Approximate Gaussian Elimination for Laplacians – Fast, Sparse, and Simple

Rasmus Kyng, Sushant Sachdeva
Department of Computer Science
Yale University
New Haven, USA
rasmus.kyng@yale.edu, sachdeva@cs.yale.edu

***Abstract*—We show how to perform sparse approximate Gaussian elimination for Laplacian matrices. We present a simple, nearly linear time algorithm that approximates a Laplacian by the product of a sparse lower triangular matrix with its transpose. This gives the first nearly linear time solver for Laplacian systems that is based purely on random sampling, and does not use any graph theoretic constructions such as low-stretch trees, sparsifiers, or expanders. Our algorithm performs a subsampled Cholesky factorization, which we analyze using matrix martingales. As part of the analysis, we give a proof of a concentration inequality for matrix martingales where the differences are sums of conditionally independent variables.**[1]

***Keywords*-Gaussian elimination; Cholesky factorization; Laplacian systems; Linear system solvers; Randomized numerical linear algebra; Matrix martingales**

I. Introduction

A symmetric matrix L is called Symmetric and Diagonally Dominant (SDD) if for all i, $L(i,i) \geq \sum_{j \neq i} |L(i,j)|$. An SDD matrix L is a Laplacian if $L(i,j) \leq 0$ for $i \neq j$, and for all i, $\sum_j L(i,j) = 0$. A Laplacian matrix is naturally associated with a graph on its vertices, where i, j are adjacent if $L(i,j) \neq 0$. The problem of solving systems of linear equations $Lx = b$, where L is an SDD matrix (and often a Laplacian), is a fundamental primitive and arises in varied applications in both theory and practice. Example applications include solutions of partial differential equations via the finite element method [1], [2], semi-supervised learning on graphs [3]–[5], and computing maximum flows in graphs [6]–[9]. It has also been used as a primitive in the design of several fast algorithms [10]–[14]. It is known that solving SDD linear systems can be reduced to solving Laplacian systems [15].

Cholesky Factorization: A natural approach to solving systems of linear equations is Gaussian elimination, or its variant for symmetric matrices, Cholesky factorization. Cholesky factorization of a matrix L produces a factorization $L = \mathcal{L}\mathcal{D}\mathcal{L}^\top$, where $\mathcal{L}$ is a lower-triangular matrix, and $\mathcal{D}$ is a diagonal matrix. Such a factorization allows us to solve a system $Lx = b$ by computing $x = L^{-1}b = (\mathcal{L}^{-1})^\top \mathcal{D}^{-1} \mathcal{L}^{-1} b$, where the inverse of $\mathcal{L}$, and $\mathcal{D}$ can be applied quickly since they are lower-triangular and diagonal, respectively.

The fundamental obstacle to using Cholesky factorization for quickly solving systems of linear equations is that $\mathcal{L}$ can be a dense matrix even if the original matrix L is sparse. The reason is that the key step in Cholesky factorization, eliminating a variable, say x_i, from a system of equations, creates a new coefficient $L'(j,k)$ for every pair j, k such that $L(j,i)$ and $L(i,k)$ are non-zero. This phenomenon is called *fill-in*. For Laplacian systems, eliminating the first variable corresponds to eliminating the first vertex in the graph, and the fill-in corresponds to adding a clique on all the neighbors of the first vertex. Sequentially eliminating variables often produces a sequence of increasingly-dense systems, resulting in an $O(n^3)$ worst-case time even for sparse L. Informally, the algorithm for generating the Cholesky factorization for a Laplacian can be expressed as follows:

```
1: for $i = 1$ to $n - 1$ do
2:     Use equation $i$ to express the variable for vertex $i$
       in terms of the remaining variables.
3:     Eliminate vertex $i$, adding a clique on the neighbors
       of $i$.
4: end for
```

Eliminating the vertices in an order given by a permutation π generates a factorization $L = P_\pi \mathcal{L}\mathcal{D}\mathcal{L}^\top P_\pi^\top$, where P_π denotes the permutation matrix of π, i.e., $(P_\pi z)_i = z_{\pi(i)}$ for all z. Though picking a good order of elimination can significantly reduce the running time of Cholesky factorization, it gives no guarantees for general systems, e.g., for sparse expander graphs, every ordering results in an $\Omega(n^3)$ running time [16].

Our Results: In this paper, we present the first nearly linear time algorithm that generates a sparse approximate Cholesky decomposition for Laplacian matrices, with provable approximation guarantees. Our algorithm SparseCholesky can be described informally as follows (see Section III for a precise description):

[1]An extended version of this paper is available on Arxiv at http://arxiv.org/abs/1605.02353

```
1: Randomly permute the vertices.
2: for $i = 1$ to $n - 1$ do
3:     Use equation $i$ to express the variable for vertex $i$ in terms of the remaining variables.
4:     Eliminate vertex $i$, adding random samples from the clique on the neighbors of $i$.
5: end for
```

We prove the following theorem about our algorithm, where for symmetric matrices A, B, we write $A \preceq B$ if $B - A$ is positive semidefinite (PSD).

Theorem 1.1: The algorithm SPARSECHOLESKY, given an $n \times n$ Laplacian matrix L with m non-zero entries, runs in expected time $O(m \log^3 n)$ and computes a permutation π, a lower triangular matrix $\mathcal{L}$ with $O(m \log^3 n)$ non-zero entries, and a diagonal matrix $\mathcal{D}$ such that with probability $1 - \frac{1}{\mathrm{poly}(n)}$, we have

$$1/2 \cdot L \preceq Z \preceq 3/2 \cdot L,$$

where $Z = P_\pi \mathcal{L} \mathcal{D} \mathcal{L}^\top P_\pi^\top$, *i.e.*, Z has a sparse Cholesky factorization.

The sparse approximate Cholesky factorization for L given by Theorem 1.1 immediately implies fast solvers for Laplacian systems. We can use the simplest iterative method, called iterative refinement [17, Chapter 12] to solve the system $Lx = b$ as follows. We let,

$$x^{(0)} = 0, \qquad x^{(i+1)} = x^{(i)} - 1/2 \cdot Z^{+}(Lx^{(i)} - b),$$

where we use the Z^+, the pseudo-inverse of Z since Z has a kernel identical to L. Let $\mathbf{1}$ denote the all ones vector, and for any vector v, let $\|v\|_L \overset{\mathrm{def}}{=} \sqrt{v^\top L v}$.

Theorem 1.2: For all Laplacian systems $Lx = b$ with $\mathbf{1}^\top b = 0$, and all $\epsilon > 0$, using the sparse approximate Cholesky factorization Z given by Theorem 1.1, the above iterate for $t = 3 \log 1/\epsilon$ satisfies $\left\|x^{(t)} - L^+ b\right\|_L \leq \epsilon \left\|L^+ b\right\|_L$. We can compute such an $x^{(t)}$ in time $O(m \log^3 n \log 1/\epsilon)$.

In our opinion, this is the simplest nearly-linear time solver for Laplacian systems. Our algorithm only uses random sampling, and no graph-theoretic constructions, in contrast with all previous Laplacian solvers. A complete analysis is quite short (see extended version for missing proofs). We remark that there is a possibility that our analysis is not tight, and that the bounds can be improved by a stronger matrix concentration result.

Technical Contributions: There are several key ideas that are central to our result: The first is randomizing the order of elimination. At step i of the algorithm, if we eliminate a fixed vertex and sample the resulting clique, we do not know useful bounds on the sample variances that would allow us to prove concentration. Randomizing over the choice of vertex to eliminate allows us to bound the sample variance by roughly $1/n$ times the Laplacian at the previous step.

The second key idea is our approach to estimating effective resistances: A core element in all nearly linear time Laplacian solvers is a procedure for estimating effective resistances (or leverage scores) for edges in order to compute sampling probabilities. In previous solvers, these estimates are obtained using fairly involved procedures (e.g. low-stretch trees, ultrasparsifiers, or the subsampling procedure due to Cohen et al. [18]). In contrast, our solver starts with the crudest possible estimates of 1 for every edge, and then uses the triangle inequality for effective resistances (Lemma 5.1) to obtain estimates for the new edges generated. We show that these estimates suffice for constructing a nearly linear time Laplacian solver.

Finally, we develop concentration bounds for a class of matrix martingales that we call bags-of-dice martingales. The name is motivated by a simple scalar model: at each step, we pick a random bag of dice from a collection of bags (in the algorithm, this corresponds to picking a random vertex to eliminate), and then we independently roll each die in the selected bag (corresponding to drawing independent samples from the clique added). The concentration bound gives us a powerful tool for handling conditionally independent sums of variables. We give two proofs of the concentration result, one using Lieb's concavity theorem [19], and another using the Matrix Freedman inequality [20]. We defer a formal description of the martingales and the concentration bound to Section IV-B.

Comparison to other Laplacian solvers: Though the current best algorithm for solving a general $n \times n$ positive semidefinite linear system with m non-zero entries takes time $O(\min\{mn, n^{2.2373}\})$ [21], a breakthrough result by Spielman and Teng [22], [23] showed that linear systems in graph Laplacians could be solved in time $O(m \cdot \mathrm{poly}(\log n) \log 1/\epsilon)$. There has been a lot of progress over the past decade [24]–[29], and the current best running time is $O(m \log^{1/2} n \log 1/\epsilon)$ (up to polylog n factors) [27]. All of these algorithms have relied on graph-theoretic constructions – low-stretch trees [22], [24]–[27], graph sparsification [22], [24], [25], [27], [28], and explicit expander graphs [29].

In contrast, our algorithm requires no graph-theoretic construction, and is based purely on random sampling. Our result only uses two algebraic facts about Laplacian matrices:

1) They are closed under taking Schur complements, and
2) They satisfy the effective resistance triangle inequality (Lemma 5.1).

[29] presented the first nearly linear time solver for block Diagonally Dominant (bDD) systems – a generalization of SDD systems. If bDD matrices satisfy the effective resistance triangle inequality (we are unaware if they do), then the algorithm in the main body of this paper immediately applies to bDD systems, giving a sparse approximate block

Cholesky decomposition and a nearly linear time solver for bDD matrices.

In Appendix D (extended version), we sketch a near-linear time algorithm for computing a sparse approximate block Cholesky factorization for bDD matrices. It combines the approach of SPARSECHOLESKY with a recursive approach for estimating effective resistances, as in [29], using the subsampling procedure [18]. Though the algorithm is more involved than SPARSECHOLESKY, it runs in time $O(m\log^3 n + n\log^5 n)$, and produces a sparse approximate Cholesky decomposition with only $O(m\log^2 n + n\log^4 n)$ entries. The algorithm only uses that bDD matrices are closed under taking Schur complements, and that the Schur complements have a clique structure similar to Laplacians (see Section II).

Comparison to Incomplete Cholesky Factorization: A popular approach to tackling fill-in is *Incomplete Cholesky factorization*, where we throw away most of the new entries generated when eliminating variables. The hope is that the resulting factorization is still an approximation to the original matrix L, in which case such an approximate factorization can be used to quickly solve systems in L. Though variants of this approach are used often in practice, and we have approximation guarantees for some families of Laplacians [30]–[32], there are no known guarantees for general Laplacians to the best of our knowledge. Most variants of incomplete Cholesky in the literature are deterministic algorithms. A notable exception is a randomized rounding scheme proposed by Clarkson [33] that he experimentally showed performs well on some matrices.

II. PRELIMINARIES

Laplacians and Multi-Graphs.: We consider a connected undirected multi-graph $G = (V, E)$, with positive edges weights $w : E \to \mathbb{R}_+$. Let $n = |V|$ and $m = |E|$. We label vertices 1 through n, s.t. $V = \{1, \ldots, n\}$. Let $\boldsymbol{e}_i$ denote the i^{th} standard basis vector. Given an ordered pair of vertices (u, v), we define the pair-vector $b_{u,v} \in \mathbb{R}^n$ as $\boldsymbol{b}_{u,v} = \boldsymbol{e}_v - \boldsymbol{e}_u$. For a multi-edge e, with endpoints u, v (arbitrarily ordered), we define $\boldsymbol{b}_e = \boldsymbol{b}_{u,v}$. By assigning an arbitrary direction to each multi-edge of G we define the Laplacian of G as $L = \sum_{e\in E} w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$. Note that the Laplacian does not depend on the choice of direction for each edge. Given a single multi-edge e, we refer to $w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$ as the Laplacian of e.

A weighted multi-graph G is not uniquely defined by its Laplacian, since the Laplacian only depends on the sum of the weights of the multi-edges on each edge. We want to establish a one-to-one correspondence between a weighted multi-graph G and its Laplacian L, so from now on, we will consider every Laplacian to be maintained explicitly as a sum of Laplacians of multi-edges, and we will maintain this multi-edge decomposition as part of our algorithms.

Fact 2.1: If G is connected, then the kernel of L is the span of the vector $\mathbf{1}$.

Let L^+ denote the pseudo-inverse of L. Let $J \stackrel{\text{def}}{=} \mathbf{1}\mathbf{1}^\top$. Then, we define $\Pi \stackrel{\text{def}}{=} LL^+ = I - \frac{1}{n}J$.

Cholesky Factorization in Sum and Product Forms.: We now formally introduce Cholesky factorization. Rather than the usual perspective where we factor out lower triangular matrices at every step of the algorithm, we present an equivalent perspective where we subtract a rank one term from the current matrix, obtaining its Schur complement. The lower triangular structure follows from the fact that the matrix effectively become smaller at every step.

Let L be any symmetric positive-semidefinite matrix. Let $L(:, i)$ denote the i^{th} column of L. Using the first equation in the system $Lx = b$ to eliminate the variable x_1 produces another system $S^{(1)}x' = b'$, where $b'_1 = 0, x'$ is x with x_1 replaced by 0, and $S^{(1)} \stackrel{\text{def}}{=} L - (L(1,1))^{-1}L(:,1)L(:,1)^\top$, is called the *Schur complement* of L with respect to 1. The first row and column of $S^{(1)}$ are identically 0, and thus this is effectively a system in the remaining $n-1$ variables. Letting $\alpha_1 \stackrel{\text{def}}{=} L(v_1, v_1), \boldsymbol{c}_1 \stackrel{\text{def}}{=} \frac{1}{\alpha_1}L(:, v_1)$, we have $L = S^{(1)} + \alpha_1\boldsymbol{c}_1\boldsymbol{c}_1^\top$.

For computing the Cholesky factorization, we perform a sequence of such operations, where in the k^{th} step, we select an index $v_k \in V \setminus \{v_1, \ldots, v_{k-1}\}$ and eliminate the variable v_k. We define $\alpha_k = S^{(k-1)}(v_k, v_k)$, $\boldsymbol{c}_k = \alpha_k^{-1}S^{(k-1)}(:, v_k)$, and $S^{(k)} = S^{(k-1)} - \alpha_k\boldsymbol{c}_k\boldsymbol{c}_k^\top$. If at some step k, $S^{(k-1)}(v_k, v_k) = 0$, then we define $\alpha_k = 0$, and $\boldsymbol{c}_k = 0$. Continuing until $k = n-1$, $S^{(k)}$ will have at most one non-zero entry, which will be on the v_n diagonal. We define $\alpha_n = S^{(k)}$ and $\boldsymbol{c}_n = \boldsymbol{e}_{v_n}$. Let $\mathcal{C}$ be the $n\times n$ matrix with $\boldsymbol{c}_i$ as its i^{th} column, and $\mathcal{D}$ be the $n\times n$ diagonal matrix $\mathcal{D}(i,i) = \alpha_i$, then $L = \sum_{i=1}^n \alpha_i\boldsymbol{c}_i\boldsymbol{c}_i^\top = \mathcal{C}\mathcal{D}\mathcal{C}^\top$. Define the permutation matrix P by $P\boldsymbol{e}_i = \boldsymbol{e}_{v_i}$. Letting $\mathcal{L} = P^\top\mathcal{C}$, we have $L = P\mathcal{L}\mathcal{D}\mathcal{L}^\top P^\top$. This decomposition is known as Cholesky factorization. Crucially, $\mathcal{L}$ is lower triangular, since $\mathcal{L}(i,j) = (P^\top\boldsymbol{c}_j)(i) = \boldsymbol{c}_j(v_i)$, and for $i < j$, we have $\boldsymbol{c}_j(v_i) = 0$.

Clique Structure of the Schur Complement.: Given a Laplacian L, let $(L)_v \in \mathbb{R}^{n\times n}$ denote the Laplacian corresponding to the edges incident on vertex v, i.e.

$$(L)_v \stackrel{\text{def}}{=} \sum_{e\in E: e\ni v} w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top. \qquad (1)$$

For example, we denote the first column of L by $\begin{pmatrix} d \\ -\boldsymbol{a}\end{pmatrix}$, then $L_1 = \begin{bmatrix} d & -\boldsymbol{a}^\top \\ -\boldsymbol{a} & \mathbf{Diag}(\boldsymbol{a})\end{bmatrix}$. We can write the Schur complement $S^{(1)}$ as $S^{(1)} = L - (L)_v + (L)_v - (L(v,v))^{-1}L(:,v)L(:,v)^\top$. It is immediate that $L - (L)_v$ is a Laplacian matrix, since $L - (L)_v = \sum_{e\in E: e\not\ni v} w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$. A more surprising (but well-known) fact is that

$$C_v(L) \stackrel{\text{def}}{=} (L)_v - (L(v,v))^{-1} L(:,v) L(:,v)^\top \qquad (2)$$

is also a Laplacian, and its edges form a clique on the neighbors of v. It suffices to show it for $v = 1$. We write $i \sim j$ to denote $(i,j) \in E$. Then

$$C_1(L) = L_1 - (L(1,1))^{-1} L(:,1) L(:,1)^\top \qquad (3)$$

$$= \begin{bmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{Diag}(\boldsymbol{a}) - \frac{\boldsymbol{a}\boldsymbol{a}^\top}{d} \end{bmatrix} = \sum_{i\sim 1} \sum_{j \sim 1} \frac{w(1,i)w(1,j)}{d} \boldsymbol{b}_{(i,j)} \boldsymbol{b}_{(i,j)}^\top .$$

Thus $S^{(1)}$ is a Laplacian since it is a sum of two Laplacians. By induction, for all k, $S^{(k)}$ is a Laplacian.

III. The SparseCholesky Algorithm

Fig. 1 gives the pseudo-code for our algorithm SparseCholesky. Our main result, Theorem 3.1 (a more precise version of Theorem 1.1), shows that the algorithm computes an approximate sparse Cholesky decomposition in nearly linear time. We assume the Real RAM model. We prove the theorem in Section IV.

Theorem 3.1: Given a connected undirected multi-graph $G = (V,E)$, with positive edges weights $w : E \to \mathbb{R}_+$, and associated Laplacian L, and scalars $\delta > 1$, $0 < \epsilon \leq 1/2$, the algorithm SparseCholesky(L, ϵ, δ) returns an approximate sparse Cholesky decomposition $(P, \mathcal{L}, \mathcal{D})$ s.t. with probability at least $1 - 2/n^\delta$,

$$(1-\epsilon)L \preceq P\mathcal{L}\mathcal{D}\mathcal{L}^\top P^\top \preceq (1+\epsilon)L. \qquad (4)$$

For all $c > 1$, the maximum number of non-zero entries in $\mathcal{L}$ and the total running time are both bounded by $O(c\frac{\delta^2}{\epsilon^2} m \log^3 n)$ with probability $1 - 1/n^c$.
Fig. 2 gives the pseudo-code for our CliqueSample algorithm.

The most significant obstacle to making Cholesky factorization of Laplacians efficient is the *fill-in* phenomenon, namely that each clique $C_v(S)$ has roughly $(\deg_S(v))^2$ non-zero entries. To solve this problem, we develop a sampling procedure CliqueSample that produces a sparse Laplacian matrix which approximates the clique $C_v(S)$. As input, the procedure requires a Laplacian matrix S, maintained as a sum of Laplacians of multi-edges, and a vertex v. It then computes a sampled matrix that approximates $C_v(S)$. The elimination step in SparseCholesky removes the $\deg_S(v)$ edges incident on v, and CliqueSample(S, v) only adds at most $\deg_S(v)$ multi-edges. This means the total number of multi-edges does not increase with each elimination step, solving the *fill-in* problem. The sampling procedure is also very fast: It takes $O(\deg_S(v))$ time, much faster than the order $(\deg_S(v))^2$ time required to even write down the clique $C_v(S)$.

Although it is notationally convenient for us to pass the whole matrix S to CliqueSample, the procedure only relies on multi-edges incident on v, so we will only pass these multi-edges.

Remark 3.2: The algorithm SparseCholesky can be made slightly faster by performing sparsification after eliminating $n/\text{poly}\log n$ of the vertices. This gives a running time and sparsity of $O(\frac{\delta^2}{\epsilon^2} m \log^2 n \log\log n)$ w.h.p. In Appendix C (extended version) we sketch a proof of this.

Remark 3.3: The algorithm SparseCholesky can be modified so that it applies to bDD matrices. In Appendix D (extended version) we sketch a proof of this.

IV. Analysis of the Algorithm using Matrix Concentration

In this section, we analyze the SparseCholesky algorithm, and prove Theorem 3.1. To prove the theorem, we need several intermediate results which we will now present. In Section IV-A, we show how the output the SparseCholesky and CliqueSample algorithms can be used to define a matrix martingale. In Section IV-B, we introduce a new type of martingale, called a bags-of-dice martingale, and a novel matrix concentration result for these martingales. In Section IV-C, we show how to apply our new matrix concentration results to the SparseCholesky martingale and prove Theorem 3.1. We defer proofs of the lemmas that characterize CliqueSample to Section V, and proofs of our matrix concentration results to Section 6 (extended version).

A. Clique Sampling and Martingales for Cholesky Factorization

Throughout this section, we will study matrices that arise in the when using SparseCholesky to produce a sparse approximate Cholesky factorization of the Laplacian L of a multi-graph G. We will very frequently need to refer to matrices that are normalized by L. We adopt the following notation: Given a symmetric matrix S s.t. $\ker(L) \subseteq \ker(S)$,

$$\overline{S} \stackrel{\text{def}}{=} (L^+)^{1/2} S (L^+)^{1/2}.$$

We will only use this notation for matrices S that satisfy the condition $\ker(L) \subseteq \ker(S)$. Note that $\overline{L} = \Pi$, and $A \preceq B$ iff $\overline{A} \preceq \overline{B}$. Normalization is always done with respect to the Laplacian L input to SparseCholesky. We say a multi-edge e is $1/\rho$-bounded if

$$\left\| w(e) \overline{\boldsymbol{b}_e \boldsymbol{b}_e^\top} \right\| \leq 1/\rho.$$

Given a Laplacian S that corresponds to a multi-graph G_S, and a scalar $\rho > 0$, we say that S is $1/\rho$-bounded if every multi-edge of S is $1/\rho$-bounded. Since every multi-edge of L is trivially 1-bounded, we can obtain a $1/\rho$-bounded Laplacian that corresponds to the same matrix, by splitting each multi-edge into $\lceil \rho \rceil$ identical copies, with a fraction $1/\lceil \rho \rceil$ of the initial weight. The resulting Laplacian has at most $\lceil \rho \rceil m$ multi-edges.

```
1:  $\widehat{S}^{(0)} \leftarrow L$ with edges split into $\rho = \left\lceil 12(1+\delta)^2\epsilon^{-2}\ln^2 n \right\rceil$ copies with $1/\rho$ of the original weight
2:  Define the diagonal matrix $\mathcal{D} \leftarrow \mathbf{0}_{n\times n}$
3:  Let $\pi$ be a uniformly random permutation. Let $P_\pi$ be its permutation matrix, i.e., $(P_\pi x)_i = x_{\pi_i}$
4:  for $k = 1$ to $n-1$ do
5:      $\mathcal{D}(\pi(k),\pi(k)) \leftarrow (\pi(k),\pi(k))$ entry of $\widehat{S}^{(k-1)}$
6:      $\boldsymbol{c}_k \leftarrow \pi(k)^{\text{th}}$ column of $\widehat{S}^{(k-1)}$ divided by $\mathcal{D}(\pi(k),\pi(k))$ if $\mathcal{D}(\pi(k),\pi(k)) \neq 0$, or zero otherwise
7:      $\widehat{C}_k \leftarrow \text{CLIQUESAMPLE}(\widehat{S}^{(k-1)},\pi(k))$
8:      $\widehat{S}^{(k)} \leftarrow \widehat{S}^{(k-1)} - \left(\widehat{S}^{(k-1)}\right)_{\pi(k)} + \widehat{C}_k$
9:  end for
10: $\mathcal{D}(\pi(n),\pi(n)) \leftarrow \widehat{S}^{(n)}$ and $\boldsymbol{c}_n \leftarrow \boldsymbol{e}_{\pi(n)}$
11: $\mathcal{L} \leftarrow P_\pi^\top \begin{pmatrix}\boldsymbol{c}_1 & \boldsymbol{c}_2 & \dots & \boldsymbol{c}_n\end{pmatrix}$
12: return $(P_\pi,\mathcal{L},\mathcal{D})$
```

Figure 1: SPARSECHOLESKY(ϵ, L) : Given an $\epsilon > 0$ and a Laplacian L, outputs $(P,\mathcal{L},\mathcal{D})$, a sparse approximate Cholesky factorization of L

```
1:  for $i \leftarrow 1$ to $\deg_S(v)$ do
2:      Sample $e_1$ from list of multi-edges incident on $v$
        with probability $w(e)/w_S(v)$.
3:      Sample $e_2$ uniformly from list of multi-edges inci-
        dent on $v$.
4:      if $e_1$ has endpoints $v, u_1$
5:        and $e_2$ has endpoints $v, u_2$ and $u_1 \neq u_2$
6:      then
7:          $Y_i \leftarrow \frac{w(e_1)w(e_2)}{w(e_1)+w(e_2)} \boldsymbol{b}_{u_1,u_2}\boldsymbol{b}_{u_1,u_2}^\top$
8:      else
9:          $Y_i \leftarrow 0$
10:     end if
11: end for
12: return $\sum_i Y_i$
```

Figure 2: $\sum_i Y_i =$ CLIQUESAMPLE(S, v) : Returns several i.i.d samples of edges from the clique generated after eliminating vertex v from the multi-graph represented by S

Our next lemma describes some basic properties of the samples output by CLIQUESAMPLE. We prove the lemma in Section V.

Lemma 4.1: Given a Laplacian matrix S that is $1/\rho$-bounded w.r.t. L and a vertex v, CLIQUESAMPLE(S, v) returns a sum $\sum_e Y_e$ of $\deg_S(v)$ IID samples $Y_e \in \mathbb{R}^{n\times n}$. The following conditions hold

1) Y_e is 0 or the Laplacian of a multi-edge with endpoints u_1, u_2, where u_1, u_2 are neighbors of v in S.
2) $\mathbb{E}\sum_e Y_e = C_v(S)$.
3) $\left\|\overline{Y}_e\right\| \leq 1/\rho$, i.e. Y_e is $1/\rho$-bounded w.r.t. L.

The algorithm runs in time $O(\deg_S(v))$.
The lemma tells us that the samples in expectation behave like the clique $C_v(S)$, and that each sample is $1/\rho$-bounded. This will be crucial to proving concentration properties of our algorithm. We will use the fact that the expectation of the CLIQUESAMPLE algorithm output equals the matrix produced by standard Cholesky elimination, to show that in expectation, the sparse approximate Cholesky decomposition produced by our SPARSECHOLESKY algorithm equals the original Laplacian. We will also see how we can use this expected behaviour to represent our sampling process as a martingale. We define the kth approximate Laplacian as

$$L^{(k)} = \widehat{S}^{(k)} + \sum_{i=1}^{k} \alpha_i \boldsymbol{c}_i \boldsymbol{c}_i^\top. \tag{5}$$

Thus our final output equals $L^{(n)}$. Note that Line (10) of the SPARSECHOLESKY algorithm does not introduce any sampling error, and so $L^{(n)} = L^{(n-1)}$. The only significance of Line (10) is that it puts the matrix in the form we need for our factorization. Now

$$\begin{aligned} L^{(k)} - L^{(k-1)} &= \alpha_k \boldsymbol{c}_k \boldsymbol{c}_k^\top + \widehat{S}^{(k)} - \widehat{S}^{(k-1)} \\ &= \alpha_k \boldsymbol{c}_k \boldsymbol{c}_k^\top + \widehat{C}_k - \left(\widehat{S}^{(k-1)}\right)_{\pi(k)} \\ &= \widehat{C}_k - C_{\pi(k)}(\widehat{S}^{(k-1)}). \end{aligned}$$

Each call to CLIQUESAMPLE returns a sum of sample edges. Letting $Y_e^{(k)}$ denote the eth sample in the kth call to CLIQUESAMPLE, we can write this sum as $\sum_e Y_e^{(k)}$. Thus, conditional on the choices of the SPARSECHOLESKY algorithm until step $k-1$, and conditional on $\pi(k)$, we can apply Lemma 4.1 to find that the expected value of $\widehat{C}_k$ is $\sum_e \mathbb{E}_{Y_e^{(k)}} Y_e^{(k)} = C_{\pi(k)}(\widehat{S}^{(k-1)})$. Hence the expected value of $L^{(k)}$ is exactly $L^{(k-1)}$, and we can write

$$L^{(k)} - L^{(k-1)} = \sum_e Y_e^{(k)} - \mathop{\mathbb{E}}_{Y_e^{(k)}} Y_e^{(k)}.$$

By defining $X_e^{(k)} \overset{\text{def}}{=} Y_e^{(k)} - \mathbb{E}_{Y_e^{(k)}} Y_e^{(k)}$, this becomes

$$L^{(k)} - L^{(k-1)} = \sum_e X_e^{(k)}.$$

So, without conditioning on the choices of the SPARSECHOLESKY algorithm, we can write

$$L^{(n)} - L = L^{(n)} - L^{(0)} = \sum_{i=1}^{n-1} \sum_{e} X_e^{(i)}.$$

This is a martingale. To prove multiplicative concentration bounds, we need to normalize the martingale by L, and so instead we consider

$$\overline{L^{(n)}} - \overline{L} = \overline{L^{(n-1)}} - \overline{L} = \overline{L^{(n)}} - \Pi = \sum_{i=1}^{n-1} \sum_{e} \overline{X_e^{(i)}}. \quad (6)$$

This martingale has considerable structure beyond a standard martingale. Conditional on the choices of the SPARSECHOLESKY algorithm until step $k-1$, and conditional on $\pi(k)$, the terms $\overline{X_e^{(k)}}$ are independent.

In Section IV-B we define a type of martingale that formalizes the important aspects of this structure.

B. Bags-of-Dice Martingales and Matrix Concentration Results

We use the following shorthand notation: Given a sequence of random variables $(r_1, R^{(1)}, r_2, R^{(2)}, \ldots, r_k, R^{(k)})$, for every i, we write

$$\mathop{\mathbb{E}}_{(i)} [\,\cdot\,] = \mathop{\mathbb{E}}_{r_1} \mathop{\mathbb{E}}_{R^{(1)}} \cdots \mathop{\mathbb{E}}_{r_i} \mathop{\mathbb{E}}_{R^{(i)}} [\,\cdot\,].$$

Extending this notation to conditional expectations, we write,

$$\mathop{\mathbb{E}}_{r_i} \left[\,\cdot\, \middle| (i-1) \right] = \mathop{\mathbb{E}}_{r_i} \left[\,\cdot\, \middle| r_1, R^{(1)}, \ldots, r_{i-1}, R^{(i-1)} \right]$$

Definition 4.2: **A bags-of-dice martingale** is a sum of random $d \times d$ matrices $Z = \sum_{i=1}^{k} \sum_{e=1}^{l_i} Z_e^{(i)}$, with some additional structure. We require that there exists a sequence of random variables $(r_1, R^{(1)}, r_2, R^{(2)}, \ldots, r_k, R^{(k)})$, s.t. for all $1 \le i \le k$, conditional on $(i-1)$ and r_i, l_i is a non-negative integer, and $R^{(i)}$ is a tuple of l_i independent random variables: $R^{(i)} = (R_1^{(i)}, \ldots, R_{l_i}^{(i)})$. We also require that conditional on $(r_1, R^{(1)}, r_2, R^{(2)}, \ldots, r_i, R^{(i)})$ and r_i, for all $1 \le e \le l_i$ $Z_e^{(i)}$ is a symmetric matrix, and a deterministic function of $R_e^{(i)}$. Finally, we require that $\mathbb{E}_{R_e^{(i)}} \left[Z_e^{(i)} \middle| (i-1), r_i \right] = 0$.
Note that l_i is allowed to be random, as long as it is fixed conditional on $(i-1)$ and r_i. The martingale given in Equation (6) is a bags-of-dice martingale, with $r_i = \pi(i)$, and $R_e^{(i)} = Y_e^{(i)}$. The name is motivated by a simple model: At each step of the martingale we pick a random bag of dice from a collection of bags (this corresponds to the outcome of r_i) and then we independently roll each die in the bag (corresponds to the outcomes $R_e^{(i)}$).

It is instructive to compare the bags-of-dice martingales with matrix martingales such as those considered by Tropp [34]. A naive application of the results from [34] tells us that if we have good uniform norm bounds on each term $Z_e^{(i)}$, and there exists fixed matrices $\Omega_e^{(i)}$ s.t. that for all i, e and for all possible outcomes we get deterministic bounds on the matrix variances: $\mathbb{E}_{R_e^{(i)}} (Z_e^{(i)})^2 \preceq \Omega_e^{(i)}$, then the concentration of the martingale is governed by the norm of the sum of the variances $\left\| \sum_i \sum_e \Omega_e^{(i)} \right\|$. In our case, this result is much too weak: Good enough $\Omega_e^{(i)}$ do not exist.

A slight extension of the results from [34] allows us to do a somewhat better: Since the terms $Z_e^{(i)}$ are independent conditional on r_i, it suffices to produce fixed matrices $\Omega^{(i)}$ s.t. that for all i, e and for all possible outcomes we get deterministic bounds on the matrix variances of the sum of variables in each "bag": $\sum_e \mathbb{E}_{R_e^{(i)}} (Z_e^{(i)})^2 \preceq \Omega^{(i)}$. Then the concentration of the martingale is governed by $\left\| \sum_i \Omega^{(i)} \right\|$. Again, this result is not strong enough: Good fixed $\Omega^{(i)}$ do not seem to exist.

We show a stronger result: If we can produce a uniform norm bound on $\mathbb{E}_{R_e^{(i)}} (Z_e^{(i)})^2$, then it suffices to produce fixed matrices $\widehat{\Omega}^{(i)}$ that upper bound the matrix variance *averaged over all bags*: $\mathbb{E}_{r_i} \sum_e \mathbb{E}_{R_e^{(i)}} (Z_e^{(i)})^2 \preceq \widehat{\Omega}^{(i)}$. The concentration of the martingale is then governed by $\left\| \sum_i \widehat{\Omega}^{(i)} \right\|$. In the context where we apply our concentration result, our estimates of $\left\| \sum_i \Omega^{(i)} \right\|$ are larger than $\left\| \sum_i \widehat{\Omega}^{(i)} \right\|$ by a factor $\approx \frac{n}{\ln n}$. Consequently, we would not obtain strong enough concentration using the weaker result. The precise statement we prove is:

Theorem 4.3: Suppose $Z = \sum_{i=1}^{k} \sum_{e=1}^{l_i} Z_e^{(i)}$ is a bags-of-dice martingale of $d \times d$ matrices that satisfies

1) Every sample $Z_e^{(i)}$ satisfies $\left\| Z_e^{(i)} \right\|^2 \le \sigma_1^2$.
2) For every i we have
$$\left\| \sum_e \mathop{\mathbb{E}}_{R_e^{(i)}} \left[(Z_e^{(i)})^2 \middle| (i-1), r_i \right] \right\| \le \sigma_2^2.$$
3) There exist deterministic matrices Ω_i such that $\mathbb{E}_{r_i} \left[\sum_e \mathbb{E}_{R_e^{(i)}} (Z_e^{(i)})^2 \middle| (i-1) \right] \preceq \Omega_i$, and $\left\| \sum_i \Omega_i \right\| \le \sigma_3^2$.

Then, for all $\epsilon > 0$, we have

$$\Pr\left[Z \not\preceq \epsilon I \right] \le d \exp\left(-\epsilon^2 / 4\sigma^2 \right),$$

where

$$\sigma^2 = \max\left\{ \sigma_3^2, \frac{\epsilon}{2}\sigma_1, \frac{4\epsilon}{5}\sigma_2 \right\}.$$

This theorem and all the results in this section extend immediately to Hermitian matrices. We prove the above theorem in Section 6 (extended version). This result is based on the techniques introduced by Tropp [34] for using Lieb's Concavity Theorem to prove matrix concentration results. Tropp's result improved on earlier work, such as Ahlswede and Winter [35] and Rudelson and Vershynin [36]. These earlier matrix concentration results required IID sampling, making them unsuitable for our purposes.

Remark 4.4: Joel Tropp brought to our attention that Theorem 4.3 can be proven using his Matrix Freedman result [20]. Using this result, we give a proof of Theorem 4.3 in Appendix B (extended version).

Unfortunately, we cannot apply Theorem 4.3 directly to the bags-of-dice martingale in Equation (6). As we will see later, the variance of $\sum_e \overline{X_e^{(i)}}$ can have norm proportional to $\left\|\overline{L^{(i)}}\right\|$, which can grow large. However, we expect that the probability of $\left\|\overline{L^{(i)}}\right\|$ growing large is very small. Our next construction allows us to leverage this idea, and avoid the small probability tail events that prevent us from directly applying Theorem 4.3 to the bags-of-dice martingale in Equation (6).

Definition 4.5: Given a bags-of-dice martingale of $d \times d$ matrices $Z = \sum_{i=1}^{k} \sum_{e=1}^{l_i} Z_e^{(i)}$, and a scalar $\epsilon > 0$, we define for each $h \in \{1, 2, \ldots, k+1\}$ the event

$$\mathcal{A}_h = \left[\forall 1 \leq j < h. \sum_{i=1}^{j} \sum_{e=1}^{l_i} Z_e^{(i)} \preceq \epsilon I\right].$$

We also define the **ϵ-truncated martingale**:

$$\widetilde{Z} = \sum_{i=1}^{k} \left(\mathbb{1}_{\mathcal{A}_i} \sum_{e=1}^{l_i} Z_e^{(i)}\right)$$

The truncated martingale is derived from another martingale by forcing the martingale to get "stuck" if it grows too large. This ensures that so long as the martingale is not stuck, it is not too large. On the other hand, as our next result shows, the truncated martingale fails more often than the original martingale, and so it suffices to prove concentration of the truncated martingale to prove concentration of the original martingale. The theorem stated below is proven in Section 6 (extended version).

Theorem 4.6: Given a bags-of-dice martingale of $d \times d$ matrices $Z = \sum_{i=1}^{k} \sum_{e=1}^{l_i} Z_e^{(i)}$, a scalar $\epsilon > 0$, the associated ϵ-truncated martingale $\widetilde{Z}$ is also a bags-of-dice martingale, and

$$\Pr[-\epsilon I \not\preceq Z \text{ or } Z \not\preceq \epsilon I] \leq \Pr[-\epsilon I \not\preceq \widetilde{Z} \text{ or } \widetilde{Z} \not\preceq \epsilon I]$$

C. Analyzing the SPARSECHOLESKY *Algorithm Using Bags-of-Dice Martingales*

By taking $Z_e^{(k)} = \overline{X_e^{(k)}}$, $r_i = \pi(i)$ and $R_e^{(i)} = Y_e^{(i)}$, we obtain a bags-of-dice martingale $Z = \sum_{i=1}^{n-1} \sum_e Z_e^{(i)}$. Let $\widetilde{Z}$ denote the corresponding ϵ-truncated bags-of-dice martingale. The next lemma shows that $\widetilde{Z}$ is well-behaved. The lemma is proven in Section V.

Lemma 4.7: Given an integer $1 \leq k \leq n-1$, conditional on the choices of the SPARSECHOLESKY algorithm until step $k-1$, let $\sum_e Y_e = \text{CLIQUESAMPLE}(\widehat{S}^{(k-1)}, \pi(k))$. Let $X_e \stackrel{\text{def}}{=} Y_e - \mathbb{E}_{Y_e} Y_e$. The following statements hold

1) Conditional on $\pi(k)$, $\mathbb{E}_{Y_e} \mathbb{1}_{\mathcal{A}_k} \overline{X_e} = 0$.
2) $\left\|\mathbb{1}_{\mathcal{A}_k} \overline{X}_e\right\| \leq 1/\rho$ holds always.
3) Conditional on $\pi(k)$,
$$\sum_e \mathbb{E}_{Y_e} (\mathbb{1}_{\mathcal{A}_k} \overline{X}_e)^2 \preceq \mathbb{1}_{\mathcal{A}_k} \frac{1}{\rho} \overline{C_v(S)}.$$
4) $\left\|\mathbb{1}_{\mathcal{A}_k} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})}\right\| \leq 1 + \epsilon$ holds always.
5) $\mathbb{E}_{\pi(k)} \mathbb{1}_{\mathcal{A}_k} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})} \preceq \frac{2(1+\epsilon)}{n+1-k} I$.

We are now ready to prove Theorem 3.1.

Proof of Theorem 3.1: We have $\overline{L^{(n)}} = \Pi + Z$. Since for all k, e, $\ker(L) \subseteq \ker(Y_e^{(k)})$, the statement $(1-\epsilon)L \preceq L^{(n)} \preceq (1+\epsilon)L$ is equivalent to $-\epsilon\Pi \preceq Z \preceq \epsilon\Pi$. Further, $\Pi Z \Pi = Z$, and so it is equivalent to $-\epsilon I \preceq Z \preceq \epsilon I$. By Theorem 4.6, we have,

$$\begin{aligned}\Pr[(1-\epsilon)L \preceq L^{(n)} \preceq (1+\epsilon)L] &= 1 - \Pr\left[-Z \not\preceq \epsilon I \text{ or } Z \not\preceq \epsilon I\right] \\ &\geq 1 - \Pr\left[-\widetilde{Z} \not\preceq \epsilon I \text{ or } \widetilde{Z} \not\preceq \epsilon I\right] \\ &\geq 1 - \Pr\left[-\widetilde{Z} \not\preceq \epsilon I\right] - \Pr\left[\widetilde{Z} \not\preceq \epsilon I\right]\end{aligned} \tag{7}$$

To lower bound this probability, we'll prove concentration using Theorem 4.3. We now compute the parameters for applying the theorem. From Lemma 4.1, for all k and e, we have $\mathbb{E}_{Y_e} \mathbb{1}_{\mathcal{A}_k} \overline{X_e^{(k)}} = 0$, $\left\|\mathbb{1}_{\mathcal{A}_k} \overline{X_e^{(k)}}\right\| \leq \frac{1}{\rho}$. Thus, we can pick $\sigma_1 = \frac{1}{\rho}$. Next, again by Lemma 4.1, we have

$$\begin{aligned}&\left\|\left[\sum_e \mathbb{E}_{Y_e^{(k)}} \left(\mathbb{1}_{\mathcal{A}_k} \overline{X_e^{(k)}}\right)^2 \middle| (k), r_k\right]\right\| \\ &\leq \frac{1}{\rho} \left\|\mathbb{1}_{\mathcal{A}_k} \overline{C_{\pi(k)}(S)}\right\| \leq \frac{1+\epsilon}{\rho} \leq \frac{3}{2\rho}.\end{aligned}$$

Thus, we can pick $\sigma_2 = \sqrt{\frac{3}{2\rho}}$. Finally, Lemma 4.1 also gives,

$$\begin{aligned}\mathbb{E}_{\pi(k)} \sum_e \mathbb{E}_{Y_e^{(k)}} \left(\mathbb{1}_{\mathcal{A}_k} \overline{X_e^{(k)}}\right)^2 &\preceq \frac{1}{\rho} \mathbb{E}_{\pi(k)} \mathbb{1}_{\mathcal{A}_k} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})} \\ &\preceq \frac{2(1+\epsilon)}{\rho(n+1-k)} I \preceq \frac{3}{\rho(n+1-k)} I.\end{aligned}$$

Thus, we can pick $\Omega_k = \frac{3}{\rho(n+1-k)} I$, and

$$\sigma_3^2 = \frac{3 \ln n}{\rho} \geq \sum_{k=1}^{n-1} \frac{3}{\rho(n+1-k)}.$$

Similarly, we obtain concentration for $-\widetilde{Z}$ with the same parameters. Thus, by Theorem 4.3,

$$\Pr\left[-\widetilde{Z} \not\preceq \epsilon I\right] + \Pr\left[\widetilde{Z} \not\preceq \epsilon I\right] \leq 2n \exp\left(-\epsilon^2/4\sigma^2\right),$$

for

$$\sigma^2 = \max\left\{\sigma_3^2, \frac{\epsilon}{2}\sigma_1, \frac{4\epsilon}{5}\sigma_2\right\} = \max\left\{\frac{3\ln n}{\rho}, \frac{\epsilon}{2}\sqrt{\frac{3}{2\rho}}, \frac{4\epsilon}{5\rho}\right\}.$$

Picking $\rho = \left\lceil 12(1+\delta)^2\epsilon^{-2}\ln^2 n\right\rceil$, we get $\sigma^2 \leq \frac{\epsilon^2}{4(1+\delta)\ln n}$, and

$$\Pr\left[-\widetilde{Z} \not\preceq \epsilon I\right] + \Pr\left[\widetilde{Z} \not\preceq \epsilon I\right] \leq 2n\exp\left(-\epsilon^2/4\sigma^2\right)$$
$$= 2n\exp\left(-(1+\delta)\ln n\right) = 2n^{-\delta}.$$

Combining this with Equation (7) establishes Equation (4).

Finally, we need to bound the expected running time of the algorithm. We start by observing that the algorithm maintains the two following invariants:

1) Every multi-edge in $\widehat{S}^{(k-1)}$ is $1/\rho$-bounded.
2) The total number of multi-edges is at most ρm.

We establish the first invariant inductively. The invariant holds for $\widehat{S}^{(0)}$, because of the splitting of original edges into ρ copies with weight $1/\rho$. The invariant thus also holds for $\widehat{S}^{(0)} - \left(\widehat{S}^{(0)}\right)_{\pi(1)}$, since the multi-edges of this Laplacian are a subset of the previous ones. By Lemma 4.1, every multi-edge Y_e output by CLIQUESAMPLE is $1/\rho$-bounded, so $\widehat{S}^{(1)} = \widehat{S}^{(0)} - \left(\widehat{S}^{(0)}\right)_{\pi(1)} + \widehat{C}_1$ is $1/\rho$-bounded. If we apply this argument repeatedly for $k = 1, \ldots, n-1$ we get invariant (1).

Invariant (2) is also very simple to establish: It holds for $\widehat{S}^{(0)}$, because splitting of original edges into ρ copies does not produce more than ρm multi-edges in total. When computing $\widehat{S}^{(k)}$, we subtract $\left(\widehat{S}^{(k-1)}\right)_{\pi(k)}$, which removes exactly $\deg_{\widehat{S}^{(k-1)}}(\pi(k))$ multi-edges, while we add the multi-edges produced by the call to CLIQUESAMPLE$(\widehat{S}^{(k-1)}, \pi(k))$, which is at most $\deg_{\widehat{S}^{(k-1)}}(\pi(k))$. So the number of multi-edges is not increasing.

By Lemma 4.1, the running time for the call to CLIQUESAMPLE is $O(\deg_{\widehat{S}^{(k)}}(\pi(k)))$. Given the invariants, we get that the expected time for the k^{th} call to CLIQUESAMPLE is $O(\mathbb{E}_{\pi(k)}\deg_{\widehat{S}^{(k)}}(\pi(k))) = O(\rho m/(n+1-k))$. Thus the expected running time of all calls to CLIQUESAMPLE is $O(m\rho\sum_{i=1}^{n-1}\frac{1}{n-i}) = O(m\delta^2\epsilon^{-2}\ln^3 n)$. The total number of entries in the $\mathcal{L}, \mathcal{D}$ matrices must also be bounded by $O(m\delta^2\epsilon^{-2}\ln^3 n)$ in expectation, and so the permutation step in Line 11 can be applied in expected time $O(m\delta^2\epsilon^{-2}\ln^3 n)$, and this also bounds the expected running time of the whole algorithm. Finally, we can also show that the running time of the algorithm is concentrated. The conditional on previous rounds, the running time in round i of elimination is a positive random variable that is at most $2\rho m$, and in expectation $2\rho m \cdot \frac{1}{n-i}$. By subtracting the average in each round, we can get a scalar martingale. Using a standard application of the scalar Freedman Inequality [37] (see also [20] for a convenient version), one can show that the running time is upper bounded by $O(c\rho m\log n)$ with probability $1 - 1/n^c$ for all $c \geq 1$.

□

V. Clique Sampling Proofs

In this section, we prove Lemmas 4.1 and 4.7 that characterize the behaviour of our algorithm CLIQUESAMPLE, which is used in SPARSECHOLESKY to approximate the clique generated by eliminating a variable.

A important element of the CLIQUESAMPLE algorithm is our very simple approach to leverage score estimation. Using the well-known result that effective resistance in Laplacians is a distance (see Lemma 5.2), we give a bound on the leverage scores of all edges in a clique that arises from elimination. We let

$$w_S(v) = \sum_{\substack{e\in E(S)\\ e\ni v}} w(e).$$

Then by Equation (3)

$$C_v(S) = \frac{1}{2}\sum_{\substack{e\in E(S)\\ e\text{ has}\\ \text{endpoints}\\ v,u}}\sum_{\substack{e'\in E(S)\\ e'\text{ has}\\ \text{endpoints}\\ v,z\neq u}} \frac{w(e)w(e')}{w_S(v)}\boldsymbol{b}_{u,z}\boldsymbol{b}_{u,z}^\top. \qquad (8)$$

Note that the factor $1/2$ accounts for the fact that every pair is double counted.

Lemma 5.1: Suppose multi-edges $e, e' \ni v$ are $1/\rho$-bounded w.r.t. L, and have endpoints v, u and v, z respectively, and $z \neq u$, then $w(e)w(e')\boldsymbol{b}_{u,z}\boldsymbol{b}_{u,z}^\top$ is $\frac{w(e)+w(e')}{\rho}$-bounded.

To prove Lemma 5.1, we need the following result about Laplacians:

Lemma 5.2: Given a connected weighted multi-graph $G = (V, E, w)$ with associated Laplacian matrix L in G, consider three distinct vertices $u, v, z \in V$, and the pair-vectors $b_{u,v}$, $b_{v,z}$ and $b_{u,z}$.

$$\left\|\overline{\boldsymbol{b}_{u,z}\boldsymbol{b}_{u,z}^\top}\right\| \leq \left\|\overline{\boldsymbol{b}_{u,v}\boldsymbol{b}_{u,v}^\top}\right\| + \left\|\overline{\boldsymbol{b}_{v,z}\boldsymbol{b}_{v,z}^\top}\right\|.$$

This is known as phenomenon that Effective Resistance is a distance [38].

Proof of Lemma 5.1: Using the previous lemma:

$$w(e)w(e')\left\|\overline{\boldsymbol{b}_{u,z}\boldsymbol{b}_{u,z}^\top}\right\|$$
$$\leq w(e)w(e')\left(\left\|\overline{\boldsymbol{b}_{u,v}\boldsymbol{b}_{u,v}^\top}\right\| + \left\|\overline{\boldsymbol{b}_{v,z}\boldsymbol{b}_{v,z}^\top}\right\|\right) \leq \frac{1}{\rho}\left(w(e)+w(e')\right).$$

□

To prove Lemma 4.1, we need the following result of Walker [39] (see Bringmann and Panagiotou [40] for a modern statement of the result).

Lemma 5.3: Given a vector $p \in \mathbb{R}^d$ of non-negative values, the procedure UNSORTEDPROPORTIONALSAMPLING requires $O(d)$ preprocessing time and after this allows for IID sampling for a random variable x distributed s.t.

$$\Pr[x = i] = p(i) / \|p\|_1 .$$

The time required for each sample is $O(1)$.

Remark 5.4: We note that there are simpler sampling constructions than that of Lemma 5.3 that need $O(\log n)$ time per sample, and using such a method would only worsen our running time by a factor $O(\log n)$.

Proof of Lemma 4.1: From Lines (7) and (7), Y_i is 0 or the Laplacian of a multi-edge with endpoints u_1, u_2. To upper bound the running time, it is important to note that we do *not* need access to the entire matrix S. We only need the multi-edges incident on v. When calling CLIQUESAMPLE, we only pass a copy of just these multi-edges.

We observe that the uniform samples in Line (3) can be done in $O(1)$ time each, provided we count the number of multi-edges incident on v to find $\deg_S(v)$. We can compute $\deg_S(v)$ in $O(\deg_S(v))$ time. Using Lemma 5.3, if we do $O(\deg_S(v))$ time preprocessing, we can compute each sample in Line (2) in time $O(1)$. Since we do $O(\deg_S(v))$ samples, the total time for sampling is hence $O(\deg_S(v))$.

Now we determine the expected value of the sum of the samples. Note that in the sum below, each pair of multi-edges appears twice, with different weights.

$$\begin{aligned}
&\mathbb{E} \sum_i Y_i \\
&\quad = \deg_S(v) \cdot \\
&\quad \sum_{\substack{e \in E(S) \\ e \text{ has} \\ \text{endpoints} \\ v,u}} \sum_{\substack{e' \in E(S) \\ e' \text{ has} \\ \text{endpoints} \\ v,z \neq u}} \frac{w(e)}{w_S(v)} \frac{1}{\deg_S(v)} \frac{w(e)w(e')}{w(e)+w(e')} \boldsymbol{b}_{u,z} \boldsymbol{b}_{u,z}^\top \\
&\quad = C_v(S).
\end{aligned}$$

By Lemma 5.1,

$$\left\| \overline{Y_i} \right\| \leq \max_{\substack{e,e' \in E(S) \\ e,e' \text{ has} \\ \text{endpoints} \\ v,u \text{ and } v,z \neq u}} \frac{w(e)w(e')}{w(e)+w(e')} \left\| \overline{\boldsymbol{b}_{u,z} \boldsymbol{b}_{u,z}^\top} \right\| \leq 1/\rho.$$

□

Proof of Lemma 4.7: Throughout the proof of this lemma, all the random variables considered are conditional on the choices of the SPARSECHOLESKY algorithm up to and including step $k-1$.

Observe, by Lemma 4.1:

$$\sum_e \mathbb{E}_{Y_e} \overline{Y_e}^2 \preceq \mathbb{E} \sum_e \mathbb{E}_{Y_e} \left\| \overline{Y_e} \right\| \overline{Y_e} \preceq \frac{1}{\rho} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})}.$$

Now

$$\mathbb{E}_{Y_e} \mathbb{1}_{\mathcal{A}_k} \overline{X_e} = \mathbb{1}_{\mathcal{A}_k} \mathbb{E}_{Y_e} \overline{X_e} = 0.$$

Note that $\overline{Y_e}$ and $\mathbb{E}_{Y_e} \overline{Y_e}$ are PSD, and

$$\left\| \mathbb{E}_{Y_e} \overline{Y_e} \right\| \leq \mathbb{E}_{Y_e} \left\| \overline{Y_e} \right\| \leq 1/\rho$$

so

$$\left\| \overline{X_e} \right\| = \left\| \overline{Y_e} - \mathbb{E}_{Y_e} \overline{Y_e} \right\| \leq \max \left\{ \left\| \overline{Y_e} \right\|, \left\| \mathbb{E}_{Y_e} \overline{Y_e} \right\| \right\} \leq 1/\rho.$$

Also $\mathbb{E}_{Y_e} (\mathbb{1}_{\mathcal{A}_k} \overline{X_e})^2 = \mathbb{1}_{\mathcal{A}_k} \mathbb{E}_{Y_e} \overline{X_e}^2$, so

$$\mathbb{E}_{Y_e} \overline{X_e}^2 = (\mathbb{E}_{Y_e} \overline{Y_e}^2) - (\mathbb{E}_{Y_e} \overline{Y_e})^2 \preceq (\mathbb{E}_{Y_e} \overline{Y_e}^2),$$

where, in the last line, we used $0 \preceq (\mathbb{E}_{Y_e} \overline{Y_e})^2$. Thus $\sum_e \mathbb{E}_{Y_e} (\mathbb{1}_{\mathcal{A}_i} \overline{X_e})^2 \preceq \mathbb{1}_{\mathcal{A}_k} \frac{1}{\rho} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})}$. Equation (6) gives:

$$L^{(k-1)} = L + \sum_{i=1}^{k-1} \sum_e X_e^{(i)}. \tag{9}$$

Consequently, $\mathbb{1}_{\mathcal{A}_k} = 1$ gives $L^{(k-1)} \preceq (1+\epsilon)L$.

Now, $C_{\pi(k)}(\widehat{S}^{(k-1)})$ is PSD since it is a Laplacian, so $\left\| \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})} \right\| = \lambda_{\max}(\overline{C_{\pi(k)}(\widehat{S}^{(k-1)})})$. By Equation (2), we get $C_{\pi(k)}(\widehat{S}^{(k-1)}) \preceq \left(\widehat{S}^{(k-1)}\right)_{\pi(k)}$ and by Equation (1) we get $\left(\widehat{S}^{(k-1)}\right)_{\pi(k)} \preceq \widehat{S}^{(k-1)}$, finally by Equation (5) we get $\widehat{S}^{(k-1)} \preceq L^{(k-1)}$ so

$$\begin{aligned}
\left\| \mathbb{1}_{\mathcal{A}_k} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})} \right\| &\leq \mathbb{1}_{\mathcal{A}_k} \lambda_{\max}(\overline{C_{\pi(k)}(\widehat{S}^{(k-1)})}) \\
&\leq \mathbb{1}_{\mathcal{A}_k} \lambda_{\max}(\overline{L^{(k-1)}}) \leq 1 + \epsilon.
\end{aligned}$$

Again, using $C_{\pi(k)}(\widehat{S}^{(k-1)}) \preceq \left(\widehat{S}^{(k-1)}\right)_{\pi(k)}$, we get

$$\begin{aligned}
\mathbb{E}_{\pi(k)} \mathbb{1}_{\mathcal{A}_k} \overline{C_{\pi(k)}(\widehat{S}^{(k-1)})} &\preceq \mathbb{1}_{\mathcal{A}_k} \mathbb{E}_{\pi(k)} \overline{\left(\widehat{S}^{(k-1)}\right)_{\pi(k)}} \\
&= \mathbb{1}_{\mathcal{A}_k} \frac{2}{n+1-k} \overline{S^{(k-1)}} \\
&\preceq \frac{2(1+\epsilon)}{n+1-k} I.
\end{aligned}$$

□

ACKNOWLEDGEMENTS

This work was supported by NSF grant CCF-1111257, ONR Award N00014-16-1-2374, and a Simons Investigator Award to Daniel A. Spielman. We thank Daniel Spielman for suggesting this project and for helpful comments and discussions. We thank Joel Tropp for suggesting an alternative proof of Theorem 4.3, and Michael Cohen for pointing out that the simplest version of our algorithm has concentrated running time.

References

[1] G. Strang, *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[2] E. G. Boman, B. Hendrickson, and S. A. Vavasis, "Solving elliptic finite element systems in near-linear time with support preconditioners," *SIAM J. Numerical Analysis*, vol. 46, no. 6, pp. 3264–3284, 2008.

[3] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," *ICML*, 2003.

[4] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *ICML workshop on statistical relational learning and Its connections to other fields*, vol. 15, 2004, pp. 67–68.

[5] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, no. 16, pp. 321–328, 2004.

[6] S. I. Daitch and D. A. Spielman, "Faster approximate lossy generalized flow via interior point algorithms," in *Proceedings of the 40th annual ACM symposium on Theory of computing*. ACM, 2008, pp. 451–460.

[7] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proceedings of the 43rd annual ACM symposium on Theory of computing*, ser. STOC '11. New York, NY, USA: ACM, 2011, pp. 273–282.

[8] A. Madry, "Navigating central path with electrical flows: From flows to matchings, and back," in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, 2013, pp. 253–262.

[9] Y. T. Lee and A. Sidford, "Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\text{vrank})$ iterations and faster algorithms for maximum flow," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, Oct 2014, pp. 424–433.

[10] J. Kelner and A. Madry, "Faster generation of random spanning trees," in *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*. IEEE, 2009, pp. 13–21.

[11] L. Orecchia, S. Sachdeva, and N. K. Vishnoi, "Approximating the exponential, the lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator." in *Proceedings of The Fourty-Fourth Annual ACM Symposium On The Theory Of Computing (STOC '12)*, 2012.

[12] J. A. Kelner, G. L. Miller, and R. Peng, "Faster approximate multicommodity flow using quadratically coupled flows," in *Proceedings of the 44th symposium on Theory of Computing*, ser. STOC '12. New York, NY, USA: ACM, 2012, pp. 1–18.

[13] A. Levin, I. Koutis, and R. Peng, "Improved spectral sparsification and numerical algorithms for sdd matrices," in *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS)*, 2012.

[14] R. Kyng, A. Rao, and S. Sachdeva, "Fast, provable algorithms for isotonic regression in all l_p-norms," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2719–2727.

[15] K. Gremban, "Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, October 1996, cMU CS Tech Report CMU-CS-96-123.

[16] R. E. T. Richard J. Lipton, Donald J. Rose, "Generalized nested dissection," *SIAM Journal on Numerical Analysis*, vol. 16, no. 2, pp. 346–358, 1979.

[17] N. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Society for Industrial and Applied Mathematics, 2002.

[18] M. B. Cohen, Y. T. Lee, C. Musco, C. Musco, R. Peng, and A. Sidford, "Uniform sampling for matrix approximation," in *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ser. ITCS '15. New York, NY, USA: ACM, 2015, pp. 181–190.

[19] E. H. Lieb, "Convex trace functions and the wigner-yanase-dyson conjecture," *Advances in Mathematics*, vol. 11, no. 3, pp. 267 – 288, 1973.

[20] J. Tropp, "Freedman's inequality for matrix martingales," *Electron. Commun. Probab.*, vol. 16, pp. no. 25, 262–270, 2011.

[21] V. V. Williams, "Multiplying matrices faster than coppersmith-winograd," in *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '12. New York, NY, USA: ACM, 2012, pp. 887–898.

[22] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '04. New York, NY, USA: ACM, 2004, pp. 81–90.

[23] ——, "Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM. J. Matrix Anal. & Appl.*, vol. 35, p. 835885, 2014.

[24] I. Koutis, G. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, 2010, pp. 235 –244.

[25] ——, "A nearly-$m \log n$ time solver for SDD linear systems," in *Foundations of Computer Science (FOCS), 2011 52nd Annual IEEE Symposium on*, 2011, pp. 590–598.

[26] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving sdd systems in nearly-linear time," in *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*. ACM, 2013, pp. 911–920.

[27] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving sdd linear systems in nearly mlog1/2n time," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, ser. STOC '14. New York, NY, USA: ACM, 2014, pp. 343–352.

[28] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, 2014, pp. 333–342.

[29] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the Forty-Eighth Annual ACM on Symposium on Theory of Computing*, ser. STOC '16, 2016.

[30] I. Gustafsson, "A class of first order factorization methods," *BIT Numerical Mathematics*, vol. 18, no. 2, pp. 142–156, 1978.

[31] S. Guattery, "Graph embedding techniques for bounding condition numbers of incomplete factor preconditioning," 1997.

[32] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo, "Support-graph preconditioners," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 930–951, 2006.

[33] K. L. Clarkson, "Solution of linear systems using randomized rounding," 2003.

[34] J. A. Tropp, "User-friendly tail bounds for sums of random matrices," *Foundations of Computational Mathematics*, vol. 12, no. 4, pp. 389–434, 2012.

[35] R. Ahlswede and A. Winter, "Strong converse for identification via quantum channels," *Information Theory, IEEE Transactions on*, vol. 48, no. 3, pp. 569–579, 2002.

[36] M. Rudelson and R. Vershynin, "Sampling from large matrices: An approach through geometric functional analysis," *J. ACM*, vol. 54, no. 4, Jul. 2007.

[37] D. A. Freedman, "On tail probabilities for martingales," *Ann. Probab.*, vol. 3, no. 1, pp. 100–118, 02 1975.

[38] D. J. Klein and M. Randic, "Resistance distance," *Journal of Mathematical Chemistry*, vol. 12, no. 1, pp. 81–95, 1993.

[39] A. J. Walker, "An efficient method for generating discrete random variables with general distributions," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 253–256, Sep. 1977.

[40] K. Bringmann and K. Panagiotou, "Efficient sampling methods for discrete distributions," in *Automata, Languages, and Programming*. Springer, 2012, pp. 133–144.