Finding Effective Support-Tree Preconditioners

Bruce M. Maggs[1] Gary L. Miller[1] Ojas Parekh[2] R. Ravi[3] Shan Leung Maverick Woo[1]

[1]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213
{bmm,glmiller,maverick}@cs.cmu.edu

[2]Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322
ojas@mathcs.emory.edu

[3]Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213
ravi@cmu.edu

Abstract

In 1995, Gremban, Miller, and Zagha introduced support-tree preconditioners and a parallel algorithm called support-tree conjugate gradient (STCG) for solving linear systems of the form $A\boldsymbol{x} = \boldsymbol{b}$, where A is an $n \times n$ Laplacian matrix. A Laplacian is a symmetric matrix in which the off-diagonal entries are non-positive, and the row and column sums are zero. A Laplacian A with $2m$ non-zeros can be interpreted as an undirected positively-weighted graph G with n vertices and m edges, where there is an edge between two nodes i and j with weight $c((i,j)) = -A_{i,j} = -A_{j,i}$ if $A_{i,j} = A_{j,i} < 0$. Gremban *et al.* showed experimentally that STCG performs well on several classes of graphs commonly used in scientific computations. In his thesis, Gremban also proved upper bounds on the number of iterations required for STCG to converge for certain classes of graphs. In this paper, we present an algorithm for finding a preconditioner for an arbitrary graph $G = (V, E)$ with n nodes, m edges, and a weight function $c > 0$ on the edges, where w.l.o.g., $\min_{e \in E} c(e) = 1$. Equipped with this preconditioner, STCG requires $O(\log^4 n \cdot \sqrt{\Delta/\alpha})$ iterations, where $\alpha = \min_{U \subset V, |U| \leq |V|/2} c(U, V \backslash U)/|U|$ is the minimum edge expansion of the graph, and $\Delta = \max_{v \in V} c(v)$ is the maximum incident weight on any vertex. Each iteration requires $O(m)$ work and can be implemented in $O(\log n)$ steps in parallel, using only $O(m)$ space. Our results generalize to matrices that are symmetric and diagonally-dominant (SDD).

1 Introduction

Perhaps the most common and time consuming task that arises in scientific computing applications is to solve a large linear system of the form $A\boldsymbol{x} = \boldsymbol{b}$, where A is a known $n \times n$ matrix, $\boldsymbol{b}$ is a known $n \times 1$ vector in the column space of A, and $\boldsymbol{x}$ is an unknown $n \times 1$ vector. These systems arise, for example, during the discretization of differential or integral equations describing some physical system. The matrix A typically represents the structure of the physical system, while the vector $\boldsymbol{b}$ represents some boundary condition. It is not uncommon for a single application to solve many linear systems involving the same matrix A, but with different boundary conditions $\boldsymbol{b}$. Hence, a distinction can be made between the time spent preprocessing the matrix A, and the time solving linear systems involving A. Our objective is to minimize the time to solve such a system while also minimizing the time required for preprocessing.

The matrices that arise in describing physical systems are generally not arbitrary but are often sparse and display structural properties (mirroring structures in physical systems) that can be exploited. The classic textbook by Golub and Van Loan [13] provides examples in many contexts. The theory community has focused on developing approaches for structured systems, such as Laplacians, which are described below, in the context of two general approaches, *direct* and *iterative*.

Direct approaches include a class of algorithms known as nested-dissection. The work in this area was pioneered by Lipton, Rose, and Tarjan [20]. Nested dissection works well for graphs with small separators, such as trees and planar graphs. Lipton *et al.* [20], for example, showed that any n-node planar system can be solved in $O(n^{1.5})$ time. General-purpose direct methods are typically not attractive, however, for solving systems based on sparse but highly connected graphs like expanders.

An important sub-class of matrices is the class that are symmetric and diagonally-dominant (SDD); our notion of diagonal-dominance is that each diagonal entry is at least as large as the sum of the magnitudes of the entries in the corresponding row or column. SDD matrices arise in many natural applications (see [3, 23]). Iterative methods such as conjugate gradient can be applied to these systems. Our goal is to address this class of matrices; however, we restrict our attention to Laplacian matrices since Gremban [14] shows that an SDD system can be solved by solving a Laplacian system twice as large. A Laplacian matrix can be interpreted as a weighted adjacency matrix of an undirected graph with nonnegative edge weights (see Section 2.2).

Our approach is to design preconditioners for accelerating the convergence of iterative methods like conjugate gradient for solving systems involving Laplacians. The notion of a preconditioner is described in detail in Section 2.1. As we shall see, the number of iterations required to solve such a system using a preconditioned iterative method can be as low as polylogarithmic, where each iteration can be implemented in logarithmic time using linear work and space. This is the case, for example, for approximately-uniformly-weighted expander graphs. There has already been extensive work in this area, and it is described below.

1.1 Previous Work

We state the following results in terms of preconditioners for the Laplacian of a graph with n vertices and m edges. As mentioned in the previous section, the results also hold for SDD systems. Strictly speaking, the bounds to follow apply to finding a solution $\boldsymbol{x}$ such that $\|A\boldsymbol{x}-\boldsymbol{b}\| \leq \epsilon\|\boldsymbol{b}\|$, at the cost of an additional multiplicative factor of $O(\log((p\log n)/\epsilon))$ (where each entry in A is specified with p bits of precision), which we have omitted for simplicity.

In seminal work, Vaidya [26, 9, 10] showed how to use combinatorial techniques to construct preconditioners based on sparse spanning subgraphs of the underlying graph of the matrix. Vaidya's preconditioners allow a Laplacian system to be solved with at most $O(n^{1.75})$ work for any bounded-degree weighted graph and $O(n^{1.2})$ work for any weighted planar graph. Reif [22] then analyzed a recursive variant proposed by Vaidya and improved the bound on work for sparse graphs. Boman *et al.* [6] analyzed an extension proposed by Vaidya [26] of his preconditioners to all SDD systems. More recently, Boman and Hendrickson [7] demonstrated that the low-average-stretch spanning trees designed by Alon *et al.* [1] result in preconditioners with a work bound of $O(m^{1.5+o(1)})$ for any weighted graph. Although we do not state it this way, the algorithm described in this paper has a work bound of $O(mn^{1/2})$ times polylogarithmic factors, and hence can be viewed as an improvement on [7].

Gremban, Miller, and Zagha [15] and Gremban [14] considered a different kind of graph-based preconditioner. They demonstrated that the support graph of a good subgraph preconditioner need not be subgraphs of the graph represented by A. Gremban and Miller presented a way to analyze such extended subgraph preconditioners which have additional nodes. They called these *support tree preconditioners*. In particular, they designed support tree preconditioners for the Laplacians of meshes (solving systems based on regular uniform-weight d-dimensional meshes in $O(m\sqrt{dn^{1/d}\log n})$ time), such that the leaves of the support trees correspond precisely to the nodes of the original graph. In order to approximate the topology of the graph, their trees are hierarchically constructed by recursively partitioning the graph. Promising experimentally results are described in [15].

All of the work described above has been superseded by two papers by Spielman and Teng [24, 25] and a paper by Emek, Elkin, Spielman, and Teng [12]. The first paper [24] shows that building preconditioners by adding edges to the Alon *et al.* trees, and using recursion, results in a method that requires $O(m^{1.31})$ work (times polylogarithmic factors). The second paper [25] introduces techniques for "sparsifying" a graph that can be used to reduce the size of the preconditioner, reducing the total work to $O(m+n2^{O(\sqrt{\log n\log\log n})})$ times polylogarithmic factors. Finally, the third paper [12] introduces a new construction of low-average-stretch spanning trees. Replacing the trees of Alon et al. [1] with the new trees improves the performance of the algorithm in the second paper to $O(m)$ work (times polylogarithmic factors). The algorithm in [12] can also be parallelized, and as there is a lower bound of $\Omega(m)$ on work, cannot be improved much asymptotically.

1.2 Our results

The results in this paper are not an improvement on the results of [12], and indeed only match their results in special cases. Written in terms of m and n, our work bound is inferior. Nevertheless, we believe that the paper makes a contribution by presenting the first algorithm and analysis for arbitrary Laplacians based on the support-tree preconditioner approach of Gremban et al. [15], which is the only approach to introduce nodes in the preconditioner that are not present in the graph. We also point out an interesting connection between preconditioners and the hierarchical decomposition trees of weighted undirected graphs, recently introduced by Räcke [21]. In particular, we show that they can be used directly as preconditioners.

Our main result is that any Laplacian system with n nodes and m edges and weight function c can be solved in $O(\log^4 n \cdot \sqrt{\Delta/\alpha})$ iterations, where $\alpha = \min_{U \subset V, |U| \leq |V|/2} c(U, V \backslash U)/|U|$ is the minimum edge expansion of the graph, $\Delta = \max_{v \in V} c(v)$ is the maximum incident weight on any vertex, and w.l.o.g., $\min_{e \in E} c(e) = 1$. Each iteration requires $O(m)$ work and can be implemented in $O(\log n)$ steps in parallel, using only $O(m)$ space. As mentioned above, the bound on the number of iterations can also be written as $O(\sqrt{n} \log^4 n)$.

One caveat about our work is that all of the previous results cited above include the cost of constructing the preconditioner in the bound on work required to solve the system. Our bound does not. A preconditioner, however, need only be constructed once to solve a system, $A\boldsymbol{x} = \boldsymbol{b}$ for different values of $\boldsymbol{b}$, and, as noted in the introduction, this is a problem of practical interest.

Our paper begins by analyzing the effectiveness of Räcke's trees as preconditioners. With these preconditioners, STCG requires $O(\log^3 n \cdot \sqrt{\Delta/\alpha})$ iterations (rather than $O(\log^4 n \cdot \sqrt{\Delta/\alpha})$), but as Räcke noted, no polynomial-time algorithm is known for constructing his trees. However, a pair of recent papers Bienkowski, Korzeniowski, and Räcke [5] and Harrelson, Hildrum, and Rao [17] showed how to build comparable trees in polynomial time, *assuming the maximum edge weight is polynomial.* In Appendix A, we show that the trees produced by the algorithm in [5] can be used as preconditioners that lead to the $O(\log^4 n \cdot \sqrt{\Delta/\alpha})$ bound on the number of iterations.

2 Background

Let us first develop the background material and context in which we will present our work. The reader familiar with iterative methods and support theory may choose to consult this section on demand.

2.1 Preconditioners

Suppose we are solving the system $A\boldsymbol{x} = \boldsymbol{b}$. When A is sparse, iterative methods are preferable to direct ones because direct methods have difficulty exploiting the sparsity of A in general. An iterative method is one that produces a sequence of iterates, or guesses, $\{ \boldsymbol{x}^{(i)} \}$ that converges to a solution $\boldsymbol{x}$. Typically there are two components to designing such an approach, (1) providing an update step which maps $\boldsymbol{x}^{(i)}$ to $\boldsymbol{x}^{(i+1)}$ and (2) a proof that the iterates indeed converge (rapidly).

Example: The RF Method The basis of iterative methods is the RF method [16], which illustrates the idea behind such approaches very well. For the sake of exposition, we assume we are dealing with nonsingular matrices[1]; however, this is not a requirement imposed by iterative methods.

The update step for the RF method is simply $\boldsymbol{x}^{(i+1)} := \boldsymbol{x}^{(i)} - (A\boldsymbol{x}^{(i)} - \boldsymbol{b})$. The intuition behind is that $A\boldsymbol{x}^{(i)} - \boldsymbol{b}$ represents some notion of error. Indeed, it is desirable to reduce error. However, one may notice that the RF method's implementation of this idea has a shortcoming—thinking of A as a linear mapping, the error $A\boldsymbol{x}^{(i)} - \boldsymbol{b}$ lies in the range of A, whereas the iterates $\boldsymbol{x}^{(i)}$ are sought in the domain of A. Preconditioning is an attempt to address this discrepancy, and it does so by providing a mapping back from the range space to the iterate space. Thus a preconditioned RF update step becomes $\boldsymbol{x}^{(i+1)} := \boldsymbol{x}^{(i)} - B^{-1}(A\boldsymbol{x}^{(i)} - \boldsymbol{b})$, where B^{-1} is a *preconditioner* for the system $A\boldsymbol{x} = \boldsymbol{b}$.

[1]In fact, Laplacians are always singular matrices since the row sums are all zero. However, this very particular kind of singularity can be handled with a little care. See Gremban [14] for the details.

We may also interpret preconditioning as solving the system $B^{-1}A\boldsymbol{x} = B^{-1}\boldsymbol{b}$ rather than $A\boldsymbol{x} = \boldsymbol{b}$. Intuitively, we would like our preconditioner B^{-1} to approximate A^{-1} well, and we see that setting $B^{-1} := A^{-1}$ immediately renders a solution to the system. Of course, this is impractical as it requires inverting A—an operation which we are trying to avoid in the first place. (A^{-1} may *not* be sparse.)

In choosing a preconditioner B^{-1} for the system $A\boldsymbol{x} = \boldsymbol{b}$, our goal is to strike a balance between the total number of iterations and the time required per iteration. Note that each update step requires us to compute $\boldsymbol{x}^{(i)} - B^{-1}(A\boldsymbol{x}^{(i)} - \boldsymbol{b})$. As we may precompute $B^{-1}\boldsymbol{b}$, the computation time is dominated by the time required for the sparse matrix-vector multiplication $\boldsymbol{c} := A\boldsymbol{x}^{(i)}$ and determining $\boldsymbol{z} = B^{-1}\boldsymbol{c}$, which we may re-state as finding a solution $\boldsymbol{z}$ of the system $B\boldsymbol{z} = \boldsymbol{c}$. In fact, this is all we require of B, hence B need not be invertible; in the sequel we refer to either B or B^{-1} as a preconditioner for A. In particular, if we can solve the system $B\boldsymbol{z} = \boldsymbol{c}$ in $O(n)$ time, then each iteration takes a total of $O(n + m)$ time. On the other hand, bounds on the total iteration count depend on the particular iterative method used. We postpone a more detailed discussion of bounding the iteration count, except to remark that the bound we employ roughly measure how well B approximates A.

In what follows, we first present our notation and state some preliminary definitions. We then review results on convergence bounds for preconditioned iterative methods. We will close this section by presenting some techniques for bounding these convergence rates.

2.2 Notation and Preliminary Definitions

For typographic clarity, the vectors in this paper are typeset in boldface, *e.g.*, $\boldsymbol{x}$. The i-th coordinate of $\boldsymbol{x}$ is specified as $\boldsymbol{x}_i$. Similarly, subscripts are also used to specify elements of a matrix, *e.g.*, $A_{i,j}$.

The *edge-vertex incidence matrix* of a graph $G = (V, E)$ is a $|E| \times |V|$ matrix Γ of elements $\{ -1, 0, 1 \}$. For each edge $e = (u, v)$, we set $\Gamma_{e,u}$ to -1 and $\Gamma_{e,v}$ to 1 (the ordering of u and v can be arbitrarily fixed). All other entries are set to 0. When G is weighted, let $c(e)$ be the *weight* of an edge e. We introduce a $|E| \times |E|$ diagonal matrix W, where $W_{e,e} = c(e)$. The *Laplacian* of G is defined to be $\Gamma^{\mathsf{T}} W \Gamma$.

Let A be $\Gamma^{\mathsf{T}} W \Gamma$. Clearly, A is a square matrix of size $|V| \times |V|$. Since A can be decomposed into the form $\Gamma^{\mathsf{T}} W \Gamma$, A is symmetric positive semidefinite (SPSD). It is not hard to verify that $A_{i,j} = A_{j,i} = -W_{ij,ij}$ for $i \neq j$, and that the rows and columns of A sum to 0.

For convenience, we use G to refer to the Laplacian of a graph G, relying upon the context for differentiation. We also define the *incident weight* of a vertex $u \in V$ to be $c(u) = \sum_{(u,v)\in E} c((u, v))$ and also generalize this to a set of vertices. Finally, we use $\{m \otimes n\}$ as a shorthand for $\{ (i, j) \mid i \in \{ 1, \ldots, m \}, j \in \{ 1, \ldots, n \} \}$.

2.3 Support Theory

In this section, we present the definitions and concepts necessary to link the task of bounding the iteration count of an iterative method to the design of a (combinatorial) preconditioner. Note that A and B have the same size within this subsection.

Definition 2.1 *The* support *required by a matrix B for a matrix A is defined as*

$$\sigma(A/B) := \min\{ \tau \in \mathbb{R} \mid \forall \boldsymbol{x},\ \boldsymbol{x}^{\mathsf{T}}(\tau B - A)\boldsymbol{x} \geq \boldsymbol{0} \}.$$

We say that $\sigma(A/B)$ is ∞ or $-\infty$ in the cases when the minimum is taken over an empty set or is unbounded respectively. To avoid such complications, we assume that both A and B are positive semidefinite (PSD) and that $\text{null}(A) = \text{null}(B)$, which is certainly true for graph Laplacians.

The notion of support has several interpretations related to electrical networks, graph embeddings, and graph connectivity to which we will appeal through the course of this paper. Support numbers enjoy many nice properties which are proven and catalogued in Boman and Hendrickson [8].

Lemma 2.2 (Transitivity) $\sigma(A/C) \leq \sigma(A/B) \cdot \sigma(B/C)$

Definition 2.3 *The* generalized condition number *of a pair of psd matrices* (A, B) *such that* $\text{null}(A) = \text{null}(B)$ *is defined to be* $\kappa(A, B) := \sigma(A/B) \cdot \sigma(B/A)$.

We acknowledge that this is not that standard definition of $\kappa(A,B)$ (say, as in [16]); however, given our restrictions on A and B, this is an equivalent one. It is well-known that Preconditioned Conjugate Gradient (PCG) on a system $A\boldsymbol{x} = \boldsymbol{b}$ with the preconditioner B requires at most $\sqrt{\kappa(A,B)}\log(1/\epsilon)$ iterations [16] to find a solution $\boldsymbol{x}$ such that $\|A\boldsymbol{x} - \boldsymbol{b}\| \leq \epsilon\|\boldsymbol{b}\|$.

We conclude this section by presenting the main combinatorial tool upon which both Vaidya [26] and Gremban and Miller [14] relied. An *embedding* of a *guest* graph G into a *host* graph H is a mapping, $\mathcal{P}$ from the edges of G to simple paths in H. For an edge $g \in E(G)$, $|\mathcal{P}(g)|$ refers to the number of edges in the path $\mathcal{P}(g)$. This quantity is often called the *dilation* of g, or $\text{dil}(g)$. For an edge $h \in E(H)$, we call $\text{cong}(h) := \sum_{\{\, g\in E(G) \,|\, h\in\mathcal{P}(g)\,\}} w_G(g)/w_H(h)$ the *congestion* of h.

Lemma 2.4 (Congestion-Dilation) *Given an embedding $\mathcal{P}$ from G to H,*

$$\sigma(L(G)/L(H)) \leq \left(\max_{h\in E(H)} \text{cong}(h)\right)\left(\max_{g\in E(G)} \text{dil}(g)\right).$$

We note that a proof of the Congestion-Dilation Lemma is given by Bern *et al.* [4]. A more detailed and general account of the material here can also be found in Boman and Hendrickson [8] or Gremban [14].

2.4 Support Tree Conjugate Gradient

Since a support tree preconditioner is defined over a potentially larger graph than the one for which it is created, we need tools that are capable of analyzing the support between matrices of different sizes.

Let A and $B = \begin{pmatrix} T & U \\ U^\mathsf{T} & W \end{pmatrix}$ be SDD matrices such that T is nonsingular and both A and W are $n \times n$ matrices. A case of particular interest is when both A and B are Laplacians, and B corresponds to that of a tree with T representing the connectivity among the internal nodes of the tree, U representing the connectivity between the internal nodes and the leaves, and W a diagonal matrix representing the degrees of the leaves, which are identified with the vertices of A. In this case we say B is a support tree for A. Gremban's thesis describes an extension of PCG, called Support Tree Conjugate Gradient (STCG), which is designed to handle the case in which we seek to precondition A with the potentially larger matrix B [14]. In particular, he demonstrated the following.

Lemma 2.5 *Given A and B as defined above, if $Q = W - U^\mathsf{T}T^{-1}U$ then STCG on A using B as a preconditioner takes at most $\sqrt{\kappa(A,Q)}\log(1/\epsilon)$ iterations.*

By convention, Q is called the Schur complement of B (with respect to W). Viewed as a graph, Q represents the connectivity among the vertices of W in B. But as a matrix, it can have a much higher density than B. STCG is in fact designed to efficiently simulate invoking PCG with Q as a preconditioner. The latter requires solving a system of the form $Q\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$ during each iteration, whereas STCG solves for $\boldsymbol{x}$ by solving $B\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{x}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{b}\end{pmatrix}$ and simply discarding $\boldsymbol{y}$.

Proposition 2.6 *If $B\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{x}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{b}\end{pmatrix}$, then $Q\boldsymbol{x} = \boldsymbol{b}$.*

Since B is a tree, the system in Proposition 2.6 can be directly solved in both time and space linear to the number of leaves.. By Lemma 2.5 we may bound the convergence rate of STCG on A and B by bounding $\sigma(A/Q)\cdot\sigma(Q/A)$; however, in our analysis we will find it more convenient to refer directly to the structure of larger Laplacian B, so we introduce the following definitions.

Let A and B be defined as above, and let $\bar{A} = \begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & A\end{pmatrix}$ be the same size as B. The matrix $\bar{A}$ allows us to compare A and B and leads to a natural definition of $\sigma(A/B)$.

Definition 2.7 *The support of B for A, $\sigma(A/B)$ is $\sigma(\bar{A}/B)$.*

The proposition below, which follows from Proposition 6.1 in Boman and Hendrickson [8], demonstrates that the above definition is consistent.

Proposition 2.8 *Given the definitions above, if T is nonsingular then $\sigma(\bar{A}/B) = \sigma(A/Q)$.*

We may also think of $\bar{A}$ as a Laplacian, albeit one whose graph has isolated vertices, which gives us an analogue of Lemma 2.4 that we can use to directly bound $\sigma(A/B)$.

Corollary 2.9 (Extended Congestion-Dilation) *Given an embedding $\mathcal{P}$ of the edges of A as paths in a potentially larger graph B, we have $\sigma(A/B) \leq (\max_{h \in E(B)} \operatorname{cong}_{\mathcal{P}}(h)) \cdot (\max_{g \in E(A)} \operatorname{dil}_{\mathcal{P}}(g))$.*

We must take a slightly different approach in defining $\sigma(B/A)$ due to a disparity in the null spaces of $\bar{A}$ and B.

Definition 2.10 *The support of A for B, $\sigma(B/A)$ is*

$$\min\{\tau \mid \forall \boldsymbol{x},\, \tau \cdot \boldsymbol{x}^{\mathsf{T}} A \boldsymbol{x} \geq \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix}^{\mathsf{T}} B \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix},\ \textit{where } \boldsymbol{y} = -T^{-1} U \boldsymbol{x}\}.$$

Although the definition above may not seem as natural as Definition 2.7, it does capture the notion of supporting Q with A.

Proposition 2.11 *We have $\sigma(B/A) = \sigma(Q/A)$.*

3 Laplacians as Circuits

The lack of techniques to cleanly analyze the support required for a Laplacian by a smaller Laplacian has been a major difficulty in the past and have restricted researchers to analyze only simple support graph preconditioners.[2] In this section, we will explore the interpretation of a Laplacian as an electrical circuit and relate the support bounds presented in the previous sections to power dissipation, which will prove to be useful in our analysis of Räcke's decomposition tree (by no means a simple object) as a preconditioner. For a more thorough account of this interpretation, one ought to consult Doyle and Snell [11] or Gremban [14], where one can find some of the results to follow. We defer longer proofs to the appendices.

3.1 Current and Power

We can view an edge-weighted graph G as a resistive network by replacing the edges with wires and interpreting the weight of each edge as the conductance—the reciprocal of the resistance—of the corresponding wire. A node in G will then correspond to either an internal *junction* or an external *terminal.* When the distinction is not important, we will refer to both of these as *nodes* for simplicity.

Lemma 3.1 and Theorem 3.2 establish the electrical interpretation of Laplacians (as defined in Section 2.2), which allows us to switch between a weighted graph and its equivalent resistive network to make our theory more intuitive.

Lemma 3.1 (Net Current Flow) *Suppose an $n \times n$ matrix A is the Laplacian of a resistive network G with n nodes. If $\boldsymbol{y}$ is the n-vector specifying the voltage at each node of G, then $A\boldsymbol{y}$ is the n-vector representing the net current flow at each node.*

Theorem 3.2 (Power Dissipation) *Suppose an $n \times n$ matrix A is the Laplacian of a resistive network G with n nodes. If $\boldsymbol{y}$ is a n-vector specifying the voltage on each node, then $\boldsymbol{y}^{\mathsf{T}} A \boldsymbol{y}$ is the total power dissipated by G.*

[2]Bern *et al.* [4] have also identified this difficulty and proposed new tools. The line of work by Spielman and Teng [24, 25] and the most recent development of Elkin *et al.* [12] all use subgraphs and so the Laplacians are of the same size.

3.2 Power and Support

Given the intepretation of the previous section, we may think of $\sigma(A/B)$ in a new light.

Proposition 3.3 *The support of a Laplacian B for a Laplacian A, $\sigma(A/B)$, is the minimum number such that for all $\tau \geq \sigma(A/B)$, the circuit τB dissipates as much power as the circuit A under any voltage settings on the nodes.*

Proof Definition 2.1 implies $\sigma(A/B) = \min\{\tau \mid \forall \boldsymbol{x}, \tau \cdot \boldsymbol{x}^{\mathsf{T}} B \boldsymbol{x} \geq \boldsymbol{x}^{\mathsf{T}} A \boldsymbol{x}\}$, where by the previous section we may interpret $\boldsymbol{x}$ as voltage settings and $\boldsymbol{x}^{\mathsf{T}} M \boldsymbol{x}$ as the power dissipated by a circuit M under the settings $\boldsymbol{x}$. ■

We may also extend this interpretation to the results of Section 2.4, in which we compare two circuits with different numbers of nodes. In particular, we will be interested in the case when we wish to support a circuit represented by $B = \begin{pmatrix} T & U \\ U^{\mathsf{T}} & W \end{pmatrix}$ with a smaller circuit A, where the nodes of A are identified with those of W, the terminals of B.

For any voltage settings at the terminals of B, Kirchhoff's Laws dictate a set of naturally ocurring voltages for the junctions such that they have a net current flow of 0. Defintion 2.10 yields the following.

Corollary 3.4 *If the circuit τA dissipates at least as much power as the circuit B under any voltage setting in which the terminal voltages in B match the voltages in A and the internal voltages in B are determined by Kirchhoff's Laws, then $\sigma(B/A) \leq \tau$.*

Proof Let $\boldsymbol{x}$ be the voltage settings applied at the terminals of B and let the voltages at the junctions be determined by Kirchhoff's Laws. Since the net current flow at each of the junctions must be 0 (conservation of current), by Lemma 3.1, we have $B \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{b} \end{pmatrix}$ where $\boldsymbol{b}$ specifies the current flow at the terminals. Therefore, $\boldsymbol{y} = -T^{-1} U \boldsymbol{x}$ and the result now follows from Definition 2.10. ■

We finish the section with a new tool that will be useful in applying the above to our analysis of Räcke's decomposition tree as a Support Tree.

Lemma 3.5 (Power Charging) *Consider a complete (m, n) bipartite conductive network G containing conductors $c_{i,j}$ for $(i, j) \in \{m \otimes n\}$. Let the conductance of $c_{i,j}$ be $\alpha_i \cdot \beta_j$ such that $\sum_{i=1}^{m} \alpha_i = \delta = \sum_{j=1}^{n} \beta_j$ with nonnegative α_i's and β_j's. Also, let $\boldsymbol{v}$ and $\boldsymbol{w}$ be any m-vector and n-vector respectively. When applying voltages $\boldsymbol{v}$ and $\boldsymbol{w}$ on the left and right hand side of G respectively, the total power consumed by G is no less than a unit conductor across voltages $\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i$ and $\sum_{j=1}^{n} \beta_j \boldsymbol{w}_j$. Mathematically,*

$$\left(\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i - \sum_{j=1}^{n} \beta_j \boldsymbol{w}_j \right)^2 \leq \sum_{(i,j) \in \{m \otimes n\}} \alpha_i \beta_j (\boldsymbol{v}_i - \boldsymbol{w}_j)^2$$

4 Analyzing Räcke's Decomposition Tree

The focus of our paper is on analyzing a decomposition tree introduced by Räcke [21] for oblivious routing and proving that it works well as a preconditioner for graph Laplacians.

Let G be the n-node graph Laplacian for which a preconditioner is sought. Räcke [21] described how a decomposition tree T can be constructed from G. To analyze the performance of STCG on G preconditioned with T, from Lemma 2.5, we know that this amounts to bounding $\kappa(G, T)$. As shown in the following "table of content", our attempt to bound $\kappa(G, T)$ involves several steps and will be explained in the referred sections.

$$\kappa(G,T) \underset{\text{Def. 2.3}}{=} \prod \begin{cases} \sigma(G/T) \underset{\text{Cor. 2.9}}{\leq} \prod \begin{cases} \text{congestion} \leq 1 & \text{Section 4.2} \\ \text{dilation} = O(\log n) & \text{Section 4.2} \end{cases} \\ \sigma(T/G) \underset{\text{Lem. 2.2}}{\leq} \prod \begin{cases} \sigma(T/RC(T)) \leq 1 \text{ by electrical argument} & \text{Section 4.4} \\ \sigma(RC(T)/G) \underset{\text{Lem. 2.4}}{\leq} \prod \begin{cases} \text{congestion} = O(\log^3 n) & \text{Section 4.6} \\ \text{dilation} = O(F(G)\log n) & \text{Section 4.6} \end{cases} \end{cases} \end{cases}$$

(Expressions $RC(T)$ and $F(G)$ will be defined in Sections 4.3 and 4.5 respectively.)

4.1 Räcke's Decomposition Tree

We first review Räcke's tree construction and summarize the relevant notations.

Given a graph $G = (V, E)$, any laminar decomposition of G naturally specifies a decomposition tree $T = (V_T, E_T)$ in which every node $v_t \in V_T$ corresponds to a cluster $S_{v_t} \subseteq V$. Each leaf in T corresponds to a vertex in G. The root of T corresponds to V. The tree node v_t is a child of u_t iff the cluster S_{v_t} is contained within S_{u_t} and *not* any other cluster S_{w_t} that is also contained within S_{u_t}.

By first building T using G and then computing a randomized embedding of T back into G, Räcke's routing algorithm simply routes the requests on T instead. The intermediate locations of the actual path to be taken on G is specified by the randomized embedding, whereas the route between intermediate locations are specified by the solutions to a set of concurrent multicommodity flow problems (CMCFPs). Note that T can be computed using G alone and hence it is computed only once. Each CMCFP in turn depends on T only and can also be solved in a preprocessing stage.

Räcke introduced two connectivity characteristics to measure the quality of a decomposition. Let $\text{cap}(X, Y) = \sum_{\{(x,y)\in E \,|\, x\in X, y\in Y\}} c((x, y))$ be the *capacity* between the set $X \subseteq V$ and $Y \subseteq V$. We will use $\text{out}(X)$ as a shorthand for $\text{cap}(X, V\backslash X)$.

Definition 4.1 *Consider a non-root node v_t in the decomposition tree T. The weight of the edge connecting v_t to its parent is defined to be* $\text{out}(S_{v_t})$.

Let V_t^l denote all the nodes in T at level l, with the level of the root defined to be 0. The *level l-decomposition* of G corresponds to the clusters specified by the nodes in V_t^l. With respect to a fixed T, the *weight function* $w_l(X)$ is defined to be $\text{cap}(X, V) - \sum_{v_t\in V_t^l} \text{cap}(X \cap S_{v_t}, S_{v_t})$ for any $X \subseteq V$. Intuitively, $w_l(X)$ is the sum of the weights of all the edges with at least one end-point in X that cross the boundary of the level l-decomposition of G.

Definition 4.2 *For a level l node v_t in T, let the* bandwidth ratio λ_{v_t} *and the* weight ratio δ_{v_t} *be*

$$\lambda_{v_t} = \max_{\substack{U\subset S_{v_t} \\ |U|\leq|S_{v_t}|/2}} \left(\frac{\text{out}(U)}{\text{cap}(U, S_{v_t}\backslash U)}\right) \quad \textit{and} \quad \delta_{v_t} = \max_{\substack{U\subset S_{v_t} \\ |U|\leq|S_{v_t}|/2}} \left(\frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t}\backslash U)}\right) \quad \textit{respectively.}$$

The construction algorithm provided by Räcke guarantees the following three properties.

Theorem 4.3 (Theorem 9, [21]) *The height of the decomposition tree is $O(\log n)$. Furthermore, for any node v_t in the tree, both λ_{v_t} and δ_{v_t} are $O(\log n)$.*

4.2 Bounding $\sigma(G/T)$

To bound $\sigma(G/T)$, it suffices to inspect the embedding of G into T. The dilation is bounded by $O(\log n)$ since this is the diameter of T as in Theorem 4.3. As for the congestion, consider an edge $(x, y) \in E$. It corresponds to a particular leaf-to-leaf path in T that only uses the parent edge of a node v_t iff (x, y) is a boundary edge of S_{v_t}. Now consider each of the tree edge (u_t, v_t) on this path. Let u_t be the parent node. The weight assigned to (u_t, v_t) is $\text{out}(S_{v_t})$ as in Definition 4.1. Therefore, (u_t, v_t) has sufficient weight reserved for (x, y) and the congestion on it cannot exceed 1. By Corollary 2.9, we have

$$\sigma(G/T) = O(\log n). \tag{4.1}$$

4.3 The Räcke Complement

In an initialization phase, Räcke constructs a CMCFP in S_{u_t} for each u_t in T. Suppose u_t is a level $(l-1)$ node with d children $v_{t1}, \dots, v_{td}$. In the CMCFP for u_t, there are $|S_{u_t}|^2$ commodities $f_{u,v}$ for each pair of $u, v \in S_{u_t}$. The source and the sink of $f_{u,v}$ are u and v respectively and the demand is

$$d_{u,v} = \frac{w_{l+1}(v)}{w_{l+1}(S_{v_{ti}})} \cdot \mathrm{out}(S_{v_{tv}}) \cdot \frac{w_l(u)}{w_l(S_{u_t})}, \tag{4.2}$$

where $S_{v_{tv}}$ denotes the level l cluster that contains v. The flows are restricted to use only edges inside S_{u_t} and must respect the edge weights as link capacities.

Let q be the throughput fraction of a solution to the CMCFP, *i.e.*, q is the minimum, over all commodities, of the fraction of the commodity's demand that is actually met by the solution. An optimal solution maximizes q.

We view the CMCFP corresponding to u_t as a complete graph K_{u_t} on the vertex set S_{u_t} where the weight of each edge (u,v) inside S_{u_t} is the demand that will be sent between u and v, *i.e.*, $d_{u,v} + d_{v,u}$. (Note that these two terms can have different values.) We call the overlapping of the $|V_T|$ complete graphs K_{u_t}s the *Räcke complement* of T, denoted $RC(T)$.

Definition 4.4 *The* Räcke complement *of T is a complete graph on the vertex set V. The weight of an edge (u,v) is the sum of its weight in each of the K_{u_t} containing it.*

4.4 Bounding $\sigma(T/RC(T))$

The reason we introduced the analytical tools based on electrical networks in Section 3 is because we need a manageable way to bound the support required for a matrix by a smaller matrix (T by $RC(T)$). We now use Corollary 3.4 to show that $\sigma(T/RC(T)) \leq 1$.

First fix an arbitrary n-vector $\boldsymbol{x}$. We apply $\boldsymbol{x}$ as the voltages at the terminals of T and let the voltages at the junctions be the naturally-occurring voltages as determined by Kirchhoff's Laws. We would like to bound the power dissipation of T under these voltage settings by that of $RC(T)$ under $\boldsymbol{x}$. However, we do not have a clean method to directly obtain a bound for the former. Our solution is to pick the voltages at the junctions at our choice. Apply the Dirichlet Principle (see [11, p. 64]), which says that enforcing any junction voltages other than the naturally-occurring ones can only increase total power dissipation of T, this provides us with an upperbound.

The actual choice of voltages is inspired by Räcke's construction. Consider a level $(l-1)$ node u_t. In Räcke's construction, the node u_t will be simulated by the vertex u among all the vertices in S_{u_t} with probability $w_l(u)/w_l(S_{u_t})$. We will set the voltage of u_t to be

$$\sum_{u \in S_{u_t}} \frac{w_l(u)}{w_l(S_{u_t})} \boldsymbol{x}_u. \tag{4.3}$$

At this point, notice that the voltages at all the junctions ($\boldsymbol{x}$) and terminals (our choices) are all specified.

Consider a level $(l-1)$ junction u_t with d children $v_{t1}, v_{t2}, \dots, v_{td}$. The power dissipated on the d edges $(u_t, v_{t1}), (u_t, v_{t2}), \dots, (u_t, v_{td})$ is exactly (recall from electric theory $C = 1/R$ and $P = C \cdot V^2$)

$$\sum_{i=1}^{d} \left(\overbrace{\mathrm{out}(S_{v_{ti}})}^{C:\text{ conductance of } (u_t, v_{ti})} \times \left(\overbrace{\left(\sum_{v \in S_{v_{ti}}} \frac{w_{l+1}(v)}{w_{l+1}(S_{v_{ti}})} \boldsymbol{x}_v \right) - \left(\sum_{u \in S_{u_t}} \frac{w_l(u)}{w_l(S_{u_t})} \boldsymbol{x}_u \right)}^{V:\text{ voltage difference across } (u_t, v_{ti})} \right)^2 \right).$$

We can verify that the two fractions are both probabilities. By Lemma 3.5 ($\delta = 1$), this is at most

$$\sum_{i=1}^{d} \left(\operatorname{out}(S_{v_{ti}}) \times \sum_{u \in S_{u_t}, v \in S_{v_{ti}}} \left(\frac{w_{l+1}(v)}{w_{l+1}(S_{v_{ti}})} \cdot \frac{w_l(u)}{w_l(S_{u_t})} \right) (\boldsymbol{x}_v - \boldsymbol{x}_u)^2 \right).$$

Since the $S_{v_{ti}}$'s form a partition of S_{u_t}, there are exactly $|S_{u_t}|^2$ terms when we expand the sum and we get

$$\sum_{u \in S_{u_t}, v \in S_{u_t}} \left(\frac{w_{l+1}(v)}{w_{l+1}(S_{v_{tv}})} \cdot \operatorname{out}(S_{v_{tv}}) \cdot \frac{w_l(u)}{w_l(S_{u_t})} \right) (\boldsymbol{x}_v - \boldsymbol{x}_u)^2$$

where $S_{v_{tv}}$ denotes the level l cluster $S_{v_{ti}}$ that contains v. Using (4.2), this simplifies to $\sum_{u,v \in S_{u_t}} (d_{u,v} \cdot (\boldsymbol{x}_v - \boldsymbol{x}_u)^2)$, which is exactly the power dissipated in K_{u_t} when the voltage settings are specified by (the corresponding coordinates of) $\boldsymbol{x}$. The final step is to observe that power dissipation is additive on conductance. Overlapping the K_{u_t}'s for each u_t in T, we conclude

$$\sigma(T/RC(T)) \leq 1. \tag{4.4}$$

4.5 Flow Shortening

We now specify and analyze the embedding of $RC(T)$ into G, both of which are on the same set of vertices. The vertex sets are embedded straightforwardly, and it remains to show how an edge (u, v) in $RC(T)$ can be embedded into G.

In Definition 4.4, the weight of (u, v) in $RC(T)$ is the sum of the demands between u and v, summing over the CMCFPs inside the clusters containing them. Naturally we embed (u, v) into the overlapping of the flow paths in the solutions to these CMCFPs. Unfortunately the CMCFP theorem of Aumann and Rabani [2] that Räcke used makes little guarantee on the length of the flow paths, which can be up to n. We seek to find a tighter bound. Luckily, a recent paper by Kolman and Scheideler [18] provides a clean solution.

First we need two definitions. In a *product multicommodity flow problem* (PMFP) on G, a nonnegative weight $\pi(u)$ is associated with each vertex u. There is a commodity for each ordered pair of nodes (u, v) with demand $\pi(u)\pi(v)$. Let I_0 be the PMFP on G in which $\pi(u) = c(u)/\sqrt{c(V)}$ for each u. The *flow number* of G, denoted $F(G)$, is defined as the minimum, taken over all feasible solutions S of I_0, of the maximum of the congestion and the dilation of S. The following theorem was proved in [18]:

Theorem 4.5 (part of Lemma 9, [18]) *Given a graph with flow number F. For any $\epsilon \in (0, 1]$ and for any feasible solution S to an instance of CMCFP with a throughput fraction of q, there exists a feasible solution with throughput fraction $q/(1 + \epsilon)$ that uses paths of length at most $2F(1 + 1/\epsilon)$.*

For our purpose, we will fix ϵ to be 1. Theorem 4.5 allows us to half the throughput fraction of any flow solution and obtain a bound on the path lengths that can be a lot tighter than $O(n)$. The flow numbers for several common graphs are as follows [18]: $F(\text{line}) = \Theta(n)$, $F(\text{mesh}) = \Theta(\sqrt{n})$, $F(\text{hypercube}) = \Theta(\log n)$, $F(\text{expander}) = \Theta(\log(n))$. In general, the following theorem holds:

Theorem 4.6 (Theorem 4, [18]) *Consider a graph $G = (V, E)$. Let $\alpha(G) = \min_{U \subset V, |U| \leq |V|/2} \frac{\operatorname{out}(U)}{|U|}$ be the expansion of G and let $\Delta(G) = \max_{v \in V} c(v)$ be the total incident weight of the heaviest vertex in G. The flow number of G satisfies $F(G) = \Omega(\alpha^{-1}(G))$ and also $F(G) = O(\alpha^{-1}(G)\Delta(G) \log n)$.*

4.6 Bounding $\sigma(RC(T)/G)$

To analyze the embedding of $RC(T)$ into G, we will consider the embedding of each K_{v_t} individually. Recall that K_{v_t} is a complete graph on S_{v_t} and the weight of an edge between two vertices is the total demand between them.

The CMCFP in S_{v_t} can have at most n^2 commodities. Aumann and Rabani [2] showed that such a CMCFP can be satisfied up to $q = \Omega(\phi/\log n)$, where ϕ is the value of the sparsest cut. Räcke [21, Lemma 4] showed that ϕ is $\Omega(1/\log n)$ in the CMCFP in every cluster. Hence $q = \Omega(1/\log^2 n)$. Apply the flow shortening lemma (Theorem 4.5) of Kolman and Scheideler only lowers this by a constant factor. In other words, if we insist to route all the demands of the CMCFP *in full*, the congestion of an edge in G is bounded by $O(\log^2 n)$. Now observe that the level l clusters form a partition of G and hence the flows in their CMCFPs can be routed simultaneously without affecting each other. By Theorem 4.3, we know there are only $O(\log n)$ levels. Therefore, the congestion of G when fully routing all the CMCFPs in every S_{v_t} simultaneously is $O(\log^3 n)$.

We now bound the length of the (already shortened) flow paths in the CMCFP in S_{v_t} using Theorem 4.6. First, note that $\Delta(G)$ is an upperbound on $\Delta(S_{v_t})$. Then, observe that even though $\alpha(S_{v_t})$ can be smaller than $\alpha(G)$, they are related by the bandwidth ratio λ_{v_t} as follows. By Theorem 4.3, λ_{v_t} is $O(\log n)$. Hence $\text{cap}(U, S_{v_t}\backslash U)$ is $\Omega(\text{out}(U)/\log n)$ for any $U \subset S_{v_t}$ up to half the size of S_{v_t}. Observe that $\text{cap}(U, S_{v_t}\backslash U)$ is in fact $\text{out}(U)$ *within* S_{v_t}, and so it follows that $\alpha(S_{v_t})$ is $\Omega(\alpha(G)/\log n)$. Therefore, we conclude the dilation of any edge in $RC(T)$ is $O(\alpha^{-1}(G)\Delta(G)\log^2 n)$ and with Lemma 2.4 we obtain

$$\sigma(RC(T)/G) \leq O(\alpha^{-1}(G)\Delta(G)\log^5 n). \tag{4.5}$$

4.7 Our Bound on $\kappa(G,T)$

Combining the bounds from the above subsections using Definition 2.3, Lemma 2.2, Lemma 2.4 and Corollary 2.9, our bound on $\kappa(G,T)$ is as follows.

Theorem 4.7 *Let T be the decomposition tree constructed by Räcke's algorithm [21] on G. Then*

$$\kappa(T,G) = O(\alpha^{-1}(G)\Delta(G)\log^6 n).$$

Finally, let's consider a $\sqrt{n} \times \sqrt{n}$ square mesh M with unit edge weights as an example application. It can be verified that $\alpha(M)$ is $O(1/\sqrt{n})$. Our bound will then be $\kappa(M, T_M) = O(\sqrt{n}\log^6 n)$.

Acknowledgements

We would like to thank Eduardo Laber, Harald Räcke and Andrea Richa for fruitful discussions. We also thank the anonymous referees who reviewed previous revisions of this paper for providing helpful feedback.

References

[1] N. Alon, R. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

[2] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

[3] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1996.

[4] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 2002. Submitted.

[5] M. Bienkowski, M. Korzeniowski, and H. Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, 2003.

[6] E. Boman, D. Chen, B. Hendrickson, and S. Toledo. Maximum-weight-basis preconditioners. *Numerical Linear Algebra and Applications*, 2002. Submitted.

[7] E. Boman and B. Hendrickson, 2002. Personal communication.

[8] E. Boman and B. Hendrickson. Support theory for preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 2002. Submitted.

[9] D. Chen. Analysis, implementation, and evaluation of Vaidya's preconditioners. Master's thesis, School of Mathematical Sciences, Tel-Aviv University, 2001.

[10] D. Chen and S. Toledo. Implementation and evaluation of Vaidya's preconditioners. In *Preconditioning 2001*, Tahoe City, CA, 2001.

[11] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*, volume 22 of *Carus Mathematical Monographs*. Mathematical Association of America, 1984.

[12] M. Elkin, Y. Emek, D. A. Spielman, and S. Teng. Lower-stretch spanning trees. In *STOC, to appear*, 2005.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3 edition, 1996.

[14] K. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, October 1996. CMU CS Tech Report CMU-CS-96-123.

[15] K. D. Gremban, G. L. Miller, and M. Zagha. Performance evaluation of a new parallel preconditioner. In *Proceedings of the Ninth International Parallel Processing Symposium*, pages 65–69, Santa Barbara, Apr. 1995.

[16] L. A. Hageman and D. M. Young. *Applied Iterative Methods*. Computer Science and Applied Methematics. Academic Press, Inc, San Diego and London, 1981.

[17] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, 2003.

[18] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. In *SODA*, pages 184–193, 2002.

[19] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, November 1999.

[20] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16:346–358, 1979.

[21] H. Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 43–52. IEEE, 2002.

[22] J. Reif. Efficient approximate solution of sparse linear systems. *Computers and Mathematics, with Applications*, 36, 1998.

[23] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1996.

[24] D. A. Spielman and S. Teng. Solving SSPDD linear systems in time $O(m^{1.31})$. In *Proceedings of the Forty-Forth Annual Symposium on Foundations of Computer Science*, 2003.

[25] D. A. Spielman and S. Teng. Nearly-linear time algorithms for graph paritioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

[26] P. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. A talk based on an unpublished manuscript, October 1991.

A Analyzing The Decomposition Tree by Bienkowski *et al.*

A.1 Description

Give a graph $G = (V, E)$, the decomposition tree T introduced by Bienkowski, Korzeniowski and Räcke [5] also corresponds to a laminar decomposition of G. The following is a description of the differences between it and the original proposal by Räcke [21] as described in Section 4.1.

The nodes of T consist of two types: red nodes and blue nodes. For the cluster corresponding to V, there is only a red node. For each other cluster $S \subsetneq V$ in the decomposition, there are two nodes v_t^b (blue) and v_t^r (red) in T. Note that both $S_{v_t^r}$ and $S_{v_t^b}$ refer to the same cluster. The superscript of a node will be used to denote its color when the color is important.

The edges of T can be defined as follows. (In our notation, the parent node of the tree edge (u_t, v_t) is always u_t.) First, form a decomposition tree using just the red nodes as in the original proposal. Then, for every edge (u_t^r, v_t^r), let v_t^b be the blue node associated with $S_{v_t^r}$. Replace the edge by two edges (u_t^r, v_t^b) and (v_t^b, v_t^r). Notice that v_t^r is the only child of v_t^b.

The *level* of a node is defined to be the number of red nodes on the root-to-node path, not counting the root itself. For example, each of the children (blue) of the root node and its only child (red) are both on level 1. The level of a cluster is defined to be the level of its associated red node.

Let v_t be a level l node and u_t be its parent. The weight of (u_t, v_t) is not explicitly specified by Bienkowski *et al.* since their proofs are based on analyzing the competitive ratio of their algorithm to the optimal offline algorithm. For our purpose (and it is also implicit in their proofs), we will set the weight to be $\text{out}(S_{v_t})$. The definition of the *weight function* $w_l(X) = \text{cap}(X, V) - \sum_{v_t \in V_t^l} \text{cap}(X \cap S_{v_t}, S_{v_t})$ for $X \subseteq V$ remains the same.

For a level $(l-1)$ cluster S_{u_t}, again a CMCFP is defined. There are $|S_{u_t}|^2$ commodities $f_{u,v}$ for $u, v \in S_{u_t}$. The source and sink of $f_{u,v}$ are u and v respectively, and the demand is

$$\frac{w_l(u) \cdot w_l(v)}{w_l(S_{u_t})}. \tag{1.6}$$

The flows are required to stay within S_{u_t} and must respect the edge weights as link capacities.

The connectivity characteristics defined by Bienkowski *et al.* for measuring the quality of a decomposition are somewhat different from the original ones. Let γ denote the maximum possible ratio between the throughput fraction of a CMCFP and the sparsity of an approximate sparsest cut on G. Note that γ is a function of three things: the CMCFP, the graph G, and the algorithm used to approximate the cut. For instance, using the algorithm by Aumann and Rabani [2] on a k-commodities CMCFP on a general graph, we have $\gamma = O(\log k)$. Bienkowski *et al.* defined two expressions based on γ.

Definition A.1 *Define $\lambda = 64\gamma \log n$ and $q_{\min} = 1/(24\gamma\lambda)$.*

Using γ, the following two characteristics are defined. Let S be a level l cluster. We say that S fulfills the *throughput property* if the solution to the CMCFP in S has a throughput fraction of at least $q_{\min}$. Notice that $q_{\min} = \Omega(1/\log^3 n)$ since all the CMCFPs in this section have at most $O(n^2)$ commodities. Also, we say that S fulfills the *precondition* if

$$\max_{|U| \le \frac{3}{4}|S|} \frac{w_l(U)}{\text{cap}(U, S \backslash U)} \le \lambda. \tag{1.7}$$

Among other things, Bienkowski *et al.* proved the following theorem.

Theorem A.2 (Lemma 4, [5]) *Let G be a graph with n vertices. There exists a polynomial time algorithm to compute decomposition tree T from G such that every cluster in the decomposition satisfies both the throughput property and the precondition. Moreover, the height of tree is $O(\log n)$.*

Here is the “table of content” for the rest of this section.

$$\kappa(G,T) \underset{\text{Def. 2.3}}{=\prod} \begin{cases} \sigma(G/T) \underset{\text{Cor. 2.9}}{\le \prod} \begin{cases}\text{congestion} \le 1 & \text{Section A.2}\\ \quad\text{dilation} = O(\log n) & \text{Section A.2}\end{cases} \\ \sigma(T/G) \underset{\text{Lem. 2.2}}{\le \prod} \begin{cases} \sigma(T/RC(T)) \le 1 \text{ by electrical argument} & \text{Section A.4}\\ \sigma(RC(T)/G) \underset{\text{Lem. 2.4}}{\le \prod} \begin{cases}\text{congestion} = O(\log^4 n) & \text{Section A.6}\\ \quad\text{dilation} = O(F(G)\log^2 n) & \text{Section A.6}\end{cases}\end{cases}\end{cases}$$

A.2 Bounding $\sigma(G/T)$

The bound for $\sigma(G/T)$ remains $O(\log n)$ and the proof is the same. The diameter of the tree is bounded by the height of the tree as in Theorem A.2.

A.3 The Räcke Complement

Our definition of the Räcke complement is (not surprisingly) robust against this newer construction. The demand is now specified by (1.6) instead of (4.2).

A.4 Bounding $\sigma(T/RC(T))$

Let $\boldsymbol{x}$ be an n-vector specifying the terminal voltages. The choice of the junction voltages is again directed by the oblivious routing scheme specified by Bienkowski *et al.* [5, Theorem 1]. Let v_t be a node at level l and u be a vertex in S_{v_t}. If v_t is blue, then it will be simulated by a vertex u with probability $w_l(u)/w_l(S_{v_t})$. If v_t is red, then the probability is $w_{l+1}(u)/w_{l+1}(S_{v_t})$. Therefore, we set the voltage of v_t to be

$$\sum_{u\in S_{v_t}} \frac{w_l(u)}{w_l(S_{v_t})}\,\boldsymbol{x}_u \text{ if } v_t \text{ is blue, or } \sum_{u\in S_{v_t}} \frac{w_{l+1}(u)}{w_{l+1}(S_{v_t})}\,\boldsymbol{x}_u \text{ if } v_t \text{ is red.} \tag{1.8}$$

Let u_t^r be a level $(l-1)$ red node with d blue children $v_{t1}^b, v_{t2}^b, \dots, v_{td}^b$. Notice that these blue children are at level l and they each have exactly one red child, also at level l. Further let the red child of v_{ti}^b be v_{ti}^r for $i = 1, 2, \dots, d$. We will bound the power dissipation of the (u_t^r, v_{ti}^b) edges (a star centered at u_t^r) and (v_{ti}^b, v_{ti}^r) edges (d parallel edges) as two groups separately.

Group 1: (u_t^r, v_{ti}^b) **edges** The total power dissipation on these edges is

$$\sum_{i=1}^{d}\left(\overbrace{\text{out}(S_{v_{ti}^b})}^{C:\text{ conductance of }(u_t^r, v_{ti}^b)} \times \left(\overbrace{\left(\sum_{v\in S_{v_{ti}^b}} \frac{w_l(v)}{w_l(S_{v_{ti}^b})}\,\boldsymbol{x}_v\right) - \left(\sum_{u\in S_{u_t^r}} \frac{w_l(u)}{w_l(S_{u_t^r})}\,\boldsymbol{x}_u\right)}^{V:\text{ voltage difference across }(u_t^r, v_{ti}^b)}\right)^2\right).$$

We can verify that the two fractions are both probabilities. By Lemma 3.5 ($\delta = 1$), this is at most

$$\sum_{i=1}^{d}\left(\text{out}(S_{v_{ti}^b}) \times \sum_{u\in S_{u_t^r}, v\in S_{v_{ti}^b}} \left(\frac{w_l(v)}{w_l(S_{v_{ti}^b})}\cdot\frac{w_l(u)}{w_l(S_{u_t^r})}\right)(\boldsymbol{x}_v - \boldsymbol{x}_u)^2\right).$$

Since the $S_{v_{ti}^b}$'s form a partition of $S_{u_t^r}$, there are exactly $|S_{u_t^r}|^2$ terms when we expand the sum and we get

$$\sum_{u\in S_{u_t^r}, v\in S_{u_t^r}} \left(\frac{w_l(v)}{w_l(S_{v_{tv}^b})}\cdot \text{out}(S_{v_{tv}^b})\cdot\frac{w_l(u)}{w_l(S_{u_t^r})}\right)(\boldsymbol{x}_v - \boldsymbol{x}_u)^2$$

where $S_{v_{tv}^b}$ denotes the level l cluster $S_{v_{ti}^b}$ that contains v. Observe that $w_l(S_{v_{tv}^b})$ is actually $\text{out}(S_{v_{tv}^b})$ and so they cancel each other. Using (1.6), this simplifies to $\sum_{u,v\in S_{u_t^r}}(d_{u,v}\cdot(\boldsymbol{x}_v - \boldsymbol{x}_u)^2)$, which is exactly the power dissipated in $K_{u_t^r}$ when the voltage settings are specified by (the corresponding coordinates of) $\boldsymbol{x}$.

Group 2: (v_{ti}^b, v_{ti}^r) edges The total power dissipation on these edges is

$$\sum_{i=1}^{d}\left(\overbrace{\text{out}(S_{v_{ti}^r})}^{C\text{: conductance of }(v_{ti}^b, v_{ti}^r)} \times \left(\overbrace{\left(\sum_{v\in S_{v_{ti}^r}} \frac{w_{l+1}(v)}{w_{l+1}(S_{v_{ti}^r})}\boldsymbol{x}_v\right) - \left(\sum_{u\in S_{v_{ti}^b}} \frac{w_l(u)}{w_l(S_{v_{ti}^b})}\boldsymbol{x}_u\right)}^{V\text{: voltage difference across }(v_{ti}^b, v_{ti}^r)}\right)^2\right).$$

We can verify that the two fractions are both probabilities. By Lemma 3.5 ($\delta = 1$), this is at most

$$\sum_{i=1}^{d}\left(\text{out}(S_{v_{ti}^r}) \times \sum_{u\in S_{v_{ti}^b}, v\in S_{v_{ti}^r}} \left(\frac{w_{l+1}(v)}{w_{l+1}(S_{v_{ti}^r})}\cdot\frac{w_l(u)}{w_l(S_{v_{ti}^b})}\right)(\boldsymbol{x}_v - \boldsymbol{x}_u)^2\right).$$

Observe that $S_{v_{ti}^r}$ and $S_{v_{ti}^b}$ are in fact the same cluster. Hence $w_l(S_{v_{ti}^b})$ equals to $\text{out}(S_{v_{ti}^r})$ and they cancel each other. Furthermore, we can check that $w_l(u) \le w_{l+1}(u)$. Combining these observations, this is at most

$$\sum_{i=1}^{d}\left(\sum_{u,v\in S_{v_{ti}^b}}\left(\frac{w_{l+1}(v)\cdot w_{l+1}(u)}{w_{l+1}(S_{v_{ti}^r})}\right)(\boldsymbol{x}_v - \boldsymbol{x}_u)^2\right).$$

Consider each of these d sums individually. Using (1.6), each sum corresponds to $\sum_{u,v\in S_{v_{ti}^b}}(d_{u,v}\cdot(\boldsymbol{x}_v - \boldsymbol{x}_u)^2)$, which is exactly the power dissipated in $K_{v_{ti}^b}$ when the voltage settings are specified by (the corresponding coordinates of) $\boldsymbol{x}$.

Combining the results from the two groups, we conclude

$$\sigma(T/RC(T)) \le 1. \tag{1.9}$$

A.5 Flow Shortening

In the past, we have struggled to put a bound on the length of the flow paths since the construction in Räcke's original paper [21] uses asymmetric product flows (a PMFP where $\pi(u)$ can be different depending on whether u is a source or a sink). The situation is rectified by Bienkowski *et al.* [5] and also Harrelson *et al.* [17]. Both groups of authors proposed to use symmetric product flows, making the length of the flow paths analyzable by the classic result of Leighton and Rao [19]. However, since the wonderful flow shortening lemma of Kolman and Scheideler [18] applies to all CMCFPs, we can continue to use it here.

A.6 Bounding $\sigma(RC(T)/G)$

Again we consider the embedding of each K_{v_t} into G individually.

By Theorem A.2, the cluster S_{v_t} satisfies the throughput property, which states that states that the CMCFP set up inside it has a throughput fraction of at least $q_{\min} = \Omega(1/\log^3 n)$. Since all the clusters corresponding to red nodes on level l form a partition of V, their flows can be routed simultaneously without affecting each other. Similarly, this also holds for the flows in the blue clusters. Again, applying the flow shortening lemma only lower the fraction by a constant. By Theorem A.2, there are $O(\log n)$ levels. Therefore, the congestion on any edge in G when fully routing the demands in all the CMCFPs simultaneously is $O(\log^4 n)$.

Now the bandwidth property is not defined by Bienkowski *et al.*. In its replacement, we have the precondition which is a slightly weaker connectivity characteristic. Let S_{v_t} be on level l. The precondition states that for each subset U up to 3/4 of the size S_{v_t}, λ is an upperbound on the ratio $\frac{w_l(U)}{\text{cap}(U, S_{v_t}\setminus U)}$. We can manipulate this ratio to allow us to relate $\alpha(S_{v_t})$ to $\alpha(G)$ as follows. Consider the following inequality:

$$1+\lambda \geq 1+\frac{w_l(U)}{\mathrm{cap}(U, S_{v_t}\backslash U)} = \frac{\mathrm{cap}(U, S_{v_t}\backslash U)+w_l(U)}{\mathrm{cap}(U, S_{v_t}\backslash U)} = \frac{\mathrm{out}(U)}{\mathrm{cap}(U, S_{v_t}\backslash U)}$$

Again, observe that $\mathrm{cap}(U, S_{v_t}\backslash U)$ is in fact $\mathrm{out}(U)$ *within* S_{v_t}. By applying the definition of λ in Definition A.1, we have $\alpha(S_{v_t}) = \Omega((1+\lambda)\alpha(G)) = \Omega(\alpha(G)/\log^2 n)$. Therefore, we conclude the dilation of any edge in $RC(T)$ is $O(\alpha^{-1}(G)\Delta(G)\log^3 n)$ and with Lemma 2.4 we obtain

$$\sigma(RC(T)/G) \leq O(\alpha^{-1}(G)\Delta(G)\log^7 n). \tag{1.10}$$

A.7 Our Bound on $\kappa(G,T)$

Combining the bounds from the above subsections using Definition 2.3, Lemma 2.2, Lemma 2.4 and Corollary 2.9, our bound on $\kappa(G,T)$ is as follows.

Theorem A.3 *Let T be the decomposition tree constructed by the algorithm of Bienkowski* et al. *[5] on G. Then*

$$\kappa(T,G) = O(\alpha^{-1}(G)\Delta(G)\log^8 n).$$

Finally, again let's consider a $\sqrt{n}\times\sqrt{n}$ square mesh M with unit edge weights as an example application. Compared to our bound using Räcke's original decomposition tree [21], our new bound degrades by two log factors to $\kappa(M,T_M) = O(\sqrt{n}\log^8 n)$. However, the advantage of using the newer decomposition tree of Bienkowski *et al.* [5] is that it is known to be computable in polynomial time (Theorem A.2), whereas the former one by Räcke is not known to be.

B Proofs for Laplacians as Circuits

This appendix contains the proofs omitted in Section 3.

Lemma B.1 *When $u \neq v$, $A_{u,v}$ equals to the negated weight of the edge (u,v). Otherwise, $A_{v,v}$ is the sum of the weights of the edges incident on the vertex v. Therefore, A is diagonally dominant.*

Lemma B.2 *The signs on the two entries defining an edge in an edge-vertex incidence matrix Γ can be arbitrary, as long as they differ, and yet this still results in the same Laplacian matrix $A = \Gamma^{\mathsf{T}}W\Gamma$.*

Proof of Lemma 3.1:

By Lemma B.1, we have

$$(A\boldsymbol{y})_i = -\sum_{j=1}^{i-1} c_j\boldsymbol{y}_j + \sum_{\substack{j=1\\ j\neq i}}^{n} c_j\boldsymbol{y}_i - \sum_{j=i+1}^{n} c_j\boldsymbol{y}_j = \sum_{\substack{j=1\\ j\neq i}}^{n} c_j(\boldsymbol{y}_i - \boldsymbol{y}_j).$$

This is precisely the net current flow into the i-th node of G. (The net current flow into a node is the sum, over all incident wires, of the product between the conductance the wire and the potential difference across the wire.) ■

Proof of Theorem 3.2:

By Lemma B.2, *w.l.o.g.* let every wire in G be oriented in such a way that each wire starts at the vertex with the larger index and ends at the vertex with the smaller index. (So the vertex with the smaller index will correspond to a "+1" in Γ.)

Consider a wire $e_i = (v_{ia}, v_{ib}) \in E$. Observe that $\Gamma\boldsymbol{y} = \sum_{j=1}^{n}\boldsymbol{y}_j\Gamma_{(j)}$ and thus

$$(\Gamma\boldsymbol{y})_i = \sum_{j=1}^{n}\boldsymbol{y}_j\Gamma_{i,j}.$$

By our orientation assumption and the fact that only two entries are nonzero in each row of Γ, we can simplify this to

$$(\Gamma \boldsymbol{y})_i = \boldsymbol{y}_{ia} - \boldsymbol{y}_{ib}$$

for some indices ia and ib. Using the following identity that holds for arbitrary $m \times m$ matrix M and m-vector $\boldsymbol{x}$,

$$\boldsymbol{x}^\mathsf{T} M \boldsymbol{x} = \sum_{i=1}^{m} \boldsymbol{x}_i \Big(\sum_{j=1}^{m} \boldsymbol{x}_j M_{i,j} \Big) = \sum_{(i,j)\in\{m \otimes m\}} \boldsymbol{x}_i \boldsymbol{x}_j M_{i,j},$$

we have

$$\boldsymbol{y}^\mathsf{T} A \boldsymbol{y} = (\Gamma \boldsymbol{y})^\mathsf{T} W (\Gamma \boldsymbol{y}) = \sum_{(i,j)\in\{|E| \otimes |E|\}} (\boldsymbol{y}_{ia} - \boldsymbol{y}_{ib})(\boldsymbol{y}_{jc} - \boldsymbol{y}_{jd}) W_{i,j}.$$

Since $W_{i,j} = 0$ for $i \neq j$, this yields

$$\sum_{i=1}^{|E|} (\boldsymbol{y}_{ia} - \boldsymbol{y}_{ib})^2 W_{i,i},$$

which is precisely summing the power dissipated over all wires in G. (The power dissipated by a wire is the square of the voltage difference between the two end-points multiplied by the conductance of the wire.) ■

Proof of Lemma 3.5:

$$\begin{aligned}
&\text{RHS} - \text{LHS} \\
&= \sum_{(i,j)\in\{m\otimes n\}} \alpha_i \beta_j \boldsymbol{v}_i^2 - \sum_{(i,j)\in\{m\otimes n\}} 2\alpha_i \beta_j \boldsymbol{v}_i \boldsymbol{w}_j + \sum_{(i,j)\in\{m\otimes n\}} \alpha_i \beta_j \boldsymbol{w}_j^2 \\
&\quad - \Big(\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i\Big)^2 + 2\Big(\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i\Big)\Big(\sum_{j=1}^{n} \beta_j \boldsymbol{w}_j\Big) - \Big(\sum_{j=1}^{n} \beta_j \boldsymbol{w}_j\Big)^2
\end{aligned}$$

(cancelling the second term with the fifth term)

$$\begin{aligned}
&= \sum_{(i,j)\in\{m\otimes n\}} \alpha_i \beta_j \boldsymbol{v}_i^2 - \Big(\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i\Big)^2 + \sum_{(i,j)\in\{m\otimes n\}} \alpha_i \beta_j \boldsymbol{w}_j^2 - \Big(\sum_{j=1}^{n} \beta_j \boldsymbol{w}_j\Big)^2 \\
&= \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i^2 \Big(\sum_{j=1}^{n} \beta_j\Big) - \Big(\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i\Big)^2 + \sum_{j=1}^{n} \beta_j \boldsymbol{w}_j^2 \Big(\sum_{i=1}^{m} \alpha_i\Big) - \Big(\sum_{j=1}^{n} \beta_j \boldsymbol{w}_j\Big)^2
\end{aligned}$$

(α's and β's both sum to δ)

$$\begin{aligned}
&= \delta \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i^2 - \Big(\sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i\Big)^2 + \delta \sum_{j=1}^{n} \beta_j \boldsymbol{w}_j^2 - \Big(\sum_{j=1}^{n} \beta_j \boldsymbol{w}_j\Big)^2 \\
&= \delta \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i^2 - \Big(\sum_{i=1}^{m} \alpha_i^2 \boldsymbol{v}_i^2 + \sum_{\substack{(i,j)\in\{m\otimes m\} \\ i\neq j}} \alpha_i \alpha_j \boldsymbol{v}_i \boldsymbol{v}_j\Big) + \delta \sum_{j=1}^{n} \beta_j \boldsymbol{w}_j^2 - \Big(\sum_{j=1}^{n} \beta_j^2 \boldsymbol{w}_j^2 + \sum_{\substack{(i,j)\in\{n\otimes n\} \\ i\neq j}} \beta_i \beta_j \boldsymbol{w}_i \boldsymbol{w}_j\Big) \\
&= \sum_{i=1}^{m} \alpha_i (\delta - \alpha_i) \boldsymbol{v}_i^2 - \sum_{\substack{(i,j)\in\{m\otimes m\} \\ i\neq j}} \alpha_i \alpha_j \boldsymbol{v}_i \boldsymbol{v}_j + \sum_{j=1}^{n} \beta_j (\delta - \beta_j) \boldsymbol{w}_j^2 - \sum_{\substack{(i,j)\in\{n\otimes n\} \\ i\neq j}} \beta_i \beta_j \boldsymbol{w}_i \boldsymbol{w}_j
\end{aligned}$$

(α's and β's both sum to δ)

$$= \sum_{\substack{(i,j)\in\{m\otimes m\}\\ i\neq j}} \alpha_i\alpha_j(\boldsymbol{v}_i^2 - \boldsymbol{v}_i\boldsymbol{v}_j) + \sum_{\substack{(i,j)\in\{n\otimes n\}\\ i\neq j}} \beta_i\beta_j(\boldsymbol{w}_i^2 - \boldsymbol{w}_i\boldsymbol{w}_j)$$

(grouping (i,j) and (j,i) terms together)

$$= \sum_{\substack{(i,j)\in\{m\otimes m\}\\ i<j}} \alpha_i\alpha_j(\boldsymbol{v}_i - \boldsymbol{v}_j)^2 + \sum_{\substack{(i,j)\in\{n\otimes n\}\\ i<j}} \beta_i\beta_j(\boldsymbol{w}_i - \boldsymbol{w}_j)^2$$

(α_i's and β_j's are nonnegative)

$$\geq 0$$

■

C Extensions

In this section we present an extension of good preconditiong techniques for Laplacians to solving any real symmetric diagonally-dominant system with a nonnegative diagonal. First we consider matrices M that can be written as $L + D$, where L is a Laplacian and D is a nonnegative diagonal matrix. Then we present a technique presented in Gremban [14] for handling matrices with positive off-diagonal elements. Composing the two gives us the class of all symmetric diagonally-dominant real matrices.

C.1 Strict diagonal-dominance

Suppose we are given a matrix $A = L + D$, where L is a Laplacian and D is a nonnegative diagonal matrix, for which we seek to construct a preconditioner. A simple approach, which we consider folklore, is to construct a good preconditioner, P for L and then use $P + D$ as a preconditioner for A. We must slightly modify this approach since our preconditioners are Support Tree Preconditioners.

We may construct a Support Tree Preconditioner, $B = \begin{pmatrix} T & U \\ U^\mathsf{T} & W \end{pmatrix}$ for L and to use $B' = \begin{pmatrix} T & U \\ U^\mathsf{T} & W + D \end{pmatrix}$ as a preconditioner for A. If we let $Q = W - U^\mathsf{T}T^{-1}U$, by Lemma 2.5 it suffices to bound $\sigma(A/Q + D)$ and $\sigma(Q + D/A)$.

Proposition C.1 *If X, Y, and Z are spsd matrices of the same size then $\sigma(X+Z/Y+Z) \leq \max\{\sigma(X/Y), 1\}$.*

Proof We have $\sigma(X + Z/Y + Z) = \min\{\tau \mid \forall\boldsymbol{x},\ \tau\cdot\boldsymbol{x}^\mathsf{T}(Y + Z)\boldsymbol{x} \geq \boldsymbol{x}^\mathsf{T}(X + Z)\boldsymbol{x}\} = \min\{\tau \mid \forall\boldsymbol{x},\ (\tau - 1)\cdot\boldsymbol{x}^\mathsf{T}Z\boldsymbol{x} + \tau\cdot\boldsymbol{x}^\mathsf{T}Y\boldsymbol{x} \geq \boldsymbol{x}^\mathsf{T}X\boldsymbol{x}\} \leq \max\{1, \sigma(X/Y)\}$. ■

Corollary C.2 *If $\sigma(L, Q)$, $\sigma(Q, L) \geq 1$ then $\kappa(A, Q + D) = \kappa(L + D, Q + D) \leq \kappa(L, Q)$.*

Thus our bounds for Laplacians also hold for symmetric diagonally-dominant matrics with nonpositive off-diagonals.

C.2 Positive off-diagonals

In this section we present a technique of Gremban for solving (symmetric) systems with positive off-diagonals by invoking any method for solving (symmetric) systems with nonpositive off-diagonals on an expanded system.

Suppose we seek to solve $A\boldsymbol{x} = \boldsymbol{b}$. If A contains positive off-diagonal elements, we can decompse it as $N + P$, where P contains precisely the positive off-diagonal elements of A, and N contains the diagonal and negative off-diagonal elements of A. Note that the matrix $A' = \begin{pmatrix} N & -P \\ -P & N \end{pmatrix}$ contains only nonpositive off-diagonals while preserving any symmetry in A.

We may instead simply solve the system $A' \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ -\boldsymbol{b} \end{pmatrix}$, since

$$A\left(\frac{\boldsymbol{u}-\boldsymbol{v}}{2}\right) = \frac{N\boldsymbol{u}-P\boldsymbol{v}}{2} - \frac{N\boldsymbol{v}-P\boldsymbol{u}}{2} = b.$$

As stated this is simply a preprocessing trick; however, one can convert a preconditioner, B', for A' into one for A with no worse a generalized condition bound. If B' satisfies some additional symmetry constraints, then one can also solve systems over B in linear time, which would allow one to directly apply STCG to A using B as a preconditioner.