2016 IEEE 57th Annual Symposium on Foundations of Computer Science

# Faster Algorithms for Computing the Stationary Distribution, Simulating Random Walks, and More (Extended Abstract)

Michael B. Cohen*, Jonathan Kelner†, John Peebles*, Richard Peng‡, Aaron Sidford§, and Adrian Vladu†
*Computer Science and Artificial Intelligence Laboratory
MIT, Cambridge, Massachusetts
Email: {micohen, jpeebles}@mit.edu
†Department of Mathematics
MIT, Cambridge, Massachusetts
Email: {kelner, avladu}@mit.edu
‡College of Computing
Georgia Tech, Atlanta, Georgia
Email: rpeng@cc.gatech.edu
§Department of Management Science and Engineering
Stanford University, Stanford, California
Email: sidford@stanford.edu

***Abstract*—In this paper, we provide faster algorithms for computing various fundamental quantities associated with random walks on a directed graph, including the stationary distribution, personalized PageRank vectors, hitting times, and escape probabilities. In particular, on a directed graph with $n$ vertices and $m$ edges, we show how to compute each quantity in time $\tilde{O}(m^{3/4}n + mn^{2/3})$, where the $\tilde{O}$ notation suppresses polylog factors in $n$, the desired accuracy, and the appropriate condition number (i.e. the mixing time or restart probability).**

**Our result improves upon the previous fastest running times for these problems; previous results either invoke a general purpose linear system solver on a $n \times n$ matrix with $m$ non-zero entries, or depend polynomially on the desired error or natural condition number associated with the problem (i.e. the mixing time or restart probability). For sparse graphs, we obtain a running time of $\tilde{O}(n^{7/4})$, breaking the $O(n^2)$ barrier of the best running time one could hope to achieve using fast matrix multiplication.**

**We achieve our result by providing a similar running time improvement for solving *directed Laplacian systems*, a natural directed or asymmetric analog of the well studied symmetric or undirected Laplacian systems. We show how to solve such systems in time $\tilde{O}(m^{3/4}n + mn^{2/3})$, and efficiently reduce a broad range of problems to solving $\tilde{O}(1)$ directed Laplacian systems on Eulerian graphs. We hope these results and our analysis open the door for further study into *directed spectral graph theory*.**

***Keywords*-PageRank, Markov chain, Laplacian, solver, diagonally dominant, stationary distribution**

## I. Introduction

This is an extended abstract for the full version of our paper [1] which is available at https://arxiv.org/abs/1608.03270. The full version has statements of all theorems and lemmas as well as proofs. It is also more up to date and has more internal cross-references. As such, readers are strongly encouraged to read the full version instead.

The application and development of spectral graph theory has been one of the great algorithmic success stories of the past three decades. By exploiting the relationship between the combinatorial properties of a graph, the linear algebraic properties of its Laplacian, and the probabilistic behavior of the random walks they induce, researchers have obtained landmark results ranging across multiple areas in the theory of algorithms, including Markov chain Monte Carlo techniques for counting [2], [3], [4] and volume estimation [5], [6], [7], [8], [9], [10], approximation algorithms for clustering and graph partitioning problems [11], [12], [13], [14], [15], derandomization [16], [17], error correcting codes [18], [19], and the analysis of random processes [20], among others. In addition to their theoretical impact, spectral techniques have found broad applications in practice, forming the core of Google's PageRank algorithm, playing a ubiquitous role in practical settings like machine learning, computer vision, clustering, and graph visualization. Furthermore, they have enabled the computation of fundamental properties of various Markov chains, such as stationary distributions, escape probabilities, hitting/commute times, and mixing times.

More recently, spectral graph theory has been driving an emerging confluence of algorithmic graph theory, numerical scientific computing, and convex optimization. This recent line of work began with a sequence of papers that used combinatorial techniques to accelerate the solution of linear systems in undirected graph Laplacians, eventually leading to algorithms that solve these systems in nearly-linear time [12], [21], [22], [23], [24], [25], [26], [27], [28]. This was followed by an array of papers in the so-called "Laplacian Paradigm" [29], which either used this nearly-linear-time algorithm as a primitive or built on the structural properties underlying it to obtain faster algorithms for

0272-5428/16 $31.00 © 2016 IEEE
DOI 10.1109/FOCS.2016.69

problems at the core of algorithmic graph theory, including finding maximum flows and minimum cuts [30], [31], [32], [33], [34], solving traveling salesman problems [35], [36], sampling random trees [37], [38], sparsifying graphs [39], [40], [41], computing multicommodity flows [42], [33], and approximately solving a wide range of general clustering and partitioning problems [11], [13], [14], [15].

While these recent algorithmic approaches have been very successful at obtaining algorithms running in close to linear time for undirected graphs, the directed case has conspicuously lagged its undirected counterpart. With a small number of exceptions involving graphs with particularly nice properties and a line of research in using Laplacian system solvers inside interior point methods for linear programming [43], [44], [45], the results in this line of research have centered almost entirely on the spectral theory of undirected graphs. While there have been some interesting results in candidate directed spectral graph theory [46], [14], [47], their algorithmic ramifications have been less clear.

One problem that particularly well illustrates the discrepancy between the directed and undirected settings is the computation of the stationary distribution of a random walk. Computing this is a primary goal in the analysis of Markov chains, constitutes the main step in the PageRank algorithm, remains the missing piece in derandomizing randomized log space computations [48], and is necessary to obtain the appropriate normalization for any of the theoretical or algorithmic results in one of the few instantiations of directed spectral graph theory [46], [14].

In the undirected setting, the stationary distribution is proportional to the degree of a vertex, so it can be computed trivially. However, despite extensive study in the mathematics, computer science, operations research, and numerical scientific computing communities, the best previously known asymptotic guarantees for this problem are essentially what one gets by applying general-purpose linear algebra routines. Given a directed graph with $n$ vertices and $m$ edges these previous algorithms fall into two broad classes:

- **Iterative Methods:** These aim to compute the stationary distribution by either simulating the random walk directly or casting it as a linear system or eigenvector computation and applying either a global or coordinate-wise iterative method to find it. The running times of these methods either depend polynomially on the relevant numerical conditioning property of the instance, which in this case is, up to polynomial factors, the mixing time of the random process; or they only compute a distribution that only approximately satisfies the defining equations of the stationary distribution, with a running time that is polynomial in $1/\epsilon$. There has been extensive work on tuning and specializing these methods to efficiently compute the stationary distribution, particularly in the special case of PageRank. However, all such methods that we are aware of retain a polynomial dependence on either the mixing time, which can be arbitrary large as a function of the number of edges of the graph, or on $1/\epsilon$.[1]
- **Fast Matrix Multiplication**: By using a direct method based on fast matrix multiplication, one can find the stationary distribution in time in time $n^\omega$, where $\omega < 2.3729$ [49] is the matrix multiplication exponent. These methods neglect the graph structure and cannot exploit sparsity. As such, even if one found a matrix multiplication algorithm matching the lower bound of $\omega = 2$, this cannot give a running time lower than $\Omega(n^2)$, even when the graph is sparse.

Another problem which well demonstrates the gap between directed and undirected graph problems is that of solving linear systems involving graph Laplacians. For undirected graphs, as we have discussed there are multiple algorithms to solve associated Laplacian systems in nearly time. However, in the case of directed graphs natural extensions of solving Laplacian systems are closely related to computing the stationary distribution, and thus all known algorithms either depend polynomially on the condition number of the matrix or the desired accuracy or they require time $\Omega(n^2)$. Moreover, many of the techniques, constructions, and properties used to solve undirected Laplacian systems either have no known analogues for directed graphs or can be explicitly shown to not exist. This gap in our ability to solve Laplacian systems is one of the the primary reasons (perhaps *the* primary reason) that the recent wave of graph algorithms based on the "Laplacian Paradigm" have not produced directed results to match the undirected ones.

Given the fact that, despite several decades of work on designing specialized methods for this problem, there are no methods known that asymptotically improve upon general linear algebra routines, along with the structural problems in translating the techniques from the undirected case, it would not be unreasonable to expect that the best one can hope for is heuristic improvements in special cases, and that the worst-case asymptotics for graph Laplacians are no better than the $\min O(n^\omega, nm) \geq \Omega(n^2)$ that is known for general matrices.

In this paper, we show that this is not the case by providing an algorithm that solves directed graph Laplacian systems—a natural generalization of undirected graph Laplacian systems—in time $\tilde{O}(nm^{3/4} + n^{2/3}m)$ where here and throughout the paper the $\tilde{O}(\cdot)$ notation hides polylogarithmic factors in $n$, the desired accuracy, and the natural condition numbers associated with the problem. Consequently, we obtain the first asymptotic improvement for these systems

[1]The algorithm that is perhaps an exception to this rule, is the one which invokes conjugate gradient in a black box manner to solve the requisite linear system to compute the stationary distribution. At best this analysis would suggest an $O(mn)$ running time. However, it is not known how to realize even this running time in the standard word-RAM model of computation.

over solving general linear systems.[2] In particular, when the graph is sparse, i.e. $m = O(n)$, our algorithm runs in time $\tilde{O}(n^{7/4})$, breaking the barrier of $O(n^2)$ that would be achieved by algorithms based on fast matrix multiplication if $\omega = 2$. We then leverage this result to obtain improved running times for a host of problems in algorithmic graph theory, scientific computing, and numerical linear algebra, including:

- **Computing the Stationary Distribution**: We compute a vector within $\ell_2$ distance $\epsilon$ of the stationary distribution of a random walk on a strongly connected directed graph in time $\tilde{O}(nm^{3/4} + n^{2/3}m)$, where the natural condition number of this problem is the mixing time.
- **Solving Large Classes of Linear Systems**: We provide algorithms that solve a large class of well-studied linear systems. Compared with prior algorithms capable of solving this class, ours are the first that are asymptotically faster than solving general linear systems, and the first that break the $O(n^2)$ barrier for sufficiently sparse instances. Our methods solve directed Laplacian systems and systems where the matrix is both row- and column-diagonally dominant. The running time is $\tilde{O}\left(nm^{3/4} + n^{2/3}m\right)$.
- **Computing Personalized PageRank**: We compute a vector within $\ell_2$ distance $\epsilon$ of the personalized PageRank vector, for a directed graph with with restart probability $\beta$, in time $\tilde{O}\left(nm^{3/4} + n^{2/3}m\right)$. Here the natural condition number is $1/\beta$. In the case of small $\beta$ and $\epsilon$, this improves upon local methods that take $O(m\beta^{-1}\epsilon^{-1})$ time [50], [51], [52], [14], [53], [54].
- **Simulating Random Walks**: We show how to compute a wide range of properties of random walks on directed graphs including escape probabilities, commute times, and hitting times. We also show how to efficiently estimate the mixing time of a lazy random walk on a directed graph up to polynomial factors in $n$ and the mixing time. The runtime for all these algorithms is $\tilde{O}\left(nm^{3/4} + n^{2/3}m\right)$.
- **Estimating All Commute Times**: We show how to build a $\tilde{O}(n\epsilon^{-2}\log n)$ size data structure in time $\tilde{O}(nm^{3/4} + n^{2/3}m)$ that, when queried with any two vertices $a$ and $b$, outputs a $1 \pm \epsilon$ multiplicative approximation to the expected commute time between $a$ and $b$, i.e. the expected amount of time for a random walk starting at $a$ to reach $b$ and return to $a$. Our data structure is similar to the data structure known for computing all-pairs effective resistances in undirected graphs [39].

It is important to note that the $\tilde{O}$-notation hides factors that are polylogarithmic in both the condition number (equivalently, mixing time) and the ratio of maximum to minimum stationary probability. As such, the natural parameter regime for our algorithms is when these quantities are subexponential or polynomial. For all the above problems, the best prior algorithms had worst case runtimes no better than $O(\min\{n^\omega, nm\}) \geq \Omega(n^2)$ in this regime. We hope that our results open the door for further research into directed spectral graph theory, and serve as foundation for the development of faster algorithms for directed graphs.

### A. Approach

Our approach for solving these problems centers around solving linear systems in a class of matrices we refer to as *directed (graph) Laplacians*, a natural generalization of undirected graph Laplacians. A directed Laplacian, $\mathcal{L} \in \mathcal{R}^{n\times n}$, is simply a matrix with non-positive off-diagonal entries such that each diagonal entry is equal to the sum of the absolute value of the other off-diagonal entries in that column, i.e. $\mathcal{L}_{ij} \leq 0$ for $i \neq j$ and $\mathcal{L}_{ii} = -\sum_{j\neq i}\mathcal{L}_{ji}$ (equivalently $\vec{1}^\top\mathcal{L} = \vec{0}$). As with undirected Laplacians, every directed Laplacian there is naturally associated with a directed graph $G = (V, E, w)$, where the vertices $V$ correspond to the columns of $\mathcal{L}$ and there is an edge from vertex $i$ to vertex $j$ of weight $\alpha$ if and only if $\mathcal{L}_{ji} = -\alpha$.

Another close feature of directed and undirected Laplacians is the close connection between random walks on the associated graph $G$ and solutions to linear systems in $\mathcal{L}$. We ultimately show that solving a small number of directed Laplacian systems suffices to obtain all of our desired applications (See Section V and Section VII). Unfortunately, solving linear systems in $\mathcal{L}$ directly is quite challenging. Particularly troubling is the fact that we while we know $\mathcal{L}$ has a non-trivial kernel (since $\vec{1}^\top\mathcal{L} = \vec{0}^\top$), we do not have a simple method to compute it efficiently. Moreover, $\mathcal{L}$ is not symmetric, complicating the analysis of standard iterative algorithms.Furthermore, the standard approach of multiplying on the left by the transpose, so that we are solving linear systems in $\mathcal{L}^\top\mathcal{L}$, would destroy the combinatorial structure of the problem and cause an intolerably large condition number. A natural idea is to try to work with a symmetrization of this matrix, $\frac{1}{2}(\mathcal{L} + \mathcal{L}^\top)$, but it turns out that this may not even be positive semidefinite (PSD).[3] Consequently, it is not clear a priori how to define an efficient iterative method for computing the stationary $\mathcal{L}$ or solve systems in it without depending polynomially on the condition number of $\mathcal{L}$.

Fortunately, we do know how to characterize the kernel of $\mathcal{L}$, even if computing it is difficult *a priori*. If we let

[2] In follow up work, the authors of this paper in collaboration with Anup Rao have improved the running time to almost linear in the number of edges in the graph, meaning the running time is linear if we ignore contributions to the running time that are smaller than any polynomial. This paper will be made available online as soon as possible.

[3] Consider the directed edge Laplacian $\mathcal{L} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$. Then, $\mathcal{L} + \mathcal{L}^+ = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix}$ has an eigenvector $(\sqrt{2}-1, 1)$ with a corresponding eigenvalue of $(1 - \sqrt{2})$.

$\mathbf{D} \in \mathcal{R}^{n \times n}$ denote the diagonal matrix consistent with $\mathcal{L}$, i.e., $\mathbf{D}_{ii} = \mathcal{L}_{ii}$, then we see that $\mathcal{L}\mathbf{D}^{-1} = \mathbf{I} - \mathbf{W}$ where $\mathbf{I}$ is the identity matrix and $\mathbf{W}$ is the random walk matrix associated with $G$. In other words, for any distribution $p$, we have that $\mathbf{W}p$ is the resulting distribution of one step of the random walk where, at a vertex $i \in [n]$, we pick a random outgoing edge with probability proportional to its weight and follow that edge. The Perron-Frobenius Theorem implies that as long as the graph is strongly connected there is some stationary distribution $s \in \mathcal{R}_{>0}$ such that $\mathbf{W}s = s$. Consequently, the kernel of $\mathcal{L}$ is simply the stationary distribution of the natural random walk on $G$ multiplied by $\mathbf{D}$.

Consequently, we can show that for every directed Laplacian $\mathcal{L}$ that corresponds to a strongly connected graph, there is always a vector $x \in \mathcal{R}^n_{>0}$ such that $\mathcal{L}x = 0$ (See Lemma 1). In other words, letting $\mathbf{X}$ denote the diagonal matrix associated with $x$ the directed Laplacian $\mathcal{L}' = \mathcal{L}\mathbf{X}$ satisfies $\mathcal{L}'\mathbf{X}1 = 0$. This says that the total weight of incoming edges to a vertex is the same as the total weight of outgoing edges from that vertex, i.e., that $\mathcal{L}'$ corresponds to the Laplacian of an Eulerian graph. We call such a vector an *Eulerian scaling* of $\mathcal{L}$.

Now, solving systems in an Eulerian Laplacian $\mathcal{L}$ (i.e., a Laplacian corresponding to an Eulerian graph) seems easier than solving an arbitrary directed Laplacian. In particular, we know the kernel of a $\mathcal{L}$, since it is just the all ones vector. In addition, we have that $\frac{1}{2}(\mathcal{L} + \mathcal{L}^\top)$ is symmetric and PSD—in fact it is just the Laplacian of an undirected graph! Unfortunately, this does not immediately yield an algorithm, as it is not known how to use the ability to solve systems in such a symmetrization to solve systems in the original matrix.

Ultimately, this line of reasoning leaves us with two fundamental questions:

1) Can we solve Eulerian Laplacian systems in time $o(n^\omega, nm)$?
2) Can we use an Eulerian Laplacian system solver for more than solving Eulerian Laplacian systems?

The major contribution of this paper is answering both of these questions in the affirmative. We show the following:

- We show that we can solve Eulerian Laplacian systems in time $\tilde{O}\left(nm^{3/4} + n^{2/3}m\right)$.
- We show that using Eulerian Laplacian systems we can solve broader classes of matrices we refer to as RCDD Z-matrices, and $\alpha$ RCDD Z-matrices.
- We show that using solvers for $\alpha$ RCDD Z-matrices, we can estimate an Eulerian scaling of a directed Laplacian.
- Putting these components together we achieve our desired applications. Some of these are applications are straightforward, whereas others require some significant work.

A serious question that arises throughout these results is the degree of precision do we need to carry out our arithmetic operations. This arrises both in using undirected Laplacian system solvers to solving Eulerian Laplacian systems, and then again in using Eulerian Laplacian system solvers to derive the rest of our results. These numerical issues are not merely technicalities—they crucially affect the algorithms we can use to solve our problem. In fact, we will see in Section IV that, if we disregarded the numerics and relied on frequently-quoted assertions relating the behavior of conjugate gradient to the existence of polynomials with certain properties, we would actually obtain a better running time, but that these assertions do not carry over to reasonable finite-precision setting.

Given these subtleties, we discuss numerical issues throughout the full version of the paper, showing that we can achieve all our results in the standard unit cost RAM model (or any other reasonable model of computation).

We now briefly comment on the key technical ingredients of each of these results.

*1) Solving Eulerian Laplacian Systems:* To solve a Eulerian Laplacian system $\mathcal{L}x = b$, we first precondition, multiplying both sides by $\mathcal{L}^\top\mathbf{U}^+$, where $\mathbf{U} \stackrel{\text{def}}{=} \frac{1}{2}(\mathcal{L}^\top + \mathcal{L})$ is a Laplacian of an undirected graph corresponding to $\mathcal{L}$, and $\mathbf{U}^+$ is its Moore-Penrose pseudoinverse. This shows that it suffices to instead solve, $\mathcal{L}^\top\mathbf{U}^+\mathcal{L}x = \mathcal{L}^\top\mathbf{U}^+x$. Now using a nearly-linear-time Laplacian system solver, we can apply $\mathbf{U}^+$ to a vector efficiently. As such, we simply need to show that we can efficiently solve systems in the symmetric matrix $\mathcal{L}^\top\mathbf{U}^+\mathcal{L}$.

Next, we show that the matrix $\mathcal{L}^\top\mathbf{U}^+\mathcal{L}$ is, in an appropriate sense, approximable by $\mathbf{U}$. Formally we show that $\mathbf{U}$ is smaller in the sense that $\mathbf{U} \preceq \mathcal{L}^\top\mathbf{U}^+\mathcal{L}$, and that it is not too much larger in the sense that $\mathrm{tr}(\mathbf{U}^{+/2}\mathcal{L}^\top\mathbf{U}^+\mathcal{L}\mathbf{U}^{+/2}) = O(n^2)$. While the first proof is holds for a broad class of asymmetric matrices, to prove the second fact we exploit structure of Eulerian Laplacians, particularly the fact that an Eulerian graph has a decomposition into simple cycles.

Unfortunately, this property doesn't immediately yield an algorithm for solving Laplacian systems. The natural approach would be to use preconditioned Krylov methods, such as the Chebyshev method or conjugate gradient. These essentially apply a polynomial of $\mathbf{U}^+\mathcal{L}^\top\mathbf{U}^+\mathcal{L}$ to the right hand side. Unfortunately, Chebyshev iterations only yield a $\Omega(mn)$ time algorithm with this approach. For conjugate gradient, it can be shown that the trace bound leads to $o(mn)$ time algorithm in exact arithmetic, but, unfortunately, this analysis does not appear to be numerically stable, and we do not know how to show it yields this running time in our computational model.

Instead we implement an approach based on preconditioning and subsampling. We precondition with $\mathcal{L}^\top\mathbf{U}^+\mathcal{L} + \alpha\mathbf{U}$ for a value of $\alpha$ we tune. This reduces the problem to only solving $\tilde{O}(\sqrt{\alpha})$ linear systems in $\mathcal{L}^\top\mathbf{U}^+\mathcal{L} + \alpha\mathbf{U}$. To solve

these systems we note that we can write this equivalently as $\mathcal{L}^\top \mathbf{U}^+ \mathbf{U} \mathbf{U}^+ \mathcal{L} + \alpha \mathbf{U}$ and using the factorization of $\mathbf{U}$ into its edges we can subsample the inner $\mathbf{U}$ while preserving the matrix. Ultimately, this means we only need to solve systems in $\alpha \mathbf{U}$ plus a low rank matrix which we can do efficiently using the fact that there is an explicitly formula for low rank updates (i.e. Sherman-Morrison-Woodbury Matrix Identity). Trading between the number of such systems to solve, the preprocessing to solve these systems, and the time to solve them gives us our desired running time for solving such linear systems. We show in the appendix that we can, with some care, stably implement a preconditioned Chebyshev method and low rank update formulas. This allows us to circumvent the issues in using conjugate gradient and achieve our running time in the desired computational model.

*2) Solving RCDD Z-matrices:* A row column diagonal dominant (RCDD) matrix is simply a matrix $\mathbf{M}$ where $\mathbf{M}_{ii} \geq \sum_{j \neq i} |\mathbf{M}_{ij}|$ and $\mathbf{M}_{ii} \geq \sum_{j \neq i} |\mathbf{M}_{ji}|$ and a Z-matrix is a matrix $\mathbf{M}$ where the off-diagonal entries are negative. We show how to solve such matrices by directly reducing them to solving Eulerian Laplacian systems. Given a RCDD Z-matrix $\mathbf{M}$, we add an additional row and column, filling in the entries in the natural way so that the resulting matrix is an Eulerian Laplacian. We show that, from the solution to such a linear system, we can immediately glean the solution to systems in $\mathbf{M}$. This reduction is analogous to the reduction from solving symmetric diagonally dominant (SDD) systems to solving undirected Laplacian systems. In the appendix we show that this method is stable.

*3) Computing the Stationary:* Given a RCDD Z-matrix solver we use it to compute the scaling that makes a directed Laplacian $\mathcal{L}$ Eulerian, i.e., we compute the stationary distribution. To do this, we pick an initial non-negative diagonal scaling $\mathbf{X}$ and a initial non-negative diagonal matrix $\mathbf{E}$ such that $(\mathbf{E} + \mathcal{L})\mathbf{X}$ is $\alpha$-RCDD, that is each diagonal entry is a $1 + \alpha$ larger in absolute value than the sum of the off-diagonal entries in both the corresponding row and column.

We then iteratively decrease $\mathbf{E}$ and update $\mathbf{X}$ while maintaining that the matrix is $\alpha$-RCDD. The key to this reduction is the fact that there is a natural way to perform a rank 1 update of $(\mathbf{E}+\mathcal{L})\mathbf{X}$ to obtain an Eulerian Laplacian, and that the stationary distribution of this Laplacian can be obtained by solving a single system in $(\mathbf{E} + \mathcal{L})\mathbf{X}$. Ultimately, this method yields a sequence of stationary distributions that, when multiplied together entrywise, yield a good approximate stationary distribution for $\mathcal{L}$. For a more detailed over this approach and this intuition underlying it, see Section III-A.

*4) Applications:* Our algorithms for computing personalized page rank, solving arbitrary RCDD systems, and solving aribtrary directed Laplacian systems are all proven in a similar fashion. We obtain an approximate stationary distribution, rescale the system to make it strictly RCDD, then solve it—all using algorithms from the previous sections in a black box fashion. Therefore, the running times for these applications—and in fact all our applications—depend solely[4] on the black-box costs of computing the stationary distribution and solving RCDD matrices.

However, our algorithms must determine how much accuracy to request when they invoke these two black-box routines. For computing personalized PageRank, one can determine the accuracy to request based solely on the restart probability. However, for our other applications, the accuracy our algorithms request has a dependence on the condition number $\kappa(\mathcal{L})$ of $\mathcal{L}$ and the ratio $\kappa(\mathbf{S}^*)$ of max over min stationary probability. In order to get an unconditional running time—and out of intrinsic interest—we show how to efficiently compute reasonable upper bounds on these quantities. We use an approach motivated by the close relationship of $\kappa(\mathcal{L})$ and mixing time. Specifically, we formulate a notion of personalized PageRank mixing time, then get a polynomially good estimate of this quantity using our ability to solve personalized PageRank. Finally, we show that $\kappa(\mathcal{L})$ and personalized pagerank mixing time are equivalent up to factors that are good enough[5] for our purposes. With a reasonable bound on $\kappa(\mathcal{L})$, we are then able to choose a restart probability that is small enough in order to guarantee that personalized solving PageRank gives a good approximation of the stationary distribution.

Our algorithms for computing hitting times, escape probabilities, and all pairs commute times all start by taking a natural definition for the quantity of interest and massaging it into an explicit formula that has an $\mathcal{L}^+$ in it. Then, they use various methods to approximately evaluate the formula. In the case of hitting times, we simply plug everything into the formula and invoke our approximate solver for $\mathcal{L}^+$ with appropriately small error. Escape probabilities are handled similarly, except that there are also two unknown parameters which we show we can estimate to reasonable accuracy and plug in.

Perhaps the most sophisticated application is computing all pairs commute times. We show that the commute time from $u$ to $v$ is given by the simple formula $(\vec{1}_u - \vec{1}_v)^\intercal (\mathcal{L}_b^\top \mathbf{U}_b^+ \mathcal{L}_b)^+ (\vec{1}_u - \vec{1}_v)$ where $\mathcal{L}_b$ is the matrix obtained by performing the diagonal rescaling of $\mathcal{L}$ that turns its diagonal into the stationary distribution, which also makes the graph Eulerian. An interesting feature of this formula is that it involves applying the pseudo-inverse of a matrix of the very same form as the preconditioned system $\mathcal{L}^\top \mathbf{U}^+ \mathcal{L}$ that our Eulerian Laplacian solver uses. Another interesting feature is that when $\mathcal{L}$ is symmetric, this formula simplifies to $(\vec{1}_u - \vec{1}_v)^\intercal \mathbf{U}_b^+ (\vec{1}_u - \vec{1}_v) = 2m \cdot (\vec{1}_u - \vec{1}_v)^\intercal \mathbf{U}^+ (\vec{1}_u - \vec{1}_v)$. Thus, it is a generalization of the well-known characterization of commute times in terms of effective resistance from undirected graphs. In undirected graphs, all pairs commute

[4]up to polylogarithmic factors

[5]They are equivalent up to factors polynomial in $n$ and themselves. Since these quantities appear only in logs in our runtimes, this is good enough.

times can be computed efficiently via Johnson-Lindenstrauss sketching [39]. We show that a similar approach extends to directed Laplacians as well. While the general approach is similar, the error analysis is complicated by the fact that we only have access to an approximate stationary distribution. If this were used naively, one would have to deal with an approximate version of $\mathcal{L}_b$ that, importantly, is only approximately Eulerian. We bypass this issue by showing how to construct an Eulerian graph whose stationary is exactly known and whose commute times approximate the commute times of the original graph. This may be of independent interest.

The fact that a matrix of the form $\mathcal{L}^\top \mathbf{U}^+ \mathcal{L}$ comes up both in solving Eulerian Laplacians and in sketching commute times indicates that it is a considerably more natural object than it might appear at first.

### B. Paper Organization

The rest of the paper is organized as follows:

- **Section II** - we cover preliminary information
- **Section III** - we show how to compute the stationary distribution
- **Section IV** - we provide our fast Eulerian Laplacian system solver
- **Section V** - we reduce strict RCDD linear systems to solving Eulerian systems
- **Section** VI - we provide condition number quantities for applications and prove equivalences
- **Section VII** - we provide our applications

## II. Preliminaries

### A. Notation

**Matrices:** We use bold to denote matrices and let $\mathbf{I}_n, \mathbf{0}_n \in \mathcal{R}^{n\times n}$ denote the identity matrix and zero matrix respectively. For symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathcal{R}^{n\times n}$ we use $\mathbf{A} \preceq \mathbf{B}$ to denote the condition that $x^\top \mathbf{A} x \leq x^\top \mathbf{B} x$ and we define $\succeq$, $\prec$, and $\succ$ analogously. We call a symmetric matrix $\mathbf{A} \in \mathcal{R}^{n\times n}$ positive semidefinite if $\mathbf{A} \succeq \mathbf{0}_n$ and we let $\|x\|_{\mathbf{A}} \stackrel{\text{def}}{=} \sqrt{x^\top \mathbf{A} x}$. For any norm $\|\cdot\|$ define on vectors in $\mathcal{R}^n$ we define the *operator norm* it induces on $\mathcal{R}^{n\times n}$ by $\|\mathbf{A}\| = \max_{x\neq 0} \frac{\|\mathbf{A}x\|}{\|x\|}$ for all $\mathbf{A} \in \mathcal{R}^{n\times n}$.

**Diagonals**: For $x \in \mathcal{R}^n$, $\mathbf{diag}(x) \in \mathcal{R}^{n\times n}$ denotes the matrix whose diagonal is $x$ and off-diagonal entries are 0. For $\mathbf{A} \in \mathcal{R}^{n\times n}$, $\mathrm{diag}(\mathbf{A})$ denotes the vector corresponding to the diagonal of $\mathbf{A}$ and we let $\mathbf{diag}(\mathbf{A}) \stackrel{\text{def}}{=} \mathbf{diag}(\mathrm{diag}(\mathbf{A}))$, i.e. $\mathbf{A}$ with the off-diagonal set to 0.

**Vectors**: We let $\vec{0}_n, \vec{1}_n \in \mathcal{R}^n$ denote the all zeros and ones vectors respectively. We use $\vec{1}_i \in \mathcal{R}^n$ to denote the indicator vector for coordinate $i \in [n]$, i.e. $[\vec{1}_i]_j = 0$ for $j \neq i$ and $[\vec{1}_i]_i = 1$. Occasionally we apply scalar operations to vectors with the interpretation that they should be applied coordinate-wise, e.g. for $x, y \in \mathcal{R}^n$ we let $\max\{x, y\}$ denote the vector $z \in \mathcal{R}^n$ with $z_i = \max\{x_i, y_i\}$ and we use $x \geq y$ to denote the condition that $x_i \geq y_i$ for all $i \in [n]$.

**Condition Numbers**: Given a invertible matrix $\mathbf{A} \in \mathcal{R}^{n\times n}$ we let $\kappa(\mathbf{A}) \stackrel{\text{def}}{=} \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2$ denote the condition number of $\mathbf{A}$. Note that if $\mathbf{X} \in \mathcal{R}^{n\times n}$ is a diagonal matrix then $\kappa(\mathbf{X}) = \frac{\max_{i\in[n]} |\mathbf{X}_{ii}|}{\min_{i\in[n]} |\mathbf{X}_{ii}|}$.

**Sets**: We let $[n] \stackrel{\text{def}}{=} \{1, ..., n\}$ and $\Delta^n \stackrel{\text{def}}{=} \{x \in \mathcal{R}^n_{\geq 0} \mid \vec{1}_n^\top x = 1\}$, i.e. the $n$-dimensional simplex.

### B. Matrix Classes

**Diagonal Dominance:** A possibly asymmetric matrix $\mathbf{A} \in \mathcal{R}^{n\times n}$ *is $\alpha$-row row-column diagonally dominant (RCDD)* if $\mathbf{A}_{ii} \geq (1+\alpha)\sum_{j\neq i} |\mathbf{A}_{ij}|$ for all $i \in [n]$, *$\alpha$-column diagonally dominant (CDD)* if $\mathbf{A}_{ii} \geq (1+\alpha)\sum_{j\neq i} |\mathbf{A}_{ji}|$, and *$\alpha$-RCDD* if it is both $\alpha$-RDD and $\alpha$-CDD. For brevity, we call $\mathbf{A}$ RCDD if it is 0-RCDD and strictly RCDD if it is $\alpha$-RCDD for $\alpha > 0$.

**Z-matrix**: A matrix $\mathbf{M} \in \mathcal{R}^{n\times n}$ is called a Z-matrix if $\mathbf{M}_{ij} \leq 0$ for all $i \neq j$, i.e. every off-diagonal entry is non-positive.

**Directed Laplacian**: A matrix $\mathcal{L} \in \mathcal{R}^{n\times n}$ is called a *directed Laplacian* if it a Z-matrix with $\vec{1}_n^\top \mathcal{L} = \vec{0}_n$, that is $\mathcal{L}_{ij} \leq 0$ for all $i \neq j$ and $\mathcal{L}_{ii} = -\sum_{j\neq i} \mathcal{L}_{ji}$ for all $i$. To every directed Laplacian $\mathcal{L}$ we associate a graph $G_{\mathcal{L}} = (V, E, w)$ with vertices $V = [n]$ and an edge $(i, j)$ of weight $w_{ij} = -\mathcal{L}_{ji}$ for all $i \neq j \in [n]$ with $\mathcal{L}_{ji} \neq 0$. Occasionally we write $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$ to denote that we decompose $\mathcal{L}$ into the diagonal matrix $\mathbf{D}$ where $\mathbf{D}_{ii} = \mathcal{L}_{ii}$ is the out degree of vertex $i$ in $G_{\mathcal{L}}$ and $\mathbf{A}$ is weighted adjacency matrix of $G_{\mathcal{L}}$ with $\mathbf{A}_{ij} = w_{ij}$ if $(i, j) \in E$ and $\mathbf{A}_{ij} = 0$ otherwise. We call $\mathbf{W} = \mathbf{A}^\top \mathbf{D}^{-1}$ the random walk matrix associated with $G_{\mathcal{L}}$. We call $\mathcal{L}$ Eulerian if additionally $\mathcal{L}\vec{1}_n = \vec{0}_n$ as in this case the *associated graph* $G_{\mathcal{L}}$ is Eulerian.

**(Symmetric) Laplacian**: A matrix $\mathbf{U} \in \mathcal{R}^{n\times n}$ is called a *Symmetric Laplacian* or just a *Laplacian* if it is symmetric and a Laplacian. This coincides with the standard definition of Laplacian and in this case note that the associated graph $G_{\mathbf{U}} = (V, E, w)$ is symmetric. For a Laplacian we also associate a matrix $\mathbf{B} \in \mathcal{R}^{E\times V}$ known as the weighted incidence matrix. Each row $b^{(i)}$ of $\mathbf{B}$ corresponds to an edge $\{j, k\} \in E$ and for a canonical orientation ordering of $\{j, k\}$ we have $b^{(i)}_j = \sqrt{w_{\{j,k\}}}$, $b^{(i)}_k = -\sqrt{w_{\{j,k\}}}$, and $b^{(i)}_l = 0$ if $l \notin \{j, k\}$. Note that $\mathbf{U} = \mathbf{B}^\top \mathbf{B}$ and thus $\mathcal{L}$ is always PSD.

**Random Walk Matrix**: A matrix $\mathbf{W} \in \mathcal{R}^{n\times n}$ is called a *random walk matrix* if $\mathbf{W}_{ij} \geq 0$ for all $i, j \in [n]$

and $\vec{1}_n^\top \mathbf{W} = \vec{1}_n$. To every random walk matrix $\mathbf{W}$ we associated a directed graph $G_{\mathbf{W}} = (V, E, w)$ with vertices $V = [n]$ and an edge from $i$ to $j$ of weight $w_{ij} = \mathbf{W}_{ij}$ for all $i, j \in [n]$ with $\mathbf{W}_{ij} \neq 0$. Note if we say that $\mathcal{L} = \mathbf{I} - \mathbf{W}$ is a directed Laplacian, then $\mathbf{W}$ is a random walk matrix and the directed graphs associated with $\mathcal{L}$ and $\mathbf{W}$ are identical.

**Lazy Random Walk Matrix:** Given a random walk matrix $\mathbf{W} \in \mathcal{R}^{n \times n}$ the $\alpha$-lazy random walk matrix associated with $\mathbf{W}$ for $\alpha \in [0, 1]$ is given by $\alpha \mathbf{I} + (1 - \alpha)\mathbf{W}$. When $\alpha = \frac{1}{2}$ we call this a lazy random walk matrix for short and typically denote it $\widetilde{\mathbf{W}}$.

**Personalized PageRank Matrix**: Given a random walk matrix $\mathbf{W} \in \mathcal{R}^{n \times n}$ the personalized PageRank matrix with restart probability $\beta \in [0, 1]$ is given by $\mathbf{M}_{pp(\beta)} = \beta(\mathbf{I} - (1 - \beta)\mathbf{W})^{-1}$. Given any probability vector $p \in \Delta^n$ the personalized PageRank vector with restart probability $\beta$ and vector $p$ is the vector $x$ which satisfies $\beta p + (1 - \beta)\mathbf{W}x = x$. Rearranging terms we see that $x = \mathbf{M}_{\beta} p$ hence justifying our naming of $\mathbf{M}_{pp(\beta)}$ as the personalized PageRank matrix.

### C. Directed Laplacians of Strongly Connected Graphs

The most important properties of directed Laplacians tha twe use are encapsulated below.

**Lemma 1.** *For directed Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top \in \mathcal{R}^{n \times n}$ whose associated graph is strongly connected there exists a positive vector $s \in \mathcal{R}^n_{>0}$ (unique up to scaling) such that the following equivalent conditions hold.*

- $\mathbf{W}s = s$ *for the random walk matrix* $\mathbf{W} = \mathbf{A}^\top \mathbf{D}^{-1}$ *associated with* $\mathcal{L}$.
- $\mathcal{L}\mathbf{D}^{-1}s = 0$
- $\mathcal{L}\mathbf{D}^{-1}\mathbf{S}$ *for* $\mathbf{S} = \mathbf{diag}(s)$ *is an Eulerian Laplacian.*

*If we scale $s$ so that $\|s\|_1 = 1$ then we call $s$ the stationary distribution associated with the random walk on the associated graph $G_{\mathcal{L}}$. We call any vector $x \in \mathcal{R}^n_{>0}$ such that $\mathcal{L}\mathbf{X}$ is an Eulerian Laplacian an eulerian scaling for $\mathcal{L}$. Furthermore,* $\ker(\mathcal{L}) = \mathrm{span}(\mathbf{D}^{-1}s)$ *and* $\ker(\mathcal{L}^\top) = \mathrm{span}(\vec{1}_n)$.

## III. Computing the Stationary Distribution

Here we show to compute the stationary distribution given an $\alpha$-RCDD Z-matrix linear system solver. Throughout this section, we let $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top \in \mathcal{R}^{n \times n}$ denote a directed Laplacian and our primary goal in this section is to compute an approximate stationary vector $s \in \mathcal{R}^V_{>0}$ such that $\mathcal{L}\mathbf{D}^{-1}\mathbf{S}$ is approximately Eulerian. The main result of this section is the following:

**Theorem 2** (Stationary Computation Using a RCDD Solver)**.** *Given $\alpha \in (0, \frac{1}{2})$ and $\mathcal{L} \in \mathcal{R}^{n \times n}$, a directed Laplacian with $m$ nonzero-entries, we can compute in time $O((m+\mathcal{T}) \cdot \log \alpha^{-1})$ an approximate stationary distribution $s \in \Delta^n$ such that $(3\alpha n \cdot \mathbf{D} + \mathcal{L})\mathbf{D}^{-1}\mathbf{S}$ is $\alpha$-RCDD where $\mathbf{D} = \mathbf{diag}(\mathcal{L})$, $\mathbf{S} = \mathbf{diag}(s)$, and $\mathcal{T}$ is the cost of computing an $\epsilon$-approximate solution to a $n \times n$ $\alpha$-RCDD Z-matrix linear system with $m$-nonzero entries, i.e. computing $x$ such that $\|x - \mathbf{M}^{-1}b\|_{\mathbf{diag}(\mathbf{M})} \leq \frac{\epsilon}{\alpha}\|b\|_{\mathbf{diag}(\mathbf{M})^{-1}}$ for $\alpha$-RCDD Z-matrix $\mathbf{M} \in \mathcal{R}^{n \times n}$ with $O(m)$ non-zero entries and $\epsilon = O(\mathsf{poly}(n/\alpha))$. Furthermore, $\kappa(\mathbf{S}) \leq \frac{20}{\alpha^2} n$, where $\kappa(\mathbf{S})$ is the ratio between the largest and smallest elements of $s$.*

An analysis of this algorithm is in the full version of this paper.

### A. The Approach

Our approach to computing the stationary distribution is broadly related to the work of Daitch and Spielman [55] for computing the diagonal scaling makes a symmetric $M$-matrix diagonally dominant. However, the final rescaling that we are trying to calculate is given by the uniqueness of the stationary distribution, which in turn follows from the Perron-Frobenius theorem.

As in [55] we use an iterative that brings a matrix increasingly close to being RCDD. However, our convergence process is through potential functions instead of through combinatorial entry count; instead of making parts of $\mathcal{L}$ RCDD incrementally we instead start with a relaxation of $\mathcal{L}$ that is RCDD and iteratively bring this matrix closer to $\mathcal{L}$ while maintaining that it is RCDD. We remark that this scheme can also be adapted to find rescalings of symmetric M-matrices.

Our algorithm hinges on two key insights. The first is that if we have positive vectors $e, x \in \mathcal{R}^n_{>0}$ so that $\mathbf{M} \stackrel{\text{def}}{=} (\mathbf{E} + \mathcal{L})\mathbf{X}$ is $\alpha$-RCDD, then for any vector $g \in \mathcal{R}^n_{>0}$ with $\|g\|_1 = 1$ we can compute the stationary distribution of the directed Laplacian $\mathcal{L}' = \mathbf{E} - ge^\top + \mathcal{L}$ by solving a single linear system in $\mathbf{M}$.

This implies that if we have a RCDD Z-matrix we can compute the stationary of a related matrix. However a priori it is unclear how this allows to compute the stationary of $\mathcal{L}$. The second insight is that if we compute the stationary of $\mathcal{L}'$, e.g. we compute some $y \in \mathcal{R}^n_{>0}$ such that $\mathcal{L}'\mathbf{Y}$ is an Eulerian Laplacian, then $(\mathbf{E} + \mathcal{L})\mathbf{Y}$ is strictly RCDD. Since $\mathcal{L}'\mathbf{Y}$ is Eulerian, $(\mathbf{E} - \mathcal{L} - \mathcal{L}')\mathbf{Y} = ge^\top\mathbf{Y}$ is an all positive matrix which is entrywise less than $\mathcal{L}'$ in absolute value. In other words, removing $ge^\top\mathbf{Y}$ from $\mathcal{L}'\mathbf{Y}$ strictly decreases the absolute value of the off diagonal entries and increases the value of the diagonal entries causing $(\mathbf{E}+\mathcal{L})\mathbf{Y}$ to be strictly RCDD. Consequently, given $y$ we can hope to decrease $e$ to achieve $e'$ to obtain an $\alpha$-RCDD Z-matrix $\mathbf{M}' = (\mathbf{E}' + \mathcal{L})\mathbf{Y}$ where $e' \leq e$.

Combining these insights naturally yields our algorithm, which converges quickly provided we use a sufficiently accurate RCDD linear system solver.

## IV. Eulerian Laplacian Solver

Throughout this section, let $\mathcal{L}$ denote an Eulerian directed Laplacian with $n$ vertices and $m$ edges, and let $\mathbf{U}$ denote the associated undirected Laplacian: $\mathbf{U} \stackrel{\text{def}}{=} \frac{1}{2}(\mathcal{L} + \mathcal{L}^\top)$. We define $\mathcal{T}_{\text{solve}} \stackrel{\text{def}}{=} (nm^{3/4} + n^{2/3}m)(\log n)^3$ to simplify the statements of our runtime bounds.

This section's goal is to proving Theorem 3, showing that we can efficiently solve linear systems in $\mathcal{L}$. We make no attempt to minimize the number of logarithmic factors in this presentation (including e.g. in parameter balancing); we suspect that with careful use of the recent results [41] and [27] the $\log n$ factors can all be eliminated.

**Theorem 3.** *Let $b$ be an $n$-dimensional vector in the image of $\mathcal{L}$, and let $x$ be the solution to $\mathcal{L}x = b$. Then for any $0 < \epsilon \leq \frac{1}{2}$, one can compute in $O(\mathcal{T}_{\text{solve}} \log(1/\epsilon))$ time, a vector $x'$ which with high probability is an approximate solution to the linear system in that $\|x' - x\|_{\mathbf{U}} \leq \epsilon \|b\|_{\mathbf{U}^+}$.*

Our proof is crucially based on the symmetric matrix $\mathbf{X} \stackrel{\text{def}}{=} \mathcal{L}^\top \mathbf{U}^+ \mathcal{L}$. We begin by noting that $\mathbf{X}$ is somewhat well approximated by $\mathbf{U}$:

**Lemma 4.** *$\mathbf{X} \succeq \mathbf{U}$, while $\text{tr}(\mathbf{X}\mathbf{U}^+) = \text{tr}(\mathcal{L}^\top \mathbf{U}^+ \mathcal{L} \mathbf{U}^+) \leq 2(n-1)^2$.*

We use a technique similar to the method of ultrasparsification used in Laplacian solvers, in particular [22]. Our method deviates from much of this previous work in that it is non-recursive and uses only the Woodbury matrix identity (instead of partial Cholesky factorization) to solve the preconditioner. This allows us to obtain a preconditioner with a better relative condition number than $\mathbf{U}$ itself, which can still be applied nearly as efficiently. Details are given in the full version of this paper.

## V. Solving Strictly RCDD Systems

Here we show how to reduce solving strictly RCDD systems using an Eulerian Laplacian solver. We provide and prove Theorem 5 which achieves this goal by using the Eulerian Laplacian system solver presented in Section V. The full version has a corollary specializing this to the $\alpha$-RCDD case.

**Theorem 5.** *Let $\mathbf{A} \in \mathcal{R}^{n \times n}$ be a strictly RCDD $Z$-matrix, let $b \in \mathcal{R}^n$, let $x$ be the solution to $\mathbf{A}x = b$, and let $0 < \epsilon \leq \frac{1}{2}$. Then in $O(\mathcal{T}_{\text{solve}} \log(1/\epsilon))$ time we can compute a vector $x'$ that satisfies $\|x' - x\|_{\frac{1}{2}(\mathbf{A}+\mathbf{A}^\top)} \leq \epsilon \|b\|_{(\frac{1}{2}(\mathbf{A}+\mathbf{A}^\top))^{-1}}$ with high probability.*

## VI. Condition Number Bounds

For our applications (Section VII) we use three quantities to measure the degeneracy or condition number of the matrices involved. These quantities only appear inside logs, so it suffices to relate them up to various polynomial factors. This is done in the full version. Let $\mathbf{W}$ and $s$ be the random walk matrix and stationary distribution associated with a directed Laplacian. The quantities are $\|\mathcal{L}^+\|_2$, the mixing time of the lazy random walk associated with $\mathbf{W}$, and personalized PageRank mixing time. This last quantity is the smallest $k \geq 0$ such that, setting $\beta = \frac{1}{k}$, one has $\|\mathbf{M}_{pp(\beta)} p - s\|_1 \leq \frac{1}{2}$, for all $p \in \Delta^n$; where $\mathbf{M}_{pp(\beta)} = \beta \left(\mathbf{I} - (1-\beta)\mathbf{W}\right)^{-1}$.

## VII. Applications

We now use the algorithms from the previous sections to efficiently solve several of problems of interest, mostly related to computing random walk-related quantities of directed graphs. We emphasize that unlike all prior work for directed graphs, our results have only a polylogarithmic dependence on the condition number—or equivalently—mixing time. We can compute

- $\mathbf{A}^{-1}b$ where $\mathbf{A}$ is any invertible RCDD matrix, without strictness or $Z$-matrix requirements
- $\mathcal{L}^+ b$ where $\mathcal{L}$ is a directed Laplacian and $b \in \mathcal{R}^n$
- personalized PageRank
- the mixing time of a Markov chain—up to various polynomial factors—and its stationary distribution
- hitting times for any particular pair of vertices
- escape probabilities for any triple of vertices
- all pairs commute times via JL sketching

Like Theorem 2, all of our routines will utilize a solver for RCDD linear systems in a black-box manner, giving runtimes of $\widetilde{O}(\mathcal{T}_{\text{solve}}) \leq \tilde{O}(nm^{3/4} + n^{2/3}m)$.


## Acknowledgment

M.B.C., J.K., and A.V.: This material is based upon work supported by the National Science Foundation under Grant No. 1111109.

J.P.: This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374 and by the National Science Foundation under Grant No. 1065125.

R.P.: Part of this work was done while at MIT.

A.S.: This work was supported in part by the Simons Institute for the Theory of Computing, NSF award 1111109, a NSF Graduate Research Fellowship (grant no. 1122374), Microsoft Research New England, and Stanford University.


## References


[1] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu, "Faster algorithms for computing the stationary distribution, simulating random walks, and more," 2016.

[2] A. Sinclair and M. Jerrum, "Approximate counting, uniform generation and rapidly mixing markov chains," *Inf. Comput.*, vol. 82, no. 1, pp. 93–133, 1989.

[3] R. M. Karp, M. Luby, and N. Madras, "Monte-carlo approximation algorithms for enumeration problems," *Journal of algorithms*, vol. 10, no. 3, pp. 429–448, 1989.

[4] M. Jerrum, A. Sinclair, and E. Vigoda, "A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries," *Journal of the ACM (JACM)*, vol. 51, no. 4, pp. 671–697, 2004.

[5] L. Lovász and M. Simonovits, "The mixing rate of markov chains, an isoperimetric inequality, and computing the volume," in *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*. IEEE, 1990, pp. 346–354.

[6] M. Dyer, A. Frieze, and R. Kannan, "A random polynomial-time algorithm for approximating the volume of convex bodies," *Journal of the ACM (JACM)*, vol. 38, no. 1, pp. 1–17, 1991.

[7] S. Vempala, "Geometric random walks: a survey," *Combinatorial and computational geometry*, vol. 52, no. 573-612, p. 2, 2005.

[8] L. Lovász and S. Vempala, "Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006, pp. 57–68.

[9] ——, "Hit-and-run from a corner," *SIAM Journal on Computing*, vol. 35, no. 4, pp. 985–1005, 2006.

[10] Y. T. Lee and S. S. Vempala, "Geodesic walks on polytopes," *arXiv preprint arXiv:1606.04696*, 2016.

[11] N. Alon and V. D. Milman, "$\lambda 1$, isoperimetric inequalities for graphs, and superconcentrators," *Journal of Combinatorial Theory, Series B*, vol. 38, no. 1, pp. 73–88, 1985.

[12] D. Spielman and S. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 835–885, 2014, available at http://arxiv.org/abs/cs/0607105.

[13] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 497–515, 2004.

[14] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 475–486.

[15] L. Orecchia, S. Sachdeva, and N. K. Vishnoi, "Approximating the exponential, the lanczos method and an o (m)-time spectral algorithm for balanced separator," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 1141–1160.

[16] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bulletin of the American Mathematical Society*, vol. 43, no. 4, pp. 439–561, 2006.

[17] O. Reingold, "Undirected connectivity in log-space," *Journal of the ACM (JACM)*, vol. 55, no. 4, p. 17, 2008.

[18] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. ACM, 1995, pp. 388–397.

[19] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[20] L. Lovász, "Random walks on graphs," *Combinatorics, Paul erdos is eighty*, vol. 2, pp. 1–46, 1993.

[21] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, ser. FOCS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 235–244, available at http://arxiv.org/abs/1003.2958. [Online]. Available: http://arxiv.org/abs/1003.2958

[22] ——, "A nearly-m log n time solver for SDD linear systems," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ser. FOCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 590–598, available at http://arxiv.org/abs/1102.4842.

[23] 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*, ser. STOC '13. New York, NY, USA: ACM, 2013, pp. 911–920, available at http://arxiv.org/abs/1301.6628.

[24] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 147–156.

[25] 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.

[26] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, ser. STOC '14. New York, NY, USA: ACM, 2014, pp. 333–342, available at http://arxiv.org/abs/1311.3286.

[27] Y. T. Lee, R. Peng, and D. A. Spielman, "Sparsified cholesky solvers for SDD linear systems," *CoRR*, vol. abs/1506.08204, 2015.

[28] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified Cholesky and multigrid solvers for connection Laplacians," *CoRR*, vol. abs/1512.01892, 2015, available at http://arxiv.org/abs/1512.01892.

[29] S.-H. Teng, "The laplacian paradigm: Emerging algorithms for massive graphs," in *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, ser. TAMC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 2–14.

[30] 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 forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 273–282.

[31] Y. T. Lee, S. Rao, and N. Srivastava, "A new approach to computing maximum flows using electrical flows," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 755–764.

[32] J. Sherman, "Nearly maximum flows in nearly linear time," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 263–269.

[33] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2014, pp. 217–226.

[34] R. Peng, "Approximate undirected maximum flows in $O(m\text{polylog}(n))$ time," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, 2016, pp. 1862–1867.

[35] A. Asadpour, M. X. Goemans, A. Madry, S. O. Gharan, and A. Saberi, "An o (log n/log log n)-approximation algorithm for the asymmetric traveling salesman problem." in *SODA*, vol. 10. SIAM, 2010, pp. 379–389.

[36] N. Anari and S. O. Gharan, "Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp," in *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, 2015, pp. 20–39.

[37] J. A. 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.

[38] A. Mądry, D. Straszak, and J. Tarnawski, "Fast generation of random spanning trees and the effective resistance metric," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2015, pp. 2019–2036.

[39] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, ser. STOC '08. New York, NY, USA: ACM, 2008, pp. 563–568.

[40] Z. Allen-Zhu, Z. Liao, and L. Orecchia, "Spectral sparsification and regret minimization beyond matrix multiplicative updates," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*. ACM, 2015, pp. 237–245.

[41] Y. T. Lee and H. Sun, "Constructing linear-sized spectral sparsification in almost-linear time," in *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, Oct 2015, pp. 250–269.

[42] J. A. Kelner, G. L. Miller, and R. Peng, "Faster approximate multicommodity flow using quadratically coupled flows," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 1–18.

[43] 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.

[44] Y. T. Lee and A. Sidford, "Path finding methods for linear programming: Solving linear programs in $\tilde{O}\sqrt{rank}$ iterations and faster algorithms for maximum flow," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014, pp. 424–433, available at http://arxiv.org/abs/1312.6677 and http://arxiv.org/abs/1312.6713.

[45] M. B. Cohen, A. Madry, P. Sankowski, and A. Vladu, "Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{o}(m^{10/7}$ log W) time," *CoRR*, vol. abs/1605.01717, 2016.

[46] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19.

[47] K. Guo and B. Mohar, "Hermitian adjacency matrix of digraphs and mixed graphs," *ArXiv e-prints*, May 2015.

[48] K. M. Chung, O. Reingold, and S. Vadhan, "S-t connectivity on digraphs with a known stationary distribution," in *Computational Complexity, 2007. CCC '07. Twenty-Second Annual IEEE Conference on*, June 2007, pp. 236–249.

[49] V. V. Williams, "Multiplying matrices faster than coppersmith-winograd," in *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, 2012, pp. 887–898.

[50] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web." 1999.

[51] G. Jeh and J. Widom, "Scaling personalized web search," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 271–279.

[52] D. Fogaras and B. Rácz, "Towards scaling fully personalized pagerank," in *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 2004, pp. 105–117.

[53] R. Andersen, C. Borgs, J. Chayes, J. Hopcraft, V. S. Mirrokni, and S.-H. Teng, "Local computation of pagerank contributions," in *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 2007, pp. 150–165.

[54] P. A. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri, "Fast-ppr: scaling personalized pagerank estimation for large graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1436–1445.

[55] S. I. Daitch and D. A. Spielman, "Faster approximate lossy generalized flow via interior point algorithms," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, ser. STOC '08. New York, NY, USA: ACM, 2008, pp. 451–460.