$$

Proof. We note that by preprocessing the graph with [BSS14], we may assume that the graph has $O(n)$ edges with at most a constant factor loss in the final approximation error. We employ Theorem 2.5 with $k = \gamma\log^{10} n$ and path sparsification algorithm given by Theorem 1.9. By the theorem's guarantee, this returns a subgraph G' in polynomial time with

$$n + O\left(\frac{n\log^{10} n}{k} + \frac{n\log\gamma}{k^2} + \frac{n}{\gamma}\right) = n + O\left(\frac{n\log^{10} n}{\gamma\log^{10} n} + \frac{n}{\gamma}\right) = n + O\left(\frac{n}{\gamma}\right)$$

edges. Further, the output subgraph is a κ-distortion subgraph with

$$\kappa = O\left(n\gamma \exp\left(\sqrt{8\log\gamma \cdot \log\left(48\log\left(\gamma\log^{10} n\right)\sqrt{\log\gamma}\right)}\right)\log\left(\gamma\log^{10} n\right)\sqrt{\log\gamma}\right).$$

We note that $\mathrm{tr}\left[\mathcal{L}_G^{1/2}\mathcal{L}_{G'}^{\dagger}\mathcal{L}_G^{1/2}\right] \leq \kappa$ by definition of κ-distortion, and that the collection of rank-1 matrices $w_e\mathcal{L}_G^{-1/2}b_e b_e^\top\mathcal{L}_G^{-1/2}, e \in G$ sums to $\mathbf{I}$. If $b_e b_e^\top$ are the edge Laplacian matrices that form $\mathcal{L}_G$, applying Theorem A.2 with $\alpha := 3\frac{n}{\kappa\gamma}$ yields a set S of $\lceil(\kappa + 2n)\alpha\rceil = O\left(\frac{n}{\gamma}\right)$ edges with corresponding weights $w_i' > 0$ such that

$$3\alpha\mathbf{I} \preceq \mathcal{L}_G^{-1/2}\mathcal{L}_{G'}\mathcal{L}_G^{-1/2} + \sum_{e\in S} w_e'\mathcal{L}_G^{-1/2}b_e b_e^\top\mathcal{L}_G^{-1/2} \preceq 3\mathbf{I}.$$

Scaling down the resulting matrix and multiplying both sides of the matrices by $\mathcal{L}_G^{1/2}$ gives a reweighted subgraph H with $n + O\left(\frac{n}{\gamma}\right)$ such that $\alpha\mathcal{L}_G \preceq \mathcal{L}_H \preceq \mathcal{L}_G$ as desired. □

As a corollary of this result, we prove Theorem 1.7, which we now recall.

Theorem 1.7 (Improved Ultrasparsifiers). *There exists a polynomial time algorithm which given an input graph G with polynomially-bounded edge weights can compute a reweighted subgraph H with either of the following guarantees:*

- *For any constant c, H has $n + \frac{n}{(\log\log n)^c}$ edges and satisfies*

$$\mathcal{L}_G \preceq \mathcal{L}_H \preceq O((\log\log n)^{c+\sqrt{8c}+1+o(1)})\mathcal{L}_G.$$

- *For any constant $\delta > 0$ and $\alpha = \omega(\log^\delta n)$, H has $n + \frac{n}{\alpha}$ edges and satisfies*

$$\mathcal{L}_G \preceq \mathcal{L}_H \preceq \alpha^{1+o(1)}\mathcal{L}_G.$$

Proof. We employ Theorem A.4 with different values of γ. For the first claim, we choose $\gamma = (\log\log n)^c$, and note that Theorem A.4 yields α-ultrasparsifiers, with

$$\begin{aligned}\alpha &= O\left(\gamma \exp\left(\sqrt{8\log\gamma \cdot \log\left(48\log\left(\gamma\log^{10} n\right)\sqrt{\log\gamma}\right)}\right)\log\left(\gamma\log^{10} n\right)\sqrt{\log\gamma}\right)\\ &= O\left((\log\log n)^{c+\sqrt{8c}+1+o(1)}\right)\end{aligned}$$

by applying the definition of γ. For the second claim, we choose $\gamma = \alpha$ in Theorem A.4: we obtain an ultrasparsifier of quality

$$O\left(\alpha \exp\left(\sqrt{8\log\alpha \cdot \log\left(48\log\left(\alpha\log^{10} n\right)\sqrt{\log\alpha}\right)}\right)\log\left(\alpha\log^{10} n\right)\sqrt{\log\alpha}\right).$$

Since $\alpha = \omega(\log^{\delta} n)$ for some fixed constant $\delta > 0$, we have $\log\left(\alpha\log^{10} n\right) \leq \log\left(\alpha^{1+10/\delta}\right) + O(1) = \left(1+\frac{10}{\delta}\right)\log\alpha + O(1) \leq \left(2+\frac{20}{\delta}\right)\log\alpha$ for sufficiently large n. Substituting this in, we obtain

$$\begin{aligned}
&O\left(\alpha \exp\left(\sqrt{8\log\alpha \cdot \log\left(48\log\left(\alpha\log^{10} n\right)\sqrt{\log\alpha}\right)}\right)\log\left(\alpha\log^{10} n\right)\sqrt{\log\alpha}\right)\\
&\leq O\left(\alpha \exp\left(\sqrt{8\log\alpha \cdot \log\left(\left(96+\frac{960}{\delta}\right)\log^{3/2}\alpha\right)}\right)\log^{3/2}\alpha\right)\\
&= O\left(\alpha \exp\left(\sqrt{8\log\alpha \cdot \left(\frac{3}{2}\log\log\alpha + O(1)\right)}\right)\log^{3/2}\alpha\right) = \alpha^{1+o(1)}
\end{aligned}$$

as claimed. □

B PRIMAL DUAL CHARACTERIZATION OF SHORTEST FLOW

Here we prove our primal-dual characterization of minimum cost flow that we use to reason about vertex disjoint paths in expanders. Though this is fairly standard and straightforward, we include a brief derivation here for completeness.

Lemma 3.6 (Dual Characterization of Shortest Flow)**.** *For directed graph $G = (V, E)$ and vertices $s, t \in V$ if there are at most F edge-disjoint paths from s to t then there is an integral s-t flow $f \in \{0,1\}^E$ corresponding to F edge-disjoint paths from s to t using a minimum number of edges and $v \in \mathbb{R}^V$ such that for every $(a,b) \in e$ if $f_e = 1$ then $v_a - v_b \geq 1$ and if $f_e = 0$ then $v_a - v_b \leq 1$.*

Proof of Lemma 3.6. For all $a, b \in V$ let $1_{a,b} \coloneqq 1_a - 1_b$ where for all $c \in V$ we et $1_c \in \mathbb{R}^V$ denote the indicator vector for c, i.e. the vector that is a zero in all coordinates except for c where it has value 1. Further, let $\mathbf{B} \in \mathbb{R}^{E\times V}$ denote the edge-vertex incidence matrix of graph where for each edge $e = (a,b) \in E$ row e of $\mathbf{B}$ is $\delta_{a,b}$.

Leveraging this notation, we consider the following minimum cost flow problem of computing the flow of minimum total length that sends F units of flow from s to t and puts at most one unit of non-negative flow is put on each edge:

$$\min_{f\in\mathbb{R}^E : f_e \in [0,1] \text{ for all } e\in E \text{ and } \mathbf{B}^\top f = F\cdot\delta_{s,t}} f^\top \vec{1} \tag{11}$$

To see that (11) corresponds to the desired flow problem, note that for all $a \in V$ and $f \in \mathbb{R}^E$, $[\mathbf{B}^\top f]_a = 1_a^\top \sum_{e=(a,b)\in E} \delta_{a,b} f_e = \sum_{e=(a,b)\in E} f_e - \sum_{e=(b,a)} f_e$, i.e. the net flow leaving leaving vertex a through f in the graph.

There is always an integral minimizer for this problem and it corresponds to F disjoint paths from s to t using a minimum number of edges.[8] Letting, $\mathbf{0}_E, \mathbf{I}_E \in \mathbb{R}^{E\times E}$

[8]This is a standard result regarding minimum cost flow. One way to see this is to note that given any solution f to (11) if the edges e with $f_e \notin \{0,1\}$ form a cycle (viewing each directed edge (a,b) as an

denote the all zero matrix and identity matrix respectively, letting $\vec{0}_E, \vec{1}_E \in \mathbb{R}^E$ denote the all zero vector and all ones vector respectively, and letting

$$\mathbf{A} = \begin{pmatrix} \mathbf{B} & \mathbf{I}_E \\ \mathbf{0}_E & \mathbf{I}_E \end{pmatrix}, b = \begin{pmatrix} \vec{1}_m \\ \vec{0}_m \end{pmatrix}, \text{ and } c = \begin{pmatrix} F \cdot \delta_{s,t} \\ \vec{1}_m \end{pmatrix}$$

we can write this problem equivalently as

$$(P) = \min_{x \in \mathbb{R}^{E+E}_{\geq 0} \,:\, \mathbf{A}^\top x = b} b^\top x \text{ and } (D) = \max_{y \in \mathbb{R}^{V+E}, s \in \mathbb{R}^{E+E}_{\geq 0} \,:\, \mathbf{A}x - s = b} c^\top y \tag{12}$$

where we use $\mathbb{R}^{E+E}$ and $\mathbb{R}^{V+E}$ denote concatenations of two $\mathbb{R}^E$ vectors and concatenation of a $\mathbb{R}^V$ vector with a $\mathbb{R}^E$ vector, respectively. That (P) is equivalent to the original minimum cost flow problem follows from the fact that $f \leq \vec{1}$ entrywise if and only if $f + x = \vec{1}$ for some $x \in \mathbb{R}^E_{\geq 0}$ and that $(D) = (P)$ follows from standard strong duality of linear programs.

Now, let $x \in \mathbb{R}^{E+E}_{\geq 0}$ and $(y, s) \in \mathbb{R}^{V+E} \times \mathbb{R}^{E \times E}_{\geq 0}$ be optimal solutions ot (P) and (D) respectively in (12). Further, let without loss of generality f be the concatenation of $f \in \mathbb{R}^E_{\geq 0}$ and $\vec{1} - f \in \mathbb{R}^E_{\geq 0}$, let y be the concatenation of $v \in \mathbb{R}^V$ and $z \in \mathbb{R}^E$, and let s be the concatenation of $s^1 \in \mathbb{R}^E_{\geq 0}$ and $s^2 \in \mathbb{R}^E_{\geq 0}$. Further, let f be an integral minimizer and note that it corresponds to F disjoint paths, and either $f_e = 0$ or $1 - f_e = 0$ for all $e \in E$.

Now, by optimality of x and s we know that $x^\top s = 0$ and therefore $f_e \cdot s^1_e = 0$ for all $e \in E$ and $(1 - f_e) \cdot s^2_e = 0$ for all $e \in E$. Further, for all edges $e \in (a, b)$ feasibility of (y, s) for (D) implies

$$v_a - v_b + z_e - s^1_e = 1 \text{ and } z_e - s^2_e = 0 \,.$$

Consequently, if $f_e = 1$ then $s^1_e = 0$ and we have $s^1_e = 0$, $z_e = s^2_e$ and $v_a - v_b = 1 - s^2_e$, i.e. $v_a - v_b \leq 1$. Further, if $f_e = 0$ then $s^2_e = 0$, $z_e = 0$, and $v_a - v_b = 1 + s^1_e$, i.e. $v_a - v_b \geq 1$. □

C RANDOMIZED PRECONDITIONED ACCELERATED GRADIENT DESCENT

In this section we prove the following Theorem 4.4 regarding preconditioned accelerated gradient descent (AGD) for solving linear systems with random error in the preconditioner. That this accelerated preconditioned linear system solver handles randomized error aids our analysis in Section 4.

Theorem 4.4 (Randomized Preconditioned AGD). *Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$ be symmetric PSD matrices with $\boldsymbol{A} \preceq \boldsymbol{B} \preceq \kappa \boldsymbol{A}$ for $\kappa \geq 1$, let $b \in \mathrm{im}(\boldsymbol{A})$, let $\epsilon \in (0, 1)$ and let $\mathrm{Solve}_{\boldsymbol{B}}$ be a $\frac{1}{10\kappa}$-approximate solver for $\boldsymbol{B}$. Then $\mathrm{PreconditionedNoisyAGD}(\boldsymbol{A}, b, \epsilon, \kappa, \mathrm{Solve}_{\boldsymbol{B}})$ (Algorithm 13) is an ϵ-approximate solver for $\boldsymbol{A}$ and its runtime is the runtime of*

undirected edges $\{a, b\}$) then flow can be sent in one direction of the cycle without increasing $f^\top \vec{1}$ while preserving feasibility until at least one less edge has $f_e \notin \{0, 1\}$. Consequently, there is an optimal solution to (11) where the edges with $f_e \notin \{0, 1\}$ are acyclic (when viewed as undirected edges). However, by the constraint $\mathbf{B}^\top f = F \cdot \delta_{s,t}$ this implies that all edges have $f_e \in \{0, 1\}$ in this case. Consequently, there is an optimal integral flow f. (This holds more generally, see e.g. [DS, Sch03].) Further, if there is a directed cycle in G with a positive value of f_e on each edge, a feasible f with decreased $f^\top \vec{1}$ can be found by sending flow in the reverse of each cycle. Consequently, there is an optimal integral acyclic flow and again by the the constraints this implies that f corresponds to F disjoint paths from s to t.

$O(\sqrt{\kappa}\log(1/\epsilon))$ iterations each of which consist of applying $\boldsymbol{A}$ to a vector, invoking $\mathsf{Solve}_{\boldsymbol{B}}$*, and additional $O(n)$ time operations.*

The robustness of accelerated methods to error has been studied in a variety of contexts (see e.g. [LS, MS13, DGN14, BJL+]).We provide the proof in this section for completeness and to obtain a statement tailored to our particular setting. Limited attempts were made to optimize for the parameters and error tolerance for the preconditioner in the method.

We remark that this theorem is similar to one in [CKM+] which analyzed preconditioned Chebyshev iteration with bounded solving error. Interestingly, a similar result as to Theorem 4.4 can be achieved by analyzing the method of that paper with randomized error in the solver. Straightforward modification of their analysis yields a variant of Theorem 4.4 where Equation (1) in the definition of a solver is replaced with $\mathbb{E}\left\|x - \mathbf{A}^{\dagger}b\right\|_{\mathbf{A}} \leq \sqrt{\epsilon}\left\|b\right\|$ and the accuracy required for the solver for $\mathbf{B}$ scales with ϵ. We chose to provide the analysis of AGD as it provides an interesting alternative to preconditioned Chebyshev and naturally supported analysis of expected squared errors and preconditoners with accuracy that does not scale with ϵ.

In the remainder of this section we provide PreconditionedNoisyAGD (Algorithm 13) and prove Theorem 4.4. Our notation and analysis are similar to [HSS] and [CKM+] in places and specialized to our setting in others.

Algorithm 13: $F = \mathsf{PreconditionedNoisyAGD}(\mathbf{A}, b, \epsilon, \kappa, \mathsf{Solve}_{\mathbf{B}})$

Input: Symmetric PSD $\mathbf{A} \in \mathbb{R}^{n\times n}$, vector $b \in \mathbb{R}^n$, and accuracy $\epsilon \in (0,1)$
Input: Condition number bound κ, $\frac{1}{10\kappa}$-solver, $\mathsf{Solve}_{\mathbf{B}}$, for symmetric PSD $\mathbf{B} \in \mathbb{R}^{n\times n}$ with $\mathbf{A} \preceq \mathbf{B} \preceq \kappa\mathbf{A}$
Output: A vector such that algorithm is an ϵ-approximate solver for $\mathbf{A}$

$x_0 := 0 \in \mathbb{R}^n$, $v_0 := 0 \in \mathbb{R}^n$, and $T := \lceil 4\sqrt{\kappa}\log(2/\epsilon)\rceil$;
for $t = 0$ *to* $T-1$ **do**
  $y_t := \alpha x_t + (1-\alpha)v_t$ where $\alpha := \frac{2\sqrt{\kappa}}{1+2\sqrt{\kappa}}$;
  $x_{t+1} := y_t - g_t$ where $g_t := \mathsf{Solve}_{\mathbf{B}}(\mathbf{A}y_t - b)$;
  $v_{t+1} := \beta v_t + (1-\beta)\left[y_t - \eta g_t\right]$ where $\eta := 2\kappa$ and $\beta := 1 - \frac{1}{2\sqrt{\kappa}}$;
end
return x_t;

To analyze PreconditionedNoisyAGD (Algorithm 13) we first provide the following lemma for bounding the error from an approximate solve.

Lemma C.1. *Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ be symmetric PSD matrices with $\boldsymbol{A} \preceq \boldsymbol{B} \preceq \kappa\boldsymbol{A}$ and let $b \in \mathrm{im}(\boldsymbol{A})$. If $g = \mathsf{Solve}_{\boldsymbol{B}}(\boldsymbol{A}x - b)$ where $\mathsf{Solve}_{\boldsymbol{B}}$ is an ϵ-approximate solver for $\boldsymbol{B}$ then, $x_* := \boldsymbol{A}^{\dagger}b$ and*

$$\Delta := g - \boldsymbol{B}^{\dagger}(\boldsymbol{A}x - b) = g - \boldsymbol{B}^{\dagger}\boldsymbol{A}(x - x_*) \tag{13}$$

satisfies

$$\mathbb{E}\left\|\Delta\right\|_{\boldsymbol{B}}^2 \leq \epsilon\left\|x - x_*\right\|_{\boldsymbol{A}\boldsymbol{B}^{\dagger}\boldsymbol{A}}^2 \leq \epsilon\left\|x - x_*\right\|_{\boldsymbol{A}}^2 \text{ and } \mathbb{E}\left\|g\right\|_{\boldsymbol{B}}^2 \leq (1+\sqrt{\epsilon})^2\left\|x - x_*\right\|_{\boldsymbol{A}\boldsymbol{B}^{\dagger}\boldsymbol{A}}^2 . \tag{14}$$

Proof. Note that $b = \mathbf{A}x_*$ by the assumption that $b \in \mathrm{im}(\mathbf{A})$ thereby proving (13). Further, by definition of an ϵ-approximate solver and Δ we have

$$\mathbb{E}\left\|\Delta\right\|_{\mathbf{B}}^2 \leq \epsilon \left\|\mathbf{B}^\dagger(\mathbf{A}x - b)\right\|_{\mathbf{B}}^2 = \epsilon \left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 \leq \epsilon \left\|x - x_*\right\|_{\mathbf{A}}^2 \tag{15}$$

where in the last step we used that since $\mathbf{A} \preceq \mathbf{B}$ we have $\mathbf{B}^\dagger \preceq \mathbf{A}^\dagger$ and $\mathbf{A}\mathbf{B}^\dagger\mathbf{A} \preceq \mathbf{A}$. Further, since $\mathbb{E}\left\|\Delta\right\|_{\mathbf{B}} \leq \sqrt{\mathbb{E}\left\|\Delta\right\|_{\mathbf{B}}^2}$ by concavity of $\sqrt{\cdot}$ we have

$$\begin{aligned}
\mathbb{E}\left\|g\right\|_{\mathbf{B}}^2 &= \mathbb{E}\left\|\Delta + \mathbf{B}^\dagger\mathbf{A}(x - x_*)\right\|_{\mathbf{B}}^2 = \mathbb{E}\left[\left\|\Delta\right\|_{\mathbf{B}}^2 + 2\left[\Delta^\top\mathbf{B}\mathbf{B}^\dagger\mathbf{A}(x - x_*)\right] + \left\|\mathbf{B}^\dagger\mathbf{A}(x - x_*)\right\|_{\mathbf{B}}^2\right] \\
&\leq \mathbb{E}\left\|\Delta\right\|_{\mathbf{B}}^2 + 2\mathbb{E}\left\|\Delta\right\|_{\mathbf{B}}\left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}} + \left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 \\
&\leq \left[\epsilon + 2\sqrt{\epsilon} + 1\right]\left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2
\end{aligned}$$

where we used Cauchy Schwarz for Euclidean semi-norms, i.e. $a^\top\mathbf{B}b \leq \left\|a\right\|_{\mathbf{B}}\left\|b\right\|_{\mathbf{B}}$ and (15). □

Next we analyze the residual error decrease, i.e. change in $\left\|x - x_*\right\|_{\mathbf{A}}$, from taking a single gradient step using Solve. We will use this to analyze the error in computing x_{t+1} from y_t using PreconditionedNoisyAGD (Algorithm 13).

Lemma C.2. *Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ be symmetric PSD matrices with $\boldsymbol{A} \preceq \boldsymbol{B} \preceq \kappa\boldsymbol{A}$ and for arbitrary x and $b \in \mathrm{im}(\boldsymbol{A})$ let $y = x - g$ where $g = \mathrm{Solve}_{\boldsymbol{B}}(\boldsymbol{A}x - b)$ and $\mathrm{Solve}_{\boldsymbol{B}}$ is an ϵ-approximate linear system solver for $\boldsymbol{B}$. $x_* \coloneqq \boldsymbol{A}^\dagger b$ satisfies*

$$\mathbb{E}\left\|y - x_*\right\|_{\boldsymbol{A}}^2 \leq \left\|x - x_*\right\|_{\boldsymbol{A}}^2 - (1 - \epsilon)\left\|x - x_*\right\|_{\boldsymbol{A}\boldsymbol{B}^\dagger\boldsymbol{A}}^2 \leq \left(1 - \frac{1 - \epsilon}{\kappa}\right)\left\|x - x_*\right\|_{\boldsymbol{A}}^2 .$$

Proof. Let $x_* \coloneqq \mathbf{A}^\dagger b$ and $\Delta \coloneqq g - \mathbf{B}^\dagger(\mathbf{A}x - b)$. By Lemma C.1 we have

$$y = x - \left(\mathbf{B}^\dagger\mathbf{A}(x - x_*) + \Delta\right) \quad \text{and} \quad \mathbb{E}\left\|\Delta\right\|_{\mathbf{B}}^2 \leq \epsilon\left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 .$$

Consequently,

$$\left\|y - x_*\right\|_{\mathbf{A}}^2 = \left\|x - x_*\right\|_{\mathbf{A}}^2 - 2(x - x_*)^\top\mathbf{A}\left[\mathbf{B}^\dagger\mathbf{A}(x - x_*) + \Delta\right] + \left\|\mathbf{B}^\dagger\mathbf{A}(x - x_*) + \Delta\right\|_{\mathbf{A}}^2 .$$

Since $\mathbf{A} \preceq \mathbf{B}$,

$$\left\|\mathbf{B}^\dagger\mathbf{A}(x - x_*) + \Delta\right\|_{\mathbf{A}}^2 \leq \left\|\mathbf{B}^\dagger\mathbf{A}(x - x_*) + \Delta\right\|_{\mathbf{B}}^2 = \left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 + 2(x - x_*)^\top\mathbf{A}\Delta + \left\|\Delta\right\|_{\mathbf{B}}^2 .$$

Now $\mathbf{B} \preceq \kappa\mathbf{A}$ and therefore $\mathbf{B}^\dagger \succeq \kappa^{-1}\mathbf{A}^\dagger$ and $\mathbf{A}\mathbf{B}^\dagger\mathbf{A} \succeq \kappa^{-1}\mathbf{A}$. Combining the above inequalities yields

$$\begin{aligned}
\mathbb{E}\left\|y - x_*\right\|_{\mathbf{A}}^2 &\leq \left\|x - x_*\right\|_{\mathbf{A}}^2 - \left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 + \mathbb{E}\left\|\Delta\right\|_{\mathbf{B}}^2 \\
&\leq \left\|x - x_*\right\|_{\mathbf{A}}^2 - (1 - \epsilon)\left\|x - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 \leq \left(1 - \frac{1 - \epsilon}{\kappa}\right)\left\|x - x_*\right\|_{\mathbf{A}}^2 .
\end{aligned}$$

□

Lemma C.3 (Single Step Analysis)**.** *In the setting of Theorem 4.4 let $\epsilon_t \coloneqq \left\|x_t - x_*\right\|_{\boldsymbol{A}}^2$ and $r_t \coloneqq \left\|v_t - x_*\right\|_{\boldsymbol{B}}^2$ for $x_* \coloneqq \boldsymbol{A}^\dagger b$. Conditioned on the value of x_t and v_t and considering the randomness in g_t we have*

$$\mathbb{E}\left[\epsilon_{t+1} + \frac{1}{2\kappa}r_{t+1}\right] \leq \left(1 - \frac{1}{4\sqrt{\kappa}}\right)\left[\epsilon_t + \frac{1}{2\kappa}r_t\right] .$$

Proof. Throughout we let $z_t := \beta v_t + (1-\beta)y_t$, $\epsilon_t^y := \|y_t - x_*\|_{\mathbf{A}}^2$ and $r_t^y := \|y_t - x_*\|_{\mathbf{B}}^2$. Note that $\|z_t - x_*\|_{\mathbf{B}}^2 \le \beta r_t + (1-\beta) r_t^y$ by the convexity of $\|\cdot\|_{\mathbf{B}}^2$ and the definition of z_t . Consequently, expanding the definition of v_{t+1} and applying that $\eta(1-\beta) = \sqrt{\kappa}$ yields

$$\begin{aligned} r_{t+1} &= \|z_t - x_* - (1-\beta)\eta g_t\|_{\mathbf{B}}^2 \\ &= \|z_t - x_*\|_{\mathbf{B}}^2 + (1-\beta)\eta\left[-2g_t^\top \mathbf{B}(z_t - x_*) + \eta(1-\beta)\|g_t\|_{\mathbf{B}}^2\right] \\ &\le \beta r_t + (1-\beta) r_t^y + \sqrt{\kappa}\left[-2g_t^\top \mathbf{B}(z_t - x_*) + \sqrt{\kappa}\|g_t\|_{\mathbf{B}}^2\right] \end{aligned}$$

Now Lemma C.1 implies that

$$g_t = \mathbf{B}^\dagger \mathbf{A}(y_t - x_*) + \Delta_t \quad \text{and} \quad \mathbb{E}\|\Delta_t\|_{\mathbf{B}}^2 \le \epsilon \|y_t - x_*\|_{\mathbf{A}}^2 = \epsilon \cdot \epsilon_t^y ,$$

and the formulas for y_t and z_t imply that

$$z_t = \frac{\beta}{1-\alpha}(y_t - \alpha x_t) + (1-\beta)y_t = y_t + \frac{\alpha\beta}{1-\alpha}(y_t - x_t) .$$

Combining yields that

$$\begin{aligned} g_t^\top \mathbf{B}(z_t - x_*) &= \left(\mathbf{B}^\dagger \mathbf{A}(y_t - x_*)\right)^\top \mathbf{B}\left(y_t - x_* + \frac{\alpha\beta}{1-\alpha}(y_t - x_t)\right) + \Delta_t^\top \mathbf{B}(z_t - x_*) \\ &= \|y_t - x_*\|_{\mathbf{A}}^2 + \Delta_t^\top \mathbf{B}(z_t - x_*) + \frac{\alpha\beta}{1-\alpha}\left[(y_t - x_*)^\top \mathbf{A}(y_t - x_t)\right] . \end{aligned}$$

Further, since

$$\|x_t - x_*\|_{\mathbf{A}}^2 = \|y_t - x_*\|_{\mathbf{A}}^2 + 2(y_t - x_*)^\top \mathbf{A}(x_t - y_t) + \|y_t - x_t\|_{\mathbf{A}}^2 .$$

we have that

$$-2g_t^\top \mathbf{B}(z_t - x_*) \le -2\epsilon_t^y - 2\Delta_t^\top \mathbf{B}(z_t - x_*) + \frac{\beta\alpha}{1-\alpha}\left[\epsilon_t - \epsilon_t^y\right]$$

Now since $\mathbb{E}\|\Delta_t\|_{\mathbf{B}}^2 \le \epsilon \cdot \epsilon_t^y$, the concavity of $\sqrt{\cdot}$ yields

$$\begin{aligned} \mathbb{E}\left[-\Delta_t^\top \mathbf{B}(z_t - x_*)\right] &\le \mathbb{E}\|\Delta_t\|_{\mathbf{B}}\|z_t - x_*\|_{\mathbf{B}} \le \sqrt{\mathbb{E}\|\Delta_t\|_{\mathbf{B}}^2}\,\|z_t - x_*\|_{\mathbf{B}} \\ &\le \sqrt{\epsilon \cdot \epsilon_t^y}\left(\beta\|v_t - x_*\|_{\mathbf{B}} + (1-\beta)\|y_t - x_*\|_{\mathbf{B}}\right) \\ &\le \sqrt{\epsilon \cdot \epsilon_t^y r_t} + \sqrt{\epsilon\cdot\kappa}(1-\beta)\epsilon_t^y \\ &\le \frac{r_t}{2}\sqrt{\frac{\epsilon}{\kappa}} + \frac{\sqrt{\epsilon}}{2}\left(\sqrt{\kappa}+1\right)\epsilon_t^y \le \frac{r_t}{2}\sqrt{\frac{\epsilon}{\kappa}} + \frac{1}{4}\epsilon_t^y . \end{aligned}$$

where in the second to last line we used that $r_t^y \le \kappa\epsilon_t^y$ and $\beta \le 1$ and in the last line we used that $\sqrt{ab} \le \frac{a}{2p} + \frac{bp}{2}$ for all $a, b \in \mathbb{R}$ and $p > 0$, that $\sqrt{\kappa}(1-\beta) = \frac{1}{2}$, and $\sqrt{\epsilon}(\sqrt{\kappa}+1) \le \frac{1}{2}$. Combining, and again using that $r_t^y \le \kappa\epsilon_t^y$ yields

$$\begin{aligned} \mathbb{E}\left[\frac{(1-\beta)}{\sqrt{\kappa}}\cdot r_t^y - 2g_t^\top \mathbf{B}(z_t - x_*)\right] &\le \frac{\kappa}{2\kappa}\cdot\epsilon_t^y - 2\epsilon_t^y + \left[r_t\sqrt{\frac{\epsilon}{\kappa}} + \frac{1}{2}\epsilon_t^y\right] + \frac{\beta\alpha}{1-\alpha}[\epsilon_t - \epsilon_t^y] \\ &\le -\epsilon_t^y + \frac{\beta\alpha}{1-\alpha}[\epsilon_t - \epsilon_t^y] + r_t\sqrt{\frac{\epsilon}{\kappa}} . \end{aligned}$$

Further, by Lemma C.1 and Lemma C.2 and that $(1+\sqrt{\epsilon})^2/(1-\epsilon) \le 2$ for $\epsilon \le 1/10$

$$\mathbb{E}\left\|g_t\right\|_{\mathbf{B}}^2 \le (1+\sqrt{\epsilon})^2 \left\|y_t - x_*\right\|_{\mathbf{A}\mathbf{B}^\dagger\mathbf{A}}^2 \le \frac{(1+\sqrt{\epsilon})^2}{1-\epsilon}\left[\epsilon_t^y - \mathbb{E}\epsilon_{t+1}\right] \le 2[\epsilon_t^y - \mathbb{E}\epsilon_{t+1}]$$

Combining then yields that

$$\begin{aligned}\mathbb{E}r_{t+1} &\le \beta r_t + \sqrt{\kappa}\left[r_t\sqrt{\frac{\epsilon}{\kappa}} - \epsilon_t^y + \frac{\beta\alpha}{1-\alpha}[\epsilon_t - \epsilon_t^y] + 2\sqrt{\kappa}[\epsilon_t^y - \epsilon_{t+1}]\right] \\ &\le \left(\beta + \sqrt{\epsilon}\right) r_t + \sqrt{\kappa}\left[-\epsilon_t^y + 2\beta\sqrt{\kappa}[\epsilon_t - \epsilon_t^y] + 2\sqrt{\kappa}[\epsilon_t^y - \mathbb{E}\epsilon_{t+1}]\right] \\ &= \left(\beta + \sqrt{\epsilon}\right) r_t + 2\kappa\left[\beta\epsilon_t - \mathbb{E}\epsilon_{t+1}\right] \ :\end{aligned}$$

in the second line we used that α was chosen so that $\frac{\alpha}{1-\alpha} = 2\sqrt{\kappa}$, and in the third line we used that $\beta = 1 - \frac{1}{2\sqrt{\kappa}}$ implying $-1 - 2\beta\sqrt{\kappa} + 2\sqrt{\kappa} = 0$. Since $\beta \le \beta + \sqrt{\epsilon} \le 1 - \frac{1}{4\sqrt{\kappa}}$ rearranging yields the desired bound. □

Leveraging the preceding analysis we can now prove the theorem.

Proof of Theorem 4.4. Applying Lemma C.3 repeatedly we have that for $x_* = \mathbf{A}^\dagger b$

$$\mathbb{E}\left\|x_T - x_*\right\|_{\mathbf{A}}^2 \le \left(1 - \frac{1}{4\sqrt{\kappa}}\right)^T \left[\left\|x_0 - x_*\right\|_{\mathbf{A}}^2 + \frac{1}{2\kappa}\left\|x_0 - x_*\right\|_{\mathbf{B}}^2\right]$$

However, since $x_0 = 0$ and $\mathbf{B} \preceq \kappa\mathbf{A}$ we have

$$\left\|x_0 - x_*\right\|_{\mathbf{A}}^2 + \frac{1}{2\kappa}\left\|x_0 - x_*\right\|_{\mathbf{B}}^2 \le \frac{3}{2}\left\|\mathbf{A}^\dagger b\right\|_{\mathbf{A}}^2 = \frac{3}{2}\left\|b\right\|_{\mathbf{A}^\dagger}^2 \ .$$

Further, by choice of T we have

$$\left(1 - \frac{1}{4\sqrt{\kappa}}\right)^T \le \exp\left(\frac{-T}{4\sqrt{\kappa}}\right) \le \frac{\epsilon}{2} \ .$$

The result follows by combining these inequalities and applying the definition of an ϵ-solver and noticing that the iterations consist only of standard arithmetic operations of vector and applying $\text{Solve}_{\mathbf{B}}$ and applying $\mathbf{A}$ to a vector. □

D EFFECTIVE RESISTANCE FACTS

Here we give a variety of facts about effective resistance that we use throughout the paper. First, in the following claim we collect a variety of well known facts about effective resistance that we use throughout the paper and then we give additional technical lemmas we use throughout the paper.

Claim D.1 (Effective Resistance Properties)**.** *For any connected graph $G = (V, E)$ with positive edge weights $w \in \mathbb{R}^E$ and all $a, b, c \in V$ it is the case that*

- ***Flow Characterization**: $\mathcal{R}_G^{\text{eff}}(a,b) = \min_{\text{unit } a,b \text{ flow } f \in \mathbb{R}^E} \sum_{e\in E} f_e^2/w_e$.*
- ***Triangle Inequality**: $\mathcal{R}_G^{\text{eff}}(a,c) \le \mathcal{R}_G^{\text{eff}}(a,b) + \mathcal{R}_G^{\text{eff}}(b,c)$.*
- ***Monotonicity**: If H is an connected edge subgraph of G then $\mathcal{R}_G^{\text{eff}}(a,b) \le \mathcal{R}_H^{\text{eff}}(a,b)$.*

It is a well known fact that the effective resistance between two vertices s and t in a graph consisting k edge-disjoint parallel paths between s and t of length at most ℓ is ℓ/k. Here we give a slight generalization of this fact to bound the effective resistance of two vertices in low-depth trees connected by many short edge-disjoint paths.

Lemma D.2 (Effective Resistance in Well-connected Trees). *Let $G = (V, E, w)$ be a weighted unweighted graph which contains two edge disjoint trees $T_1, T_2 \subseteq E$ each of which has effective resistance diameter at most d, i.e. the effective resistance between any pair of vertices in a tree is at most d. Further suppose that there are at least k edge-disjoint paths between T_1 and T_2 each of which have effective resistance length at most ℓ, i.e. for path $P \subseteq E$ we have $\sum_{e\in P}(1/w_e) \leq \ell$. Then for all $a \in T_1$ and $b \in T_2$ we have $\mathcal{R}^{\text{eff}}_G(a, b) \leq 2d + \ell/k$.*

Proof. For each of the k edge-disjoint path $P_i \subseteq \mathbb{R}^E$ let f_i denote a flow that send unit from the path's start in T_1, denoted a_i, to the path's end in T_2, denoted b_i. Further, for all $i \in [k]$ let g_i denote the unique unit flow from a to a_i using only edges of T_1 and let h_i denote the unique unit flow from b_i to b using only edges of T_2. Note, that for all $i \in [k]$ we have that $r_i \coloneqq f_i + g_i + h_i$ is a unit a to b flow in G and consequently, $f_* \coloneqq \frac{1}{k}\sum_{i\in[k]} r_i$, is a unit a to b flow in G.

Now, f_* restricted to T_1 is the unique flow f_1 on T_1 that sends one unit from a to the uniform distribution over the a_i. Since T_1 has effective resistance diameter at most d, by the flow characterization of effective resistance (Lemma D.2) we can decompose this flow into a distribution over paths of effective resistance length at most d, i.e. $f_1 = \sum_i \alpha_i t_i$ where each $\alpha_i \geq 0$, $\sum_i \alpha_i = 1$, and each t_i is a unit flow corresponding to a path P of effective resistance length at most d. Therefore, by convexity of x^2 we have

$$\begin{aligned}\sum_{e\in T_1} \frac{1}{w_e}[f_1]_e^2 &= \sum_{e\in T_1} \frac{1}{w_e}\left[\sum_i \alpha_i [t_i]_e\right]^2 \leq \sum_{e\in T_1}\frac{1}{w_e}\sum_i \alpha_i [t_i]_e^2 \\ &= \sum_i \alpha_i \sum_{e\in T_1}\frac{1}{w_e}[t_i]_e^2 \leq \sum_i \alpha d = d\,.\end{aligned}$$

By symmetric reasoning, f_* restricted to T_2, denoted f_2 has $\sum_{e\in T_2}[f_2]_e^2/w_e \leq d$.

Since T_1, T_2, and the P_i are edge disjoint, are the only edges with non-zero flow, and have effective resistance length at most ℓ we have

$$\begin{aligned}\sum_{e\in E}\frac{1}{w_e}[f_*]_e^2 &= \sum_{e\in T_1}\frac{1}{w_e}[f_1]_e^2 + \sum_{e\in T_2}\frac{1}{w_e}[f_2]_e^2 + \sum_{i\in[k]}\sum_{e\in P_i}\frac{1}{w_e}[f_*]_e^2 \\ &\leq 2d + \sum_{i\in[k]}\sum_{e\in P_i}\frac{1}{w_e}\cdot\frac{1}{k^2} \leq 2d + \ell/k\,.\end{aligned}$$

The result follows as $\mathcal{R}^{\text{eff}}_G(a, b) \leq \sum_{e\in E}\frac{1}{w_e}[f_*]_e^2$ by the flow characterization of effective resistances, Claim D.1. □