

Solving Symmetric Diagonally-Dominant Systems by Preconditioning

Bruce M. Maggs Gary L. Miller Ojas Parekh R. Ravi Shan Leung Maverick Woo
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213
{bmm,glmiller,odp,rravi,maverick}@cs.cmu.edu

Abstract

In this paper we design support-tree preconditioners for $n \times n$ matrices with m nonzeros that are symmetric and diagonally-dominant with a nonnegative diagonal (SDD matrix). This reduces to designing such a preconditioner for a Laplacian matrix, A , which can be interpreted as an undirected nonnegatively-weighted graph, G with n vertices and m edges. Preconditioners accelerate the convergence of iterative methods for solving linear systems, and our preconditioner allows us to analyze the convergence of a particular algorithm, due to Gremban and Miller, called support-tree conjugate gradient (STCG). An advantage of support-tree preconditioners is that STCG parallelizes well. We show that STCG equipped with our preconditioner requires $O(m \log^2 n \cdot \sqrt{\text{dil}_{exp}(G)})$ work and $O(m)$ space to solve the system $A\mathbf{x} = \mathbf{b}$, where $\text{dil}_{exp}(G)$ is an edge-expansion-based upper bound on the diameter of G .

Existing bounds depend only on the size of the matrix (graph), hence our bound is incomparable. For instance, if G is a bounded-degree expander graph with uniform edge weights, $\text{dil}_{exp}(G) = O(\log^2 n)$, and the work is $O(n \log^3 n)$. This is currently the best known bound for Laplacians of expander graphs. We show that $\text{dil}_{exp}(G)$ is always at most n , hence our bound is at most $O(m\sqrt{n} \log^2 n)$ for any Laplacian (or SDD) matrix. For sufficiently dense systems, when $m = \Omega(n^{1.61})$, this bound offers the best known work guarantee of any linear-space method.

The main technical contributions of this paper include (i) adapting a recent result of Räcke to designing support-tree preconditioners, (ii) extending a power dissipation approach for bounding support numbers of preconditioners, and (iii) applying the methods used in Leighton and Rao’s approximate max-flow min-cut theorem to the “asymmetric” product flows that arise in Räcke’s construction.

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 $A\mathbf{x} = \mathbf{b}$, where A is a known $n \times n$ matrix, \mathbf{b} is a known $n \times 1$ vector in the column space of A , and \mathbf{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 \mathbf{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 \mathbf{b} . Hence, a distinction can be made between the time spent preprocessing the matrix A , and the time solving linear systems involving A . Thus our objective is to minimize the time to solve such a system assuming time polynomial in the size of A is available for preprocessing.

The matrices that arise in describing physical systems are generally not arbitrary but often display structural properties (mirroring structures in physical systems) that can be exploited. The classic textbook by Golub and Van Loan [12] 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 [18]. Nested dissection works well for graphs with small separators,

such as trees and planar graphs. In fact, Lipton *et al.* [18] 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 highly connected graphs like expanders.

An important sub-class of symmetric systems are symmetric diagonally-dominant (SDD) matrices; 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, 21]). 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 [13] shows that an SDD system can be solved by solving a Laplacian system twice as large. A Laplacian matrix, A , can be interpreted as a weighted adjacency matrix of an undirected graph, G with nonnegative edge weights (see Section 2.2). As we shall see, the time to solve such a system using a preconditioned iterative method can be as low as linear times a poly-log factor, which is the case for approximately uniformly weighted expander graphs.

Our approach will be 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. 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, we have comparable results for SDD systems. Strictly speaking, the bounds to follow apply to finding a solution \mathbf{x} such that $\|A\mathbf{x} - \mathbf{b}\| \leq \epsilon\|\mathbf{b}\|$, at the cost of an additional multiplicative factor of $O(\log(1/\epsilon))$, which we have omitted for simplicity.

Vaidya's seminal work [23, 9, 10] describes preconditioners resulting from relatively sparse spanning subgraphs. 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 analyzed a recursive variant proposed by Vaidya and introduced a construction which lifts the bounded-degree requirement in Vaidya's preconditioners [20]. Boman *et al.* [6] analyzed an extension proposed by Vaidya [23] of his preconditioners to all SDD systems. More recently, Boman and Hendrickson [7] demonstrated that the spanning trees designed by Alon *et al.* [1] in a different context result in preconditioners with a work bound of $O(m^{1.5}polylog(n))$ for any weighted graph. Spielman and Teng [22] improved upon this work by showing that adding edges to the Alon *et al.* trees results in a method that requires $O(m^{1.31})$ time.

Gremban, Miller, and Zaghera [14] and Gremban and Miller [13] 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. Furthermore, in order to approximate the topology of the graph, their trees are hierarchically constructed by recursively partitioning the graph.

1.2 Our results

In this paper, we adopt the support tree approach and show that the hierarchical decomposition trees of weighted undirected graphs, recently introduced by Räcke [19], give rise to good support tree preconditioners for the Laplacians of these graphs. In particular, we demonstrate that the Laplacian of any weighted graph, G with n nodes and m edges, has a good support tree. In this way our methods can be seen as extension of the tools developed by Gremban and Miller to the case of arbitrary Laplacians rather than just meshes.

Our bound relies upon an edge-expansion-based upper bound on the diameter of G , which we call

$dil_{exp}(G)$ and describe in more detail in Section 4. Our main result is that any SDD system can be solved with total work $O(m \log^2 n \cdot \sqrt{dil_{exp}(G)})$. We give examples of graphs in Section 4 which have good dil_{exp} bounds. Since we show $dil_{exp}(G)$ is always at most n , our bound is never worse than $O(m\sqrt{n} \log^2 n)$. For sufficiently dense systems, when $m = \Omega(n^{1.61})$, this bound provides the best known work guarantee of any linear-space method for solving arbitrary SDD systems. Another advantage of our preconditioners over those of Vaidya or Spielman and Teng is that they are Laplacians of log-depth support trees, which allows for parallelization in conjunction with the support tree conjugate gradient method.

Vaidya’s and the Alon *et al.* based preconditioners can be constructed faster than Racke’s trees; however, a preconditioner need only be constructed once to solve a system, $A\mathbf{x} = \mathbf{b}$ for different values of \mathbf{b} , and, as noted in the introduction, this is a problem of practical interest. Although in his initial paper Racke noted that no polynomial-time algorithm was known for his constructing his trees, in a recent paper with Bienkowski and Korzeniowski [5] a polynomial-time algorithm has been proposed for building Racke’s trees. A similar result was also achieved independently by Harrelson *et al.* [16]. Both results assume the maximum edge weight is polynomially bounded.

2 Background

The purpose of this section is to 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

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

Example: The RF Method The basis of iterative methods is the RF method [15], which illustrates the idea behind such approaches very well. For the sake of exposition, we assume we are dealing with nonsingular matrices¹; however, this is not a requirement imposed by iterative methods.

The update step for the RF method is simply $\mathbf{x}^{(i+1)} := \mathbf{x}^{(i)} - (A\mathbf{x}^{(i)} - \mathbf{b})$. The intuition behind this step is that $A\mathbf{x}^{(i)} - \mathbf{b}$ 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 vector $A\mathbf{x}^{(i)} - \mathbf{b}$ lies in the range of A , while the iterates $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(i+1)}$ are sought to lie 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 $\mathbf{x}^{(i+1)} := \mathbf{x}^{(i)} - B^{-1}(A\mathbf{x}^{(i)} - \mathbf{b})$, where B^{-1} is a *preconditioner* for the system $A\mathbf{x} = \mathbf{b}$.

We may also interpret preconditioning as solving the system $B^{-1}A\mathbf{x} = B^{-1}\mathbf{b}$ rather than $A\mathbf{x} = \mathbf{b}$. 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 . Thus we must strike a balance, and we discuss this issue next.

In choosing a preconditioner B^{-1} for the system $A\mathbf{x} = \mathbf{b}$, 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 $\mathbf{x}^{(i)} - B^{-1}(A\mathbf{x}^{(i)} - \mathbf{b})$. As we may precompute $B^{-1}\mathbf{b}$, the computation time is dominated by the time required for the sparse matrix-vector multiplication $\mathbf{c} := A\mathbf{x}^{(i)}$ and determining $\mathbf{z} = B^{-1}\mathbf{c}$, which we may re-state as finding a solution \mathbf{z} of the system $B\mathbf{z} = \mathbf{c}$. In fact this is all we require of B , hence B need

¹In 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 [13] for the details.

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 $B\mathbf{z} = \mathbf{c}$ in $O(n)$ time, then each iteration takes a total of $O(n + m)$ time. Solving a system over B is also a large obstacle in parallelizing iterative methods; support tree preconditioners offer a natural approach to parallelizing this step.

On the other hand, bounds on the total iteration count depend on which particular iterative method is used. We postpone a more detailed discussion of bounding the iteration count, except to remark that the bounds 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.*, \mathbf{v} .

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 arbitrary), and all other entries are set to 0. When G is weighted, we introduce a $|E| \times |E|$ diagonal matrix W , where $W_{e,e} \geq 0$ specifies the weight of the edge e . The *Laplacian* of G is defined to be $\Gamma^T W \Gamma$.

Let A be $\Gamma^T W \Gamma$. Clearly, A is a square matrix of size $|V| \times |V|$. Since A can be decomposed into the form $\Gamma^T W \Gamma$, A is symmetric positive semidefinite. It is not hard to verify that $A_{ij} = A_{ji} = -W_{ij,ij}$ for $i \neq j$, and the row (column) sums of A are 0; thus A is closely related to the weighted adjacency matrix of G .

We overload notation and also use G to refer to the Laplacian of a graph G , relying upon context for differentiation. Finally, we use $\{m \otimes n\}$ as a shorthand for $\{(i, j) \mid i \in \{1, 2, \dots, m\}, j \in \{1, 2, \dots, n\}\}$.

2.3 Support Theory

In this section, we present the definitions and concepts necessary to link the task of bounding the number of iterations taken by an iterative method to that of the (combinatorial) design of a preconditioner. A more detailed and general account of the material here can be found in Boman and Hendrickson [8] or Gremban [13].

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

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

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 of Laplacians of graphs.

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 [11]. Support numbers enjoy many nice properties which are proven and catalogued in Boman and Hendrickson [8]. Transitivity is but one:

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

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 $\kappa(A, B)$ [15]; however, given our restrictions on A and B , this is an equivalent one. It is well known that Preconditioned Conjugate Gradient, or PCG, on a system $A\mathbf{x} = \mathbf{b}$ with the preconditioner B (B^{-1}) requires at most $\sqrt{\kappa(A, B)} \log(1/\epsilon)$ iterations [15] to find a solution \mathbf{x} such that $\|A\mathbf{x} - \mathbf{b}\| \leq \epsilon \|\mathbf{b}\|$.

We conclude this section by presenting the main combinatorial tool upon which both Vaidya and Gremban and Miller 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 $\text{cong}(h) := \sum_{\{g \in E(G) \mid h \in \mathcal{P}(g)\}} w_G(g)/w_H(h)$ the *congestion* of h .

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

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

The Congestion-Dilation Lemma is stated in Vaidya [23] and Gremban and Miller [13] and proven in Bern *et al.* [4].

2.4 Support Tree Conjugate Gradient

A support tree preconditioner is defined over a potentially larger graph than the one for which it is created, hence in this section we present the tools necessary to compare matrices of different sizes.

Let A and $B = \begin{pmatrix} T & U \\ U^\top & W \end{pmatrix}$ 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, 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 the preconditioned conjugate gradient (PCG) method, 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 [13]. Gremban demonstrates the following.

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

We call Q the schur complement of B (with respect to W); Q represents the connectivity among the vertices of W in B at a cost of greater 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 $Q\mathbf{x} = \mathbf{b}$ for \mathbf{x} during each iteration; however, Q may be much more dense than B . STCG finds a solution \mathbf{x} by solving $B \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix}$ and simply discarding \mathbf{y} .

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

When B is a support tree the system can be solved quite efficiently. 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 $\bar{A} = \begin{pmatrix} 0 & 0 \\ 0 & A \end{pmatrix}$ 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 $\sigma(A/B) = \sigma(A/Q)$.*

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 $\sigma(A/B) \leq (\max_{h \in E(B)} \text{cong}_{\mathcal{P}}(h)) \cdot (\max_{g \in E(A)} \text{dil}_{\mathcal{P}}(g))$.*

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*

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

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

In this section we 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 useful in our analysis. For a more thorough account of this interpretation, one ought to consult Doyle and Snell [11] or Gremban [13], where one can find some of the results to follow. We defer longer proofs to the appendices.

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 \mathbf{y} is the n -vector specifying the voltage at each node of G , then $A\mathbf{y}$ 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 \mathbf{y} is a n -vector specifying the voltage on each node, then $\mathbf{y}^T A \mathbf{y}$ is the total power dissipated by G .*

3.1 Power and Support

Given the interpretation 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 $\sigma(A/B) = \min\{\tau \mid \forall \mathbf{x}, \tau \cdot \mathbf{x}^\top B \mathbf{x} \geq \mathbf{x}^\top A \mathbf{x}\}$, where by the previous section we may interpret \mathbf{x} as voltage settings and $\mathbf{x}^\top M \mathbf{x}$ as the power dissipated by a circuit M under the settings \mathbf{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 $B = \begin{pmatrix} T & U \\ U^\top & W \end{pmatrix}$ 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 , Kirchoff's Laws dictate a set of naturally occurring voltages for the junctions such that they have a net current flow of 0. Definition 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 Kirchoff's Law, then $\sigma(B/A) \leq \tau$.*

Proof For any terminal voltage settings, \mathbf{x} , $\mathbf{y} = -T^{-1}U\mathbf{x}$ is precisely the setting of the junctions determined by Kirchoff's Laws so that $T\mathbf{x} + U\mathbf{y} = \mathbf{0}$, hence the result 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 Racke'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 \mathbf{v} and \mathbf{w} be any m -vector and n -vector respectively. When applying voltages \mathbf{v} and \mathbf{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 $\sum_{i=1}^m \alpha_i \mathbf{v}_i$ and $\sum_{j=1}^n \beta_j \mathbf{w}_j$. Mathematically,*

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

4 Analyzing Racke's Decomposition Tree

As stated in Section 1, our focus here is on analyzing a decomposition tree recently introduced by Racke [19] and proving that it works well as a preconditioner for graph Laplacians. To facilitate our analysis, we first review Racke's construction and summarize the relevant notations.

4.1 Racke's Decomposition Tree

In a recent paper, Racke showed that it is possible to obviously route online requests on a general n -node network while achieving a low $O(\log^3 n)$ competitive ratio in the congestion.

Given a network $G = (V, E)$, 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 $S_{v_t} \subseteq V$. (For visual clarity, we deviate from Racke's usage and use T instead of T_G .) In particular, each of the leaves in T corresponds to a node in G , the root of T corresponds to the whole vertex set V and the tree node v_t is a child of u_t iff the cluster S_{v_t} is contained within S_{u_t} . (As in Racke's terminology, all variables subscripted with t refer to only objects in the decomposition tree.) By first building T using G and then computing a randomized embedding of T back into G , Racke's routing algorithm simply routes the requests on T instead. The randomized embedding will then specify the intermediate locations of the actual path to be taken on G , whereas the path between intermediate locations will in turn be specified by solving 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 of the CMCFPs in turn depends on T only and hence can be solved in a preprocessing stage. In his

first paper [19], the decomposition algorithm specified by Räcke takes exponential time. In a subsequent paper with Bienkowski and Korzeniowski [5], a polynomial time algorithm has been proposed. A similar result was also achieved independently by Harrelson *et al.* [16]. We note that, however, both results assume the maximum link capacity is polynomially bounded.

Räcke introduced two connectivity characteristics to measure the quality of a decomposition. First we need to present some definitions. Let $b(x, y)$ be the *bandwidth* (or weight) of the edge $(x, y) \in E$ and let $\text{cap}(X, Y)$ be the *capacity* between the set $X \subseteq V$ and $Y \subseteq V$, *i.e.*, $\text{cap}(X, Y) = \sum_{x \in X, y \in Y} b(x, y)$. We will use $\text{out}(X)$ as a shorthand for $\text{cap}(X, V \setminus X)$. Consider a non-root node $v_t \in V_T$. The bandwidth 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 particular 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 bandwidth of all the edges with at least one end-point in X that *cross* the boundary of the level l -decomposition of G . Finally, Räcke's algorithm also guarantees that, among other things, T has height at most $O(\log n)$.

4.2 The Räcke Complement and APMFPs

In our analysis, we will consider the embedding of the CMCFPs into G and apply the Congestion-Dilation Corollary 2.9. First, we need to review the CMCFPs specified by Räcke.

In an initialization phase, Räcke constructs a CMCFP in S_{u_t} for each $u_t \in V_T$. Suppose u_t is a level $(l - 1)$ node u_t 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 $u, v \in S_{u_t}$. The source and sink of $f_{u,v}$ are u and v respectively and the demand is

$$d_{u,v} = \frac{w_{l+1}(v)}{w_{l+1}(S_{v_{ti}})} \cdot \text{out}(S_{v_{ti}}) \cdot \frac{w_l(u)}{w_l(S_{u_t})}, \quad (4.1)$$

where $S_{v_{ti}}$ denotes the level l cluster that contains v . The flows are restricted to use only edges inside S_{u_t} . Let q be the throughput fraction of a solution, *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 to the CMCFP maximizes q . Note that if we send only $q \cdot d_{u,v}$ flow for $f_{u,v}$, then the total flow that traverses an edge $e \in E$ inside S_{u_t} is no more than its bandwidth $b(e)$. Räcke showed that all the $|V_T|$ sets of demands in the CMCFPs can be simultaneously satisfied in G while inducing only $O(\log^3 n)$ congestion, given the decomposition tree constructed by his algorithm; that is, there is a simultaneous solution with $q = \Omega(1/\log^3 n)$. (His result improves to $O(\log^2 n)$ when G is planar or has constant genus, but our result will be stated using the more general $O(\log^3 n)$ bound.)

The Räcke Complement. We view the CMCFP corresponding to u_t as a complete graph K_{u_t} induced 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.*, $d_{u,v} + d_{v,u}$. We call the overlapping of the $|V_T|$ complete graphs the *Räcke complement* of T , denoted $RC(T)$. More precisely, $RC(T)$ is a complete graph with vertex set V and the weight of an edge in $RC(T)$ is the sum of its weights in all the K_{u_t} 's containing it. Using Räcke's congestion result with the Congestion-Dilation Corollary 2.9, we have

$$\sigma(RC(T)/G) \leq O(\log^3 n) \cdot \text{dil}_R(G), \quad (4.2)$$

where $\text{dil}_R(G)$ is the dilation when we embed Räcke's CMCFP solutions into G , *i.e.*, the maximum length of the flow paths in the CMCFP solutions.

4.2.1 Bounding $\text{dil}(G)$ via APMFP

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 (since removing loops may only lower the congestion).

Note that our bound has the potential to be improved if we can bound on the length of the flow paths using properties that are more sensitive to G . Fortunately, we observe that the demand in (4.1) actually resembles the product multicommodity flow problem (PMFP) defined by Leighton and Rao [17], albeit being slightly more general, and the flow path lengths for PMFPs do have a tighter bound.

Introducing APMFP. For our analysis, we define a generalization of PMFP called the *asymmetric product multicommodity flow problem* (APMFP) and show for any undirected asymmetric product flow on an undirected graph $G = (V, E)$ how to find a cut with weighted ratio cost $O(f \log n)$, where f is the maximum common fraction of each commodity that can be routed and n is the number of nodes in G . (Note that we allow $f > 1$.) We then show that such a cut exists even if flow paths are restricted in length.

Leighton and Rao showed how to find such a cut when the undirected demand for each pair of nodes $u \in V$ and $v \in V$ is $\pi(u)\pi(v)$, where $\pi(u) \geq 0$ for all $u \in V$. They called this a product multicommodity flow problem; we would call it a symmetric product multicommodity flow problem.

In an APMFP, the undirected demand between each pair of nodes $u \in V$ and $v \in V$ is

$$\sigma(u)\tau(v) + \sigma(v)\tau(u),$$

where $\sigma(u), \tau(u) \geq 0$ for all $u \in V$. The product flow in [17] can then be written as an APMFP where, for example, $\sigma(u) = \pi(u)/2$ and $\tau(u) = \pi(u)$. (Note that σ is also being used to denote the support between two graphs. However, this overloading should not cause any confusion because of their very different arguments, *e.g.*, $\sigma(G/T)$ versus $\sigma(u)$.)

In order to restate (for APMFPs) Theorem 18 of [17] on routing multicommodity flow on short paths, we first have to introduce a few more definitions.

The definitions involving the demand are the following. For a set $U \subseteq V$, let $\sigma(U) = \sum_{u \in U} \sigma(u)$ and $\tau(U) = \sum_{u \in U} \tau(u)$. Let (although this is abusive) $\binom{V}{2} = \{\{u \in V, v \in V\} \mid u \neq v\}$ denote the set of all *unordered* pairs of vertices drawn from V . Let $D = \sum_{\{u,v\} \in \binom{V}{2}} \sigma(u)\tau(v) + \sigma(v)\tau(u)$ denote the total demand.

The definitions involving the capacities of edges are the following. As in [17], each edge $e \in E$ has capacity $C(e) \geq 0$, and the capacity of a cut (U, \bar{U}) is defined as $C(U, \bar{U}) = \sum_{e \in \{\{u,v\} \mid u \in U, v \in \bar{U}\}} C(e)$.

Let $\delta(u)$ denote the sum of the capacities of the edges incident on u , and for $U \subseteq V$, let $\delta(U) = \sum_{u \in U} \delta(u)$ and $C = \delta(V)/2$. Let $\delta_{\min} = \min_{u \in V} \delta(u)$. Finally, let $\rho(\langle U, \bar{U} \rangle) = \frac{C(U, \bar{U})}{\delta(U)\delta(\bar{U})}$, and $\rho = \min_{\langle U, \bar{U} \rangle} \rho(\langle U, \bar{U} \rangle)$. Note that ρ is a function of the graph only, and not of the demands, and that $C\rho$ does not change if all capacities are multiplied by a positive scaling factor.

For an APMFP, we need a new definition of the weighted ratio cost,

$$\mathcal{S}(U, \bar{U}) = \frac{C(U, \bar{U})}{\sigma(U)\tau(\bar{U}) + \sigma(\bar{U})\tau(U)}.$$

For the entire graph, $\mathcal{S} = \min_{\langle U, \bar{U} \rangle} \mathcal{S}(U, \bar{U})$. Note that $f \leq \mathcal{S}$.

Theorem 4.1 (Theorem B.18) *Given any n -node APMFP for which the minimum weighted ratio cut has value \mathcal{S} , there is a flow with $f \geq \Omega(\mathcal{S}/\log n)$ for which every flow path has length at most*

$$L \leq O\left(\max\left\{\frac{\log(C/\delta_{\min})}{C\rho}, \frac{C \log n}{\mathcal{S}D}\right\}\right).$$

We provide the full proof of this theorem in Appendix B for completeness.

Bounding $\sigma(RC(T)/G)$ via APMFP. With Theorem 4.1, we can now go back and bound $\sigma(RC(T)/G)$ using APMFP. (Note that it is technically incorrect to bound $dil_R(G)$ in (4.2) simply by invoking Theorem 4.1. The embedding specified by the solutions to Racke’s CMCFPs and our APMFPs can be different.)

We start by viewing the demands specified by (4.1) as an instance of APMFP. In particular, suppose v is in a level l cluster $S_{v_{ti}}$ which in turn is contained in a level $(l - 1)$ cluster S_{u_t} , *i.e.*, v_{ti} is a child of u_t . Then let $\sigma(v) = w_{l+1}(v)/w_{l+1}(S_{v_{ti}}) \cdot out(S_{v_{ti}})$ and let $\tau(v) = w_l(v)/w_l(S_{u_t})$. The flow between any pair of nodes u and v will then be $\sigma(u)\tau(v) + \sigma(v)\tau(u)$. Therefore, by Theorem 4.1, we have the following corollary where n, C, D, δ_{\min} and ρ are as defined above.

Corollary 4.2 *The congestion of the embedding of $RC(T)$ into G is at most $O(\log^3 n)$, and the dilation is bounded by*

$$dil(G) = O\left(\max\left\{\frac{\log(C/\delta_{\min})}{C\rho}, \frac{C \log^2 n}{D}\right\}\right).$$

Proof The proof follows from Theorem 4.1, and Racke’s proof [19, Lemma 4] that for these demands $\mathcal{S} \geq \Omega(1/\log n)$.

We begin by bounding the first term. Racke proved [19, Lemma 4] that for each APMFP, $\mathcal{S} \geq \Omega(1/\log n)$. By Theorem 4.1, therefore, the congestion of the paths routing each APMFP is at most $O(\log^2 n)$. For each level of T , the APMFPs corresponding to the nodes of T on that level are routed in disjoint subgraphs of G , so that congestion for all of these APMFPs taken together is still $O(\log^2 n)$. There are $O(\log n)$ levels in the tree, so that the total congestion is $O(\log^3 n)$. The lengths of these paths (dilation) is bounded by Theorem 4.1. ■

4.3 Towards $\kappa(T, G)$

As we have shown in Section 2.4, the quality of T as a preconditioner for G in STCG depends on $\sigma(G/T)$ and $\sigma(T/G)$. We now bound these terms separately.

4.3.1 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 for the congestion, consider an edge $(x, y) \in E$, which corresponds to a particular leaf-to-leaf path in T that only uses nodes v_t iff (x, y) is a boundary edge of S_{v_t} . For each of the tree edge (u_t, v_t) on this path with u_t being the parent node, the bandwidth assigned is $out(S_{v_t})$ by the construction of T . Clearly, each of these tree edges has sufficient bandwidth reserved for (x, y) and so the congestion cannot exceed 1. Therefore, by the Congestion-Dilation Corollary 2.9, we have

$$\sigma(G/T) \leq O(\log n). \tag{4.3}$$

4.3.2 Bounding $\sigma(T/G)$

By Lemma 2.2, we have

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

Notice that the second term has already been bounded by the Congestion-Dilation Corollary 2.9 and Corollary 4.2.

We now show that the first term is bounded by 1 using the following electricity argument. By Corollary 3.4, $\sigma(T/RC(T)) \leq 1$ can be proved if we establish the following statement: for any voltage setting \mathbf{v} (an n -vector) on the nodes in $RC(T)$, the power dissipated in $RC(T)$ is at least the power dissipated in T with its terminal voltages set to \mathbf{v} .

First we fix an arbitrary \mathbf{v} and consider the power dissipation in T . Suppose the junction voltages of T are the naturally-occurring voltages as determined by Kirchoff’s Law using the terminal voltages \mathbf{v} . By the Dirichlet Principle (which is also the dual of Thomson’s Principle; see [11, p. 64]), any other

voltage settings on the junctions can only increase the total power dissipation of T . For the purpose of bounding $\sigma(T/RC(T))$, we will indeed pick another set of junction voltages and bound the (potentially) increased power dissipation instead. In particular, let the voltage at a level $(l-1)$ node u_t be $\sum_{u \in S_{u_t}} (w_l(u)/w_l(S_{u_t})) \mathbf{v}_u$. Note that the fraction actually corresponds to the probability that u_t is mapped to u in Racke's randomized embedding and thus the voltage at u_t is a convex combination of the voltages in S_{u_t} .

With both the junction and terminal voltages of T in place, we can calculate and bound its power dissipation. 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

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

Recall that the two fractions are both probabilities, and so by Lemma 3.5, this is at most

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

Since the $S_{v_{ti}}$'s form a partition of S_{u_t} , there are exactly $|S_{u_t}|^2$ terms in the sum. Each node $u \in S_{u_t}$ occurs $|S_{u_t}| + 1$ times, once as v and the other times as u . With (4.1), we simplify and obtain $\sum_{u,v \in S_{u_t}} (d_{u,v} + d_{v,u})(\mathbf{v}_v - \mathbf{v}_u)^2$, which in turn is the power dissipated in K_{u_t} when the voltage setting is specified by (the corresponding part of) \mathbf{v} . Hence, for any terminal voltages, K_{u_t} dissipates at least as much power as the group of edges directly under u_t at the naturally-occurring junction voltages.

The final step is to observe that power dissipation is additive on conductance. (Recall that the power dissipation across a conductor is its conductance times the square of the voltage across.) Hence, the power dissipated in $RC(T)$ is the sum of the power dissipated in K_{u_t} 's. Therefore,

$$\sigma(T/RC(T)) \leq 1. \tag{4.5}$$

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

By combining Definition 2.3, (4.3), (4.4) and (4.5) and applying the Congestion-Dilation Corollary 2.9 using Corollary 4.2, we have our main result, where n, C, D, δ_{\min} and ρ are as defined above.

Theorem 4.3 *Let T be the decomposition tree constructed by Racke's algorithm.*

$$\kappa(T, G) = O \left(\max \left\{ \frac{(\log^4 n) \log(C/\delta_{\min})}{C\rho}, \frac{C \log^6 n}{D} \right\} \right)$$

4.4.1 Example Applications

We will apply Theorem 4.3 on some simple common graphs for examples.

Uniform Square Meshes. Consider a square mesh with $\sqrt{n} \times \sqrt{n}$ nodes with uniform edge weight of 1. We have $\delta_{\min} = 1$, $C = n$, $D = \theta(\sqrt{n})$, $\rho = n^{-3/2}$ and $\mathcal{S} = O(\log n)$. Plugging these into Theorem 4.3, we have $\kappa(T, G) = O(\sqrt{n} \log^6 n)$.

Uniform Expanders. Consider a bounded-degree expander graph with n nodes with uniform edge weight of 1. We have $\delta_{\min} = 1$, $C = n$, $D = \Theta(n)$, $\rho = O(1/n)$ and $\mathcal{S} = O(\log n)$. Plugging these into Theorem 4.3, we have $\kappa(T, G) = O(\log^6 n)$.

Acknowledgements

We would like to thank Eduardo Laber and Andrea Richa for fruitful discussions.

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. [1.1](#)
- [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. [4.2.1](#)
- [3] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1996. [1](#)
- [4] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 2002. to appear. [2.3](#)
- [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. [1.2](#), [4.1](#)
- [6] E. Boman, D. Chen, B. Hendrickson, and S. Toledo. Maximum-weight-basis preconditioners. *Lin. Alg. Appl.*, 2002. To Appear. [1.1](#)
- [7] E. Boman and B. Hendrickson, 2002. Personal communication. [1.1](#)
- [8] E. Boman and B. Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 2002. Submitted. [2.3](#), [2.3](#), [2.4](#)
- [9] D. Chen. Analysis, implementation, and evaluation of Vaidya’s preconditioners. Master’s thesis, School of Mathematical Sciences, Tel-Aviv University, 2001. [1.1](#)
- [10] D. Chen and S. Toledo. Implementation and evaluation of Vaidya’s preconditioners. In *Preconditioning 2001*, Tahoe City, CA, 2001. [1.1](#)
- [11] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*, volume 22 of *Carus Mathematical Monographs*. Mathematical Association of America, 1984. [2.3](#), [3](#), [4.3.2](#)
- [12] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3 edition, 1996. [1](#)
- [13] 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. [1](#), [1.1](#), [1](#), [2.3](#), [2.3](#), [2.4](#), [3](#), [C](#)
- [14] K. Gremban, G. Miller, and M. Zagha. Performance evaluation of a parallel preconditioner. CS CMU-CS-94-205, CMU, October 1994. [1.1](#)
- [15] L. A. Hageman and D. M. Young. *Applied Iterative Methods*. Computer Science and Applied Mathematics. Academic Press, Inc, San Diego and London, 1981. [2.1](#), [2.3](#)
- [16] 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. [1.2](#), [4.1](#)
- [17] 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. [4.2.1](#), [4.2.1](#), [B.1](#), [B.1](#), [B.1](#), [B.2](#), [B.2](#)
- [18] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16:346–358, 1979. [1](#)
- [19] H. Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 43–52. IEEE, 2002. [1.2](#), [4](#), [4.1](#), [4.2.1](#)
- [20] J. Reif. Efficient approximate solution of sparse linear systems. *Computers and Mathematics, with Applications*, 36, 1998. [1.1](#)
- [21] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1996. [1](#)
- [22] D. Spielman and S. Teng. Solving SPDDD linear systems in time $O(m^{1.31})$. In *Proceedings of the Forty-Forth Annual Symposium on Foundations of Computer Science*, 2003. [1.1](#)
- [23] P. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. A talk based on this manuscript, October 1991. [1.1](#), [2.3](#)

A Proofs for Laplacians as Circuits

This appendix contains the proofs omitted in Section 3.

Lemma A.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 A.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 $A = \Gamma^T W \Gamma$.*

Proof of Lemma 3.1:

By Lemma A.1, we have

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

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 A.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 $\Gamma \mathbf{y} = \sum_{j=1}^n \mathbf{y}_j \Gamma_{(j)}$ and thus

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

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

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

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

$$\mathbf{x}^T M \mathbf{x} = \sum_{i=1}^m \mathbf{x}_i \left(\sum_{j=1}^m \mathbf{x}_j M_{i,j} \right) = \sum_{(i,j) \in \{m \otimes m\}} \mathbf{x}_i \mathbf{x}_j M_{i,j},$$

we have

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

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

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

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:

RHS – LHS

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

(cancelling the second term with the fifth term)

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

 (α 's and β 's both sum to δ)

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

 (α 's and β 's both sum to δ)

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

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

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

 (α_i 's and β_j 's are nonnegative)

$$\geq 0$$

■

B Asymmetric Product Multicommodity Flow

In this section, we prove the following theorem.

Theorem B.18 *Given any n -node APMFP for which the minimum weighted ratio cut has value \mathcal{S} , there is a flow with $f \geq \Omega(\mathcal{S}/\log n)$ for which every flow path has length at most*

$$L \leq O\left(\max\left\{\frac{\log(C/\delta_{\min})}{C\rho}, \frac{C \log n}{\mathcal{S}D}\right\}\right).$$

B.1 Finding a Minimum Cut

In the remainder of this section, we restate Lemmas 8, 10, and 11 and Corollary 9 from [17] for an APMFP, and describe the changes in their proofs. Our proof follows that in [17] lemma-by-lemma using the same numbering as in their paper. The net result is to describe an algorithm for finding a cut $\langle U, \bar{U} \rangle$ such that $\mathcal{S}(U, \bar{U}) \leq O(f \log n)$. Since $\mathcal{S}(U, \bar{U}) \geq \mathcal{S}$, this implies that $f \geq \Omega(\mathcal{S}/\log n)$.

As in [17], the solution to the dual of the multicommodity flow problem is a distance function d on the undirected edges of G . The new distance constraint from the dual problem is

$$\sum_{\{u,v\} \in \binom{V}{2}} (\sigma(u)\tau(v) + \sigma(v)\tau(u))d(u,v) \geq 1.$$

In place of Lemma 8 in [17], we use Lemma 3 of the same paper. (In [17], Lemma 8 is written in terms of p , the number of nodes u such that $\pi(u) > 0$, whereas Lemma 3 is written in terms of n , the number of nodes in G . For our purposes, Lemma 3 suffices.) The term W in the statement of Lemma 3 is the total weight of the distance function, *i.e.*,

$$W = \sum_{e \in E} C(e)d(e).$$

By duality $W = f$, *i.e.*, the total weight of the distance function is equal to the maximum common fraction of flow that can be routed. We restate the lemma below, but omit the proof because no changes are required.

Lemma B.8 *For any graph G with arbitrary edge capacities, any $\Delta > 0$, and any distance function with total weight W , it is possible to partition G into components with radius at most Δ so that the capacity of the edges connecting nodes in different components is at most $4W \log n/\Delta$.*

The major change in Corollary 9 is that the component T in case (1) must have at least $2/3$ of both the σ weight and the τ weight, rather than $2/3$ of the π weight. The radius of T in case (1) also changes from $1/p^2$ to $1/2D$. For convenience, let $s = \sigma(V)$, and $t = \tau(V)$.

Corollary B.9 *For any graph G and any distance function d with total weight W , we can either*

1. *find a component T with radius $1/2D$ for which $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$, or*
2. *find a cut $\langle U, \bar{U} \rangle$ of G with weighted ratio cost $\mathcal{S}(U, \bar{U}) > 72W \log n$.*

Proof The proof follows the proof of Corollary 9 in [17] closely. The term p^2 in that proof is replaced everywhere by D , so that we apply Lemma 8 with $\Delta = 1/2D$.

After applying Lemma 8, if there is a component T such that $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$, then condition (1) is satisfied.

Otherwise, if there is no component T for which $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$, then we form a cut as follows. If there is any component T for which $\sigma(T) \geq 2s/3$, put that component by itself on one side of the cut. For the remainder of the graph $\tau(V \setminus T) > t/3$, so that we have demand strictly greater than $2st/9$ across the cut. There is a symmetric case when $\tau(T) \geq 2t/3$.

We are left to consider the case in which there is no component T with $\sigma(T) \geq 2s/3$ or $\tau(T) \geq 2s/3$. If there is any component T with $\sigma(T) > s/3$, put it on one side of the cut. On the other side $\tau(V \setminus T) \geq t/3$, so we have a demand strictly greater than $st/9$. As before, there is a symmetric case for $\tau(T) > t/3$.

If there is no component T with either $\sigma(T) > s/3$ or $\tau(T) > t/3$, then we form the cut using the following algorithm. If there is a component T with $\sigma(T) = s/3$, but $\tau(T) < t/3$, then put T on one side by itself. Since $\tau(V \setminus T) > 2t/3$, the demand is strictly greater than $2t/9$. There is a symmetric case when $\tau(T) = t/3$.

So (at long last), we are left with the case in which $\sigma(T) < s/3$ and $\tau(T) < t/3$ for all components in T . Start building one side of the cut U by placing components there one after another. Stop as soon as either $\sigma(U) > s/3$ or $\tau(U) > t/3$. At this point, it must be the case that $\sigma(\bar{U}) > s/3$, and $\tau(\bar{U}) > t/3$, so that the demand across the cut is strictly greater than $st/9$.

To complete the proof, we note that by Lemma B.8 the capacity of the cut is at most $4W \log n / \Delta = 8WD \log n$. This gives us an upper bound

$$\mathcal{L}(U, \bar{U}) < \frac{8WD \log n}{st/9} \leq 72W \log n$$

(since $D \leq st$) on the weighted ratio cost of the cut. ■

Lemma B.10 *For any graph G and functions σ and τ on V where, for all u , $\sigma(u) \geq 0$ and $\tau(u) \geq 0$, if there is a distance function d with total weight W and a subset of nodes T such that $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$ and*

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T, u) \geq \frac{1}{2},$$

then we can display a cut with ratio cost $O(W)$.

Proof We begin with some definitions borrowed from [17]. For $i \geq 0$, let G_i^+ be the subgraph of G^+ consisting of all nodes and edges that are within distance i of T . Note that T is included in G_i^+ . Let V_i be the set of nodes of G that correspond to nodes of G_i^+ , let R_i be the weighted ratio cost of the cut $\langle V_i, V \setminus V_i \rangle$, and let $R = \min\{R_i\}$. We add two new definitions of our own: let $s_i = \sigma(V \setminus V_i)$ and $t_i = \tau(V \setminus V_i)$. Then for all i we know that

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d_{G^+}(T, u) = \sum_{i \geq 0} s_i t + st_i.$$

Since $d_{G^+}(T, u) \geq (C/W)d_G(T, u)$, and by assumption $\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d_G(T, u) \geq \frac{1}{2}$, we have

$$\sum_{i \geq 0} s_i t + st_i \geq \frac{C}{W} \sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d_G(T, U) \geq \frac{C}{2W}. \quad (2.6)$$

Since both $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$, the capacity in the cut $\langle V_i, V \setminus V_i \rangle$ is at least $R_i(2s_i t/3 + 2st_i/3) \geq R(2s_i t/3 + 2st_i/3)$. The capacity of the corresponding cut for G_i^+ in G^+ is at least this large, and since the total capacity C^+ in G^+ is at most $2C$, we have

$$\sum_{i \geq 0} \frac{2R(s_i t + st_i)}{3} \leq 2C,$$

which we can rewrite as

$$R \leq \frac{3C}{\sum_{i \geq 0} s_i t + s t_i}.$$

Using Equation 2.6, we can conclude

$$R \leq 6W. \quad \blacksquare$$

The only change to the statement of Lemma 11 is that we find a cut with weighted ratio cost $O(W \log n)$ rather than $O(W \log p)$. The proof, however, requires several changes.

Lemma B.11 *For any graph G and functions σ and τ on V where, for all u , $\sigma(u) \geq 0$ and $\tau(u) \geq 0$, and any distance function that satisfies the distance constraint, we can find a cut with weighted ratio cost $O(W \log n)$.*

Proof We begin by applying Lemma B.8 with $\Delta = 1/2D$. Then by Corollary B.9, we either have a cut with weighted ratio cost $O(W \log n)$, or there is a component T with radius $1/2D$ such that $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$. In the former case we are done. In the latter, observe that

$$\begin{aligned} \sum_{\{u,v\} \in V^2} (\sigma(u)\tau(v) + \sigma(v)\tau(u))d(u,v) &= \frac{1}{2} \sum_{u \in V} \sum_{v \in V, v \neq u} (\sigma(u)\tau(v) + \sigma(v)\tau(u))d(u,v) \\ &\leq \frac{1}{2} \sum_{u \in V} \sum_{v \in V, v \neq u} (\sigma(u)\tau(v) + \sigma(v)\tau(u)) (d(T,u) + d(T,v) + 1/D) \\ &= \sum_{u \in V} \sum_{v \in V, v \neq u} (\sigma(u)\tau(v) + \sigma(v)\tau(u)) (d(T,u) + 1/2D) \\ &\leq \frac{n^2}{2D} + \sum_{u \in V} (\sigma(u)t + s\tau(u))d(T,u) \\ &= \frac{1}{2} + \sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T,u) \end{aligned}$$

The second line follows from the fact that there is a path from u to v that first travels from u to T , then within T a distance at most twice the radius, $1/D$, of T , and finally from T to v . The third line holds by symmetry. In the fourth line we have brought $1/2D$ out of the sum and observed that $\sum_{v \in V} \tau(v) = t$ and $\sum_{v \in V} \sigma(v) = s$. The fifth line uses the scaling assumption that $D = n^2$.

By the distance constraint, $\sum_{\{u,v\} \in V^2} (\sigma(u)\tau(v) + \sigma(v)\tau(u))d(u,v) \geq 1$, so we can conclude

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T,u) \geq \frac{1}{2}.$$

This last inequality, along with $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$, allows us to apply Lemma 10 to find a cut with weighted ratio cost $O(W)$. \blacksquare

B.2 Bounding the Lengths of the Flow Paths

In this section, we prove the existence of a flow with maximum common fraction $f \geq \Omega(\mathcal{S}/\log n)$ even when the flow paths are restricted in length.

There is no change to Lemma 19. The term *distance-radius* in the statement of the lemma refers to distances measured using the distance function d , defined on the edges of e , which is the solution to the

dual problem. The term *edge-radius* refers to the length of the path in G , measured in number of edges traversed, *i.e.*, hop-length. We restate Lemma 19 here.

Lemma B.19 *For any graph G with total capacity C , any $\Delta > 0$ and any distance function with total weight W , it is possible to partition G into components with edge-radius $\Delta C/W$ and distance-radius Δ so that the capacity of the edges connecting nodes in different components is at most $4W \log n/\Delta$.*

Corollary 20 is restated as follows.

Corollary B.20 *For any graph G with minimum weighted ratio cost \mathcal{S} and any distance function with total weight $W \leq \mathcal{S}/72 \log n$, we can find a component T of G with edge radius $C/2WD$ and distance radius $1/2D$ such that $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$.*

Proof The proof follows directly from the proof of Corollary B.9. ■

Our statement and proof of Lemma B.21 is different from that of Lemma 21 of [17] in one key respect. The upper bound that we prove on the length of the paths is given in terms of ρ , rather than \mathcal{S} . The problem with Lemma 21 in [17] is that for PMFPs, the length of the path between two nodes u and v is defined to be the number of positive-weight intermediate nodes (*i.e.*, nodes x such that $\pi(x) > 0$) that the path passes through in G . For our purposes, we need the true path length in G .

The Lemma 21 is restated below. The length L is defined to be the total number of edges on a path.

Lemma B.21 *For any graph G with minimum weighted ratio cost \mathcal{S} , any distance function with total weight W , any length $L \geq (48 \ln(3C/2\delta_{\min}))/\rho C$, and any subset of nodes $T \subseteq V$ for which $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$,*

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T, u) \leq \frac{6W}{\mathcal{S}},$$

where $d(T, u)$ is computed using only paths with at most $L/4$ edges.

Proof The proof is similar to that of Lemma B.10, but the definition of G_i^+ is more involved. We use the same step-by-step method that is used in the *unweighted* (UMFP) version of Lemma 21 of [17]. We start with G_0^+ , which is the subgraph of G^+ induced by the nodes of T . Now G_{i+1}^+ is defined in terms of G_i^+ as follows. Let S_i be the set of edges with one end-point in G_i^+ and one end-point outside of G_i^+ , and let C_i be the sum of the capacities of these edges. Let C'_i be the sum of the capacities of the edges whose end-points outside of G_i^+ are nodes in G . If $C'_i \geq C_i/2$, then G_{i+1}^+ is the graph consisting of all of the nodes within distance 1 of G_i^+ , and the edges induced by these nodes. On the other hand, if $C'_i < C_i/2$, then G_{i+1}^+ is the graph consisting of G_i^+ and all of the nodes in $G^+ \setminus G$ that are exactly distance 1 from G_i^+ , *i.e.*, nodes in G that are distance 1 from G_i^+ are not included.

We now show that the maximum distance (in G) between T and any node u is at most $12 \ln(3C/2\delta_{\min})/C\rho$. The key idea is that we only incorporate nodes of G into G_{i+1}^+ on steps when at least half of the capacity of the edges out of G_i^+ lead to such nodes, and it is only on these steps that the distance from the tree (measured in terms on number of edges in G) is seen to increase by one. Let G_i denote the set of nodes in G contained in G_i^+ , and let V_i be the nodes in G_i . Suppose that $\delta(V_i) \geq C/2$ (where C is the total capacity of G). Then

$$\rho(\langle V_i, V \setminus V_i \rangle) = \frac{C_i}{\delta(V_i)\delta(V \setminus V_i)} \leq \frac{2C_i}{\delta(V \setminus V_i)C}.$$

Since $\rho(\langle V_i, V \setminus V_i \rangle) \geq \rho$, we have the following lower bound

$$C'_i \geq \frac{C_i}{2} \geq \frac{\rho\delta(V \setminus V_i)C}{4},$$

which tells us that

$$C'_i = \delta(V \setminus V_i) - \delta(V \setminus V_{i+1}) \geq \frac{\rho \delta(V \setminus V_i) C}{4},$$

so that

$$\delta(V \setminus V_{i+1}) \leq \left(1 - \frac{\rho C}{4}\right) \delta(V \setminus V_i),$$

i.e., in each step we capture a $\rho C/4$ fraction of the remaining capacity of the nodes of G not yet in G_i^+ . Since $\delta(V_i) \geq C/2$, and $\delta(V) = 2C$ (since each edge is counted twice in $\delta(V)$), $\delta(V \setminus V_i) \leq 3C/2$. There is no more capacity to capture once the remaining capacity drops below δ_{\min} , so the total number of steps before there is no more capacity is at most

$$\frac{\ln(3C/2\delta_{\min})}{\ln \frac{1}{1 - \frac{\rho C}{4}}}.$$

Using the inequality $1/(1-x) \geq 1+x+x^2 \geq e^x$ for $0 \leq x \leq 1$, and the fact that $\rho C \leq 1$, we can simplify the upper bound on the number of steps to

$$\frac{4 \ln(3C/2\delta_{\min})}{\rho C}.$$

We also have to count the number of steps required to get to the step where $\delta(V_i) \geq C/2$. Until this happens, *i.e.*, when $\delta(V_i) < C/2$,

$$\rho(\langle V_i, V \setminus V_i \rangle) = \frac{C_i}{\delta(V_i) \delta(V \setminus V_i)} \leq \frac{2C_i}{\delta(V_i) C},$$

from which we can conclude that

$$\delta(V_{i+1}) \geq \delta(V_i) \left(1 + \frac{\rho C}{4}\right).$$

Since $\delta(V_0)$ might be as small as δ_{\min} , the number of steps required until $C_i \geq C/2$ might be as large as

$$\frac{\ln \frac{C}{2\delta_{\min}}}{\ln 1 + \frac{\rho C}{4}}.$$

Since $\rho C \leq 1$ and, for $0 \leq x \leq 1$, $1+x \geq e^{x/2}$, the upper bound can be simplified to

$$\frac{8 \ln \frac{C}{2\delta_{\min}}}{\rho C}.$$

Summing the number of steps to reach the step where $\delta(V_i) \geq C/2$, and then the number of steps until all of the remaining capacity is captured, we have a total of at most

$$\frac{12 \ln \frac{3C}{2\delta_{\min}}}{\rho C}.$$

We use a different analysis to show that

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T, u) \leq \frac{6W}{\mathcal{F}}.$$

As in Lemma B.10, we let R_i be the weighted ratio cost of the cut $\langle V_i, V \setminus V_i \rangle$, let $R = \min\{R_i\}$, let $s_i = \sigma(V \setminus V_i)$, and let $t_i = \tau(V \setminus V_i)$. Then for all i we know that

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d_{G^+}(T, u) = \sum_{i \geq 0} s_i t + st_i.$$

Since both $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$, the capacity $C'_i \geq C_i/2$ is at least $R_i(s_i t/3 + st_i/3) \geq R(s_i t/3 + st_i/3)$. The capacity of the corresponding cut for G_i^+ in G^+ is at least this large, and since the total capacity C^+ in G^+ is at most $2C$, we have

$$\sum_{i \geq 0} \frac{R(s_i t + st_i)}{3} \leq 2C,$$

which we can rewrite as

$$\sum_{i \geq 0} s_i t + st_i \leq \frac{6C}{R}.$$

We can now conclude our upper bound on $\sum_{u \in V \setminus T} d(T, u)$. Let $d_{G^+}^*(T, u)$ denote the length in G^+ of the path from T to u that is formed by our method of growing G_i^+ and let $d_G^*(T, u)$ denote the length of the corresponding path in G .

$$\begin{aligned} \sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d_G(T, u) &\leq \sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d_{G^+}^*(T, u) \\ &\leq \frac{W}{C} (\sigma(u)t + s\tau(u))d_{G^+}^*(T, u) \\ &\leq \frac{W}{C} \sum_{i \geq 0} s_i t + st_i \\ &\leq \frac{6W}{R} \end{aligned}$$

■

Proof of Theorem B.18:

The proof is nearly identical to that of the PMFP version of Theorem 18 of [17]. Let

$$L/4 = \max \left\{ \frac{12 \ln \frac{3C}{2\delta_{\min}}}{\rho C}, \frac{36C \log n}{\mathcal{S}D} \right\}$$

be one-fourth of the maximum path length. We want to show that $W \geq (\mathcal{S}/72 \log n)$ if the distance constraint (when restricted to paths of length at most L) is satisfied. Suppose $W \leq \mathcal{S}/72 \log n$. By Corollary B.20, we can find a component T of G with edge radius $(C/2WD) \leq L/4$ and distance radius $1/2D$, such that $\sigma(T) \geq 2s/3$ and $\tau(T) \geq 2t/3$. By Lemma B.21, we have

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T, u) \leq \frac{6W}{\mathcal{S}},$$

where $d(T, u)$ is computed using only paths of length $L/4$ or less. If $W \leq \mathcal{S}/72 \log n$, then there is a

feasible solution with $W = \mathcal{S}/72 \log n$. The bound from Lemma B.21 then becomes

$$\sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T, u) \leq \frac{1}{12 \log n}.$$

Hence,

$$\begin{aligned} \sum_{\{u,v\} \in V^2} (\sigma(u)\tau(v) + \sigma(v)\tau(u))d(u, v) &= \frac{1}{2} \sum_{u \in V} \sum_{v \in V, v \neq u} (\sigma(u)\tau(v) + \sigma(v)\tau(u))d(u, v) \\ &\leq \frac{1}{2} \sum_{u \in V} \sum_{v \in V, v \neq u} (\sigma(u)\tau(v) + \sigma(v)\tau(u)) (d(T, u) + d(T, v) + 1/D) \\ &= \sum_{u \in V} \sum_{v \in V, v \neq u} (\sigma(u)\tau(v) + \sigma(v)\tau(u)) (d(T, u) + 1/2D) \\ &\leq \frac{n^2}{2D} + \sum_{u \in V} (\sigma(u)t + s\tau(u))d(T, u) \\ &= \frac{1}{2} + \sum_{u \in V \setminus T} (\sigma(u)t + s\tau(u))d(T, u) \\ &\leq \frac{1}{12 \log n} + \frac{1}{2} \\ &< 1 \end{aligned}$$

But this violates the distance constraint, so it must be that $W > 72/\mathcal{S} \log n$. Note that in the second inequality, we are bounding $d(u, v)$ by the length of a path consisting of four segments, each of which has length at most $L/4$. \blacksquare

B.3 Scaling

Recall that we assumed that $D = n^2$. If we are given an APMFP for which $D \neq n^2$, then we can scale the demands by multiplying each $\sigma(u)$ by a scaling factor m where $m = n^2/D$. Let $D' = mD$ be the demand for the scaled problem, and let f' and \mathcal{S}' be the maximum common fraction and weighted cost ratio, respectively, for the scaled problem. Then $\mathcal{S}' = \mathcal{S}/m$ because the demand across any cut has increased by a factor of m . We proved that $f' \geq \Omega(\mathcal{S}'/\log n) = \Omega((\mathcal{S}/m)/\log n)$. Any solution to the scaled problem with common fraction f' is also a solution to the original problem, but with common fraction $f'm$. Hence

$$f \geq f'm \geq \Omega(m(\mathcal{S}/m)/\log n) = \Omega(\mathcal{S}/\log n).$$

Similarly, let L' be the upper bound that we proved on the lengths of the paths for the scaled problem. We showed that

$$\begin{aligned} L' &\leq O\left(\max\left\{\frac{\log(C/\delta_{\min})}{\rho C}, \frac{C \log n}{\mathcal{S}' D'}\right\}\right) \\ &= O\left(\max\left\{\frac{\log(C/\delta_{\min})}{\rho C}, \frac{C \log n}{(\mathcal{S}/m)(Dm)}\right\}\right) \\ &= O\left(\max\left\{\frac{\log(C/\delta_{\min})}{\rho C}, \frac{C \log n}{\mathcal{S} D}\right\}\right). \end{aligned}$$

A solution to the scaled problem with common fraction f' and path lengths bounded by L' is also a solution to the original problem with common fraction $f'm$ with path lengths bounded by $L = L'$, so there is no

change in the bound on the path lengths. In other words, the same bounds that we proved for the special case in which $D = n^2$ hold for the general case.

C Extensions

In this section we present an extension of good preconditioning 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 [13] 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, $B = \begin{pmatrix} T & U \\ U^\top & W \end{pmatrix}$ for L and to use $B' = \begin{pmatrix} T & U \\ U^\top & W + D \end{pmatrix}$ as a preconditioner for A . If we let $Q = W - U^\top T^{-1} U$, 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 $\sigma(X + Z/Y + Z) = \min\{\tau \mid \forall \mathbf{x}, \tau \cdot \mathbf{x}^\top(Y + Z)\mathbf{x} \geq \mathbf{x}^\top(X + Z)\mathbf{x}\} = \min\{\tau \mid \forall \mathbf{x}, (\tau - 1) \cdot \mathbf{x}^\top Z \mathbf{x} + \tau \cdot \mathbf{x}^\top Y \mathbf{x} \geq \mathbf{x}^\top X \mathbf{x}\} \leq \max\{1, \sigma(X/Y)\}$. \blacksquare

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 matrices 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 $A\mathbf{x} = \mathbf{b}$. If A contains positive off-diagonal elements, we can decompose 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 $A' \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ -\mathbf{b} \end{pmatrix}$, since

$$A \begin{pmatrix} \mathbf{u} - \mathbf{v} \\ \mathbf{u} + \mathbf{v} \end{pmatrix} = \frac{N\mathbf{u} - P\mathbf{v}}{2} - \frac{N\mathbf{v} - P\mathbf{u}}{2} = \mathbf{b}.$$

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.