Combinatorial and algebraic tools for optimal multilevel algorithms

Ioannis Koutis

CMU-CS-07-131

May 2007

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Gary Miller, Chair
Alan Frieze
John Lafferty
Daniel Spielman, Yale University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported in part by the National Science Foundation under grants CCR-9902091, CCR-9706572, ACI 0086093, CCR-0085982 and CCR-0122581

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Keywords: Spectral graph theory, Combinatorial linear algebra, Combinatorial scientific computing, Linear systems, Laplacians, Planar graphs

For my parents, Andreas and Triantafyllia.
Για τους γονείς μου, Ανδρέα και Τριανταφυλλιά.

Abstract

This dissertation presents combinatorial and algebraic tools that enable the design of the first linear work parallel iterative algorithm for solving linear systems involving Laplacian matrices of planar graphs. The major departure of this work from prior suboptimal and inherently sequential approaches is centered around: (i) the partitioning of planar graphs into fixed size pieces that share small boundaries, by means of a local "bottom-up" approach that improves the customary "top-down" approach of recursive bisection, (ii) the replacement of monolithic global preconditioners by graph approximations that are built as aggregates of miniature preconditioners.

In addition, we present extensions to the theory and analysis of Steiner tree preconditioners. We construct more general Steiner graphs that lead to natural linear time solvers for classes of graphs that are known a priori to have certain structural properties. We also present a graph-theoretic approach to classical algebraic multigrid algorithms. We show that their design can be recast as the construction of Steiner graph preconditioners. This observation makes algebraic multigrid amenable to a combinatorial approach that provides natural graph-theoretical goals and provably fast parallel algorithms for the design of the two-level scheme.

Acknowledgements

I would like to thank my advisor Gary Miller. His insights, knowledge, support and constant availability made this dissertation possible.

I also wish to thank my committee members; John Lafferty for introducing me to some great research topics; Alan Frieze for valuable discussions and for encouraging me to submit my first paper; Daniel Spielman for very helpful conversations and his feedback that helped me improve this dissertation. Dan's work kept coming up and influencing me in almost all of my seemingly unrelated research efforts.

Overcoming the difficulties that I encountered throughout the years took some self-confidence. I feel that I owe a great part of it to my undergraduate advisor, Stratis Gallopoulos. Faculty, staff and colleagues that affected me positively include Lenore Blum, Sharon Burks, Christos Faloutsos, Peter Lee and Dave Tolliver.

I am thankful to many friends, including Umut Acar, Nikhil Bansal, Costas Bartzis, Costas Bekas, Panos Chrysanthis, Sotiris Damouras, Morgan Designa, Christos Faloutsos, Jill de Grove, Alex Groce, Stavros Harizopoulos, Nikos Hardavellas, Dimitris Gerogiorgis, Evangelos Katsamakas, Hyang-Ah Kim, Dimitris Margaritis, Nissan 240SX, Ioanna Pagani, Elena Raptis, Kivanc Sabirli, Giorgos Sapountzis, Bianca Schroeder, Mohamed Sharaf, Sean Slattery, but especially to Costas Chrysafinos, Spiros Papadimitriou, Stratos Papadomanolakis and Spiros Tsavachidis, and my cousins Alexandros and Christina Tzatsou.

The most important people in my life are my family; my sister Eleni and my parents Andreas and Fyllio. Almost fifteen years since I left my home in Larisa, I still wish I could bend space and see them everyday.

Contents

Abstract

This dissertation presents combinatorial and algebraic tools that enable the design of the first linear work parallel iterative algorithm for solving linear systems involving Laplacian matrices of planar graphs. The major departure of this work from prior suboptimal and inherently sequential approaches is centered around: (i) the partitioning of planar graphs into fixed size pieces that share small boundaries, by means of a local "bottom-up" approach that improves the customary "top-down" approach of recursive bisection, (ii) the replacement of monolithic global preconditioners by graph approximations that are built as aggregates of miniature preconditioners.

In addition, we present extensions to the theory and analysis of Steiner tree preconditioners. We construct more general Steiner graphs that lead to natural linear time solvers for classes of graphs that are known a priori to have certain structural properties. We also present a graph-theoretic approach to classical algebraic multigrid algorithms. We show that their design can be recast as the construction of Steiner graph preconditioners. This observation makes algebraic multigrid amenable to a combinatorial approach that provides natural graph-theoretical goals and provably fast parallel algorithms for the design of the two-level scheme.

Acknowledgements

I would like to thank my advisor Gary Miller. His insights, knowledge, support and constant availability made this dissertation possible.

I also wish to thank my committee members; John Lafferty for introducing me to some great research topics; Alan Frieze for valuable discussions and for encouraging me to submit my first paper; Daniel Spielman for very helpful conversations and his feedback that helped me improve this dissertation. Dan's work kept coming up and influencing me in almost all of my seemingly unrelated research efforts.

Overcoming the difficulties that I encountered throughout the years took some self-confidence. I feel that I owe a great part of it to my undergraduate advisor, Stratis Gallopoulos. Faculty, staff and colleagues that affected me positively include Lenore Blum, Sharon Burks, Christos Faloutsos, Peter Lee and Dave Tolliver.

I am thankful to many friends, including Umut Acar, Nikhil Bansal, Costas Bartzis, Costas Bekas, Panos Chrysanthis, Sotiris Damouras, Morgan Designa, Christos Faloutsos, Jill de Grove, Alex Groce, Stavros Harizopoulos, Nikos Hardavellas, Dimitris Gerogiorgis, Evangelos Katsamakas, Hyang-Ah Kim, Dimitris Margaritis, Nissan 240SX, Ioanna Pagani, Elena Raptis, Kivanc Sabirli, Giorgos Sapountzis, Bianca Schroeder, Mohamed Sharaf, Sean Slattery, but especially to Costas Chrysafinos, Spiros Papadimitriou, Stratos Papadomanolakis and Spiros Tsavachidis, and my cousins Alexandros and Christina Tzatsou.

The most important people in my life are my family; my sister Eleni and my parents Andreas and Fyllio. Almost fifteen years since I left my home in Larisa, I still wish I could bend space and see them everyday.

Chapter 1

Overview

Solving a system of n linear equations over n variables is one of the fundamental numerical problems. The computational complexity for a general matrix of equations is $\Omega(n^2)$. The presently best known upper bound matches the complexity of matrix multiplication. Vast improvements are possible when the matrix has special properties, for example sparsity and positive definiteness. Structured matrices are quite common in scientific computing applications. Naturally, a great deal of research efforts in computational mathematics has focused on the design of efficient solvers for restricted classes of matrices.

A fairly special but important class of matrices is the class of *Laplacians* of combinatorial graphs. Graph Laplacians are intimately connected with random walks on graphs. Their eigenvalue decomposition is rich in information related to the cut structure of the graph. Not surprisingly, some of the best known algorithms for data segmentation encode the data and their relationship as a weighted affinity graph and reduce the segmentation problem to that of the computation of a small number of Laplacian eigenvectors. In turn, the computation of eigenvectors can be reduced to a small number of solutions of linear systems involving Laplacians.

Applications of Laplacians include general clustering problems [NJW01], collaborative filtering [FPS05], or the solution to systems that arise when applying the finite element method to solve elliptic partial differential equations [BHV04]. Somewhat paradoxically, the seemingly most restricted case of two and three dimensional weighted rectangular grids is probably the most important in the applied world. A prominent example are algorithms for the segmentation of medical images [Gra06], [TM06]. Every day, physicians and laboratory technicians evaluate thousands of such images. This is a task which is not only resource consuming, but often impossible for humans. For example, very slight differentiations in the scans coming from a particular person can be crucial for a medi-

cal evaluation, but may be invisible to the human eye. Consequently, the medical field increasingly relies to software for image segmentation. The images generated by current equipment give rise to graphs with close to one billion nodes. Given the amount of images that must be analyzed, this represents an enormous computational task, and a great theoretical challenge for algorithm designers; while the image segmentation algorithms produce impressive results their practicality relies on the existence of fast Laplacian solvers.

It has been known for more than 30 years that Laplacians of very structured sparse graphs that arise in the discretization of certain partial differential equations can be solved in time *linear* in the number of variables. This is striking; the system can be solved in time proportional to the time required just to read the set of equations in the memory. A particularly appealing question presents itself; is there an optimal algorithm for more general Laplacians?

This dissertation presents an optimal algorithm for the class of weighted planar Laplacians. Although several time-efficient parallel algorithms for the solution of linear systems have been described, they do asymptotically more work than the fastest sequential algorithm. In contrast, our algorithm has a work efficient parallel version. Our result is the culmination of sequence of recent advances in the construction of combinatorial preconditioners. Interestingly, as is the case with the practical importance of Laplacians, the recent advances in the design of solvers emanate from their tight connections with random walks, graph cuts, and electrical networks. In Chapter 2 we expose some basic aspects of these connections, and we review prior work.

The major departure of our work from prior approaches is a miniaturization of the preconditioner construction, based on the fact that planar graphs can be decomposed into fixed size edge-disjoint components with small boundaries. In Chapter 3 we give a linear work parallel algorithm for computing the decomposition. In contrast with previous approaches that construct the decomposition by recursively applying bisection, our algorithm works in a local fashion. In Chapter 4 we show how the decomposition enables the construction of the preconditioners that are used in the optimal solver.

In Chapter 5 we present extensions to the theory of Steiner graph preconditioners. We extend the construction and analysis of Steiner trees to more general Steiner graphs. We show that for classes of graphs that have a priori certain structural properties -including but not limited to grids with self-similarity properties- Steiner graphs lead to natural linear time algorithms. We also present a linear work parallel algorithm for decomposing a weighted planar graph into vertex-disjoint clusters, such that the subgraph induced by each cluster has high conductance and a relatively light connection to its exterior, and we discuss the existence of similar decompositions for general graphs.

We build Chapter 6 around the observation that when a pair (A, B) of positive definite matrices has a small condition number, the eigenspaces of B are expected to provide good approximations to the eigenspaces of A. We formalize this notion by developing the appropriate relative spectral perturbation theory for the pair (A, B). We show that the perturbation bounds are tight even when A and B are Laplacians. We also apply the perturbation results in the context of the Steiner support preconditioners, giving theorems that relate the structure of the eigenvectors of the normalized Laplacian of a graph with the vertex-disjoint multi-way decompositions of Chapter 5.

In Chapter 7 we show that the design of classical algebraic multigrid (AMG) algorithms for Laplacians can be recast as the construction of graph preconditioners with Steiner vertices. The analysis of the two-level scheme can thus be reduced to the analysis of the condition number for the pair of the graph A and the Schur complement B of the Steiner preconditioner. These observations makes AMG algorithms amenable to a combinatorial approach that provides natural graph-theoretical goals and provably fast parallel algorithms for the design of the two-level scheme.

Chapter 2

Background and prior work

When A is a $n \times n$ symmetric positive definite matrix, the solution to the system $Ax = b$ is unique and it can be computed *exactly*, for example via Gaussian elimination. This almost trivial mathematical statement leads immediately to an obvious algorithmic question. Given a matrix A how fast an exact or an approximate solution can be computed? Although this might at first seem as a relatively shallow question, it is in fact so interesting and so important that has motivated and sustained related research for several decades. Two broad classes of algorithms have been developed. *Direct* algorithms compute exact solutions, whereas *iterative* algorithms compute a sequence of approximate solutions that converge monotonically to the exact solution. This dissertation as well as many other fruitful approaches to the problem of solving linear systems, is based upon a combination of algebraic and combinatorial tools, for which we present the necessary background.

2.1 Linear Algebra Guide

Throughout this thesis we make use of several basic linear algebra facts. To make our presentation complete we catalogue -mostly without proofs- the most relevant and useful definitions and lemmas. We assume that the reader is familiar with undergraduate linear algebra. There are several excellent books where the reader can find the proofs and a more complete treatment, among else [SS90, Bha97, HJ85, HJ91].

Definition 2.1.1. [range and null space] *Let $A \in \mathbb{R}^{n\times k}$ be any matrix. The vector space $\mathcal{N}(A) = \{w : Aw = 0\}$ is called the* null space *of A. The vector space $\mathcal{R}(A) = \{Aw, w \in \mathbb{R}^k\}$ is called the* range *of A.*

Lemma 2.1.2. [fundamental theorem of linear algebra] *Let $A \in \mathbb{R}^{n\times k}$ be any matrix. We have $\mathcal{R}(A) = \mathcal{N}^{\perp}(A^T)$ and thus $\mathbb{R}^n = \mathcal{R}(A) + \mathcal{N}(A^T)$.*

Definition 2.1.3. [generalized eigenvalues] *Let A, B be a pair of matrices. If $Ax = \lambda Bx$, λ is an eigenvalue of the pair (A, B) with eigenvector x. We denote by $\Lambda(A, B)$ the set of eigenvalues of the pair (A, B). In the special case $B = I$, we denote by $\Lambda(A)$ the eigenvalues of A.*

Lemma 2.1.4. *If A, B^T are matrices of dimensions $n \times k$ the matrices AB and BA have the same non-zero eigenvalues.*

Lemma 2.1.5. [similarity transformation] *If X is an invertible matrix, then $\Lambda(A) = \Lambda(X^{-1}AX)$.*

A symmetric matrix A is called semi-positive definite if $x^TAx \geq 0$ for all vectors x. It is strictly positive definite when the inequality holds strictly. A symmetric matrix A is diagonally dominant (SDD) if $A_{i,i} \geq \sum_{j\neq i} |A_{i,j}|$ for all i. Every SDD matrix is semi-positive definite. The product x^TAx and the quotient x^TAx/x^TBx are often called **Rayleigh**. Very often we will be using positive definite matrices that have *common* null spaces. When this is the case we will assume that the matrices act only on their range and treat them as strictly positive definite matrices in order to simplify our notation and make the discussion more intuitive. For example, we will denote by A^{-1} the matrix B which satisfies $ABx = BAx = x$ for all $x \in \mathcal{R}(A)$.

Lemma 2.1.6. [generalized eigenvalues properties] *Let A, B be positive definite matrices. The pair (A, B) has n real eigenvalues that are positive. If $\lambda_{\min}, \lambda_{\max}$ denote the minimum and maximum generalized eigenvalues respectively, we have*

$$\begin{aligned} \lambda_{\min}(A, B) &= \min_x \frac{x^TAx}{x^TBx} \\ \lambda_{\max}(A, B) &= \max_x \frac{x^TAx}{x^TBx} \end{aligned}$$

From this we have $\lambda_{max}(A, B) = 1/\lambda_{\min}(B, A)$, and for all full invertible matrices G $\Lambda(A, B) = \Lambda(G^TAG, G^TBG)$. The eigenvalues of (A, B) are identical to the eigenvalues of $B^{-1}A$. By Lemma 2.1.4, it can be seen that $\lambda(A, B) = \lambda(B^{-1}, A^{-1})$.

A case which requires special treatment is when $\mathcal{N}(B) \subseteq \mathcal{N}(A)$. In this case all the generalized eigenvalues of (A, B) are finite and in particular

$$\lambda_{\max}(A, B) = \max_{x \in \mathcal{R}(A)} \frac{x^T A x}{x^T B x}.$$

Definition 2.1.7. [support] *The support $\sigma(A, B)$ of a matrix A by a matrix B is defined by*

$$\sigma(A, B) = \min\{t \in \mathbb{R} : x^T(\tau B - A)x \geq 0 \text{ for all } x \text{ and all } \tau \geq t\}$$

For a catalogue of properties of the support we refer the reader to [BH03].

Lemma 2.1.8. [splitting lemma] *Let $A = \sum_i Ai$ and $B = \sum_i B_i$ where A_i, B_i are positive definite matrices. Then*

$$\lambda_{\max}(A, B) \leq \max_i \lambda_{\max}(A_i, B_i).$$

Lemma 2.1.9. *If A and B are positive definite matrices and for all vectors x, $(x^T Ax)/(x^T Bx) \leq c$, then $(x^T A^r x)/(x^T B^r x) \leq c^r$, for all $r \leq 1$.*

Proof. See [Bha97], Theorem V.1.9. □

Definition 2.1.10. [spectral radius] *The spectral radius $\rho(A)$ of a matrix A with real eigenvalues is the maximum over the absolute values of its eigenvalues.*

Lemma 2.1.11. [radius sub-additivity] *For any two symmetric matrices A, B, we have $\rho(A + B) \leq \rho(A) + \rho(B)$.*

Lemma 2.1.12. [radius sub-multiplicativity] *Let A and B be symmetric matrices. If B is semi-positive definite, $\rho(BA) \leq \rho(B)\rho(A)$.*

Proof. By Lemma 2.1.4 we have $\rho(BA) = \rho(B^{1/2}AB^{1/2})$. For any unit vector x, let $y = B^{1/2}x$. By Lemma 2.1.6 we have $|y^T y| \leq \rho(B)$. We have

$$\rho(B^{1/2}AB^{1/2}) = \max_x \left|x^T B^{1/2}AB^{1/2}x\right| \leq |y^T y| \left|\frac{y^T Ay}{y^T y}\right| \leq |y^T y|\rho(A).$$

The last inequality follows again from lemma 2.1.6. □

Definition 2.1.13. [A-norm] *If A is a positive definite matrix, we define the A-inner product by*

$$(u, v)_A = u^T A v$$

the A-norm

$$\|u\|_A^2 = (u, u)_A$$

and the corresponding matrix norm

$$\|M\|_A = \max_{u \neq 0} \frac{\|Mu\|_A}{\|u\|_A}.$$

Lemma 2.1.14. [singular values] *The singular value decomposition of an arbitrary matrix A is given by its factorization $A = U^T \Sigma V$, where Σ is a diagonal matrix with positive values that are the singular values of A, and U, V are orthonormal matrices whose columns are respectively the left and right singular vectors of A. For the maximum singular value $\sigma_{\max}(A)$ of A we have*

$$\sigma_{max}(A) = \sigma_{max}(A^T) = \max_{\|x\|_2 = \|y\|_2 = 1} |x^H A y| = \max_{\|x\|_2 = 1} \|Ax\|_2 = \rho^{1/2}(AA^T).$$

2.2 Graph theory

A weighted graph $G = (V, E, w)$ on a set of n vertices V is a set of edges $E \in V \times V$ along with a positive weight function $w : e \in E \to \mathbb{R}^+$. When $w(e) = 1$ for all $e \in E$ we will say that the graph is unweighted. We define the **volume of a vertex** u as the sum of weight of the edges that are incident to e.

$$d(u) = \sum_{e \in u \times V} w(e)$$

and its degree $deg(u)$ as the number of edges incident to u. We extend the definition to the **volume of a set of vertices** A as

$$vol(A) = \sum_{u \in A} d(u).$$

We define the **capacity** $cap(x, y)$ to be equal to 0 if $(x, y) \notin E$ and equal to $w((x, y))$ otherwise. We extend the definition to pairs of sets in the natural way

$$cap(X, Y) = \sum_{x \in X, y \in Y} cap(x, y).$$

2.2.1 Edge separators

A k-way **edge separator** consists of edges whose removal partitions the vertices of the graph into k disjoint clusters. The **sparsity** of a 2-way edge cut into sets X and $V - X$ is given by the ratio

$$\phi(X) = \frac{cap(X, V - X)}{\min\{vol(X), vol(V - X)\}}.$$

The **sparsest cut** is the edge cut that achieves the minimum sparsity over all possible cuts. The sparsity of the sparsest cut in G is called the **conductance** of G and we will denote it by ϕ_G. A family of graphs is called **expander** if the conductance of each member of the family is bounded by the same constant which is independent from n. We will often abuse terminology and call a graph an expander if it is understood to what family it belongs to. It is known that a random unweighted d-regular graph is an expander with high probability [AS00]. The computation of the sparsest cut is arguably one of the most important algorithmic problems. Several heuristics have been developed, among else the widely used in practice software package METIS [KK98].

The first algorithm with provable guarantees for the sparsest cut was the spectral method which produces a cut with sparsity at most $\phi_G^{1/2}$, and -as we shall see more extensively- it is based on the computation of the second eigenvector of the normalized Laplacian [Chu97]. Spectral methods are also widely used in practice [PSL90, HL95]. The theoretical guarantees provided by the spectral algorithm cannot be improved beyond the $\phi_G^{1/2}$ bound even if the algorithm is allowed to use several higher eigenvectors [GM95, GM98]. The complexity of the spectral algorithm follows closely the complexity of solving a linear system with the Laplacian of the graph, which currently is $O(mpolylog(n))$, where m is the number of edges in the graph [ST03, ST04, EEST05].

The first polynomial time algorithm for computing a cut of sparsity within a factor independent from ϕ_G was given by Leighton and Rao [LR99]. Their algorithm finds a cut with sparsity at most $O(\phi_G \log n)$. More recently a polynomial time algorithm that finds a cut with sparsity at most $O(\phi_G \sqrt{\log n})$ was given in [ARV04]. The running time of the algorithm was improved to $\tilde{O}(n^2)$ in [AHK04]. A faster $\tilde{O}(m + \min\{n/\phi_G, n^{1.5}\})$ algorithm with an $O(\log^2 n)$ approximation guarantee was described in [KRV06].

2.2.2 Vertex separators

A k-way **vertex separator** S is a set of vertices that decomposes the edges of the graph $G = (V, E)$ into k disjoint components that communicate only through vertices of S. The boundary of a given component is defined as its intersection with S, while the rest of

the vertices are the interior of the component. Vertex separators are often treated in the literature with respect to weights assigned to vertices. In our setting we uniformly assume that vertices have unit weights, and our statements for vertex separators are independent from the weight function w of the given graph.

Now, let S be a 2-way vertex cut into the sets of edges X and $E - X$. Let $V[X]$ denote the set of vertices of G that are not in S and touch an edge in X . Without loss of generality, assume that $|V[X]| \leq |V[E - X]|$. The **size** of the cut is $|S|$, its **cut ratio** is defined as $|S|/|V[X]|$, and its **balance** as $|V[X]|/n$. We say that a 2-way separator is balanced if its balance is at least $1/4$. We say that a graph G has a **family** of $f(n)$-separators, if every subgraph H of G, has a balanced separator of size $f(|H|)$.

A considerable part of this dissertation addresses the problem of computing multi-way separators for planar graphs. A graph is called **planar** if it can be embedded on the surface of a sphere, in other words if it can be drawn on the plane without edge crossings. Research on the problem of computing a small balanced vertex separator for planar graphs goes back to the planar separator theorem of Lipton and Tarjan [LT79]. They showed that every planar graph has a balanced 2-way vertex separator of size $O(\sqrt{n})$, which can be constructed in linear time. Several generalizations for graphs of bounded genus as well as improvements in the constants have been reported, among else in [GHT84, Mil86a].

Spectral methods provably compute separators with cut ratio at most $O(1/\sqrt{n})$ for (unweighted) bounded degree planar graphs [ST96], and at most $O(\sqrt{g/n})$ for bounded degree graphs of genus g [Kel04]. The spectral algorithm does not require the computation of an embedding of the graph which is a common step for the other algorithms. This becomes very important for graphs of bounded genus whose embedding requires time with an exponential dependence on g [Moh99]. The disadvantage of the spectral algorithm is that the separators are not in general balanced.

As first observed by Frederickson [Fre87], the recursive application of the planar separator theorem reveals that a planar graph has a small n/k-way vertex separator that decomposes the graph into components of size at most k, such that every component has $O(\sqrt{k})$ boundary vertices in average. This was generalized (with the appropriate adjustments on the average boundary size) to classes of graphs that have families of small separators [KST01]. Both approaches are constructive and provided that there is an $f(n)$-time algorithm for the computation of a *balanced* 2-way separator, they yield an $O(f(n)\log(n/k))$ algorithm for the construction of the decomposition.

Parallel algorithms for the computation of balanced 2-way vertex separators for planar graphs were studied by Gazit and Miller [GM87]. They gave an $O(\log^2 n)$ time algorithm with work complexity $O(n^{1+c})$ for any fixed $c > 0$. The algorithm can be modified to

find a slightly suboptimal $O(\sqrt{n}\log n)$ separator by doing $O(n\log^2 n)$ work. The algorithm of Gazit and Miller can be used to parallelize the existing sequential algorithms, but with an extra $\log^2 n$ factor for the total work of the algorithm, and a suboptimal size for the boundaries of the components in the partition. We note that there is an $O(n)$ time algorithm for constructing a full tree of separators for a planar graph [Goo95]. However, the separators constructed in [Goo95] are subtly different from the separators needed in [Fre87] or [KST01]. More importantly, the parallel version of this algorithm requires the computation of BFS tree for the graph. Currently known parallel algorithms for the computation of a BFS tree require at least n^2 work, and their improvement is a long standing open problem.

The work in this dissertation addresses the problem of decomposing a planar graph into components of size at most k such that every component has $O(\sqrt{k})$ boundary vertices in average, for a fixed constant k. We give a linear work $O(\log n)$ time parallel algorithm.

2.2.3 Graphs, electrical networks and Laplacians

Given an arbitrary numbering of the vertices of the graph, we define the **adjacency matrix** A_G of a graph G as $A_G(i,j) = cap(i,j)$. Let D_G be the diagonal matrix containing the volumes of the vertices of G, that is $D_G(i,i) = d_i$ and $D_G(i,j) = 0$ for $i \neq j$. We define the **Laplacian of** G as the matrix $L_G = D_G - A_G$. We also define the **normalized Laplacian** as the matrix $N_G = D_G^{-1/2} L_G D_G^{-1/2}$. If $G_1 = (V, E, w_1)$ and $G_2 = (V, E, w_2)$ and $G = (V, E, w_1 + w_2)$, we have

$$L_G = L_{G_1} + L_{G_2}. \tag{2.1}$$

There is a one-to-one correspondence between Laplacians and graphs, and because of that, we will drop subscripts whenever it is possible. It can be seen that Laplacians corresponding to connected graphs are semi-positive definite, with the constant vector $\mathbf{1}$ as their common null space.

The **edge-incidence matrix** Γ is defined as the $|V| \times |E|$ matrix with rows corresponding to vertices and edges corresponding to columns. For a column k corresponding to an edge between vertices i, j we let $\Gamma(i,k) = 1$ and $\Gamma(j,k) = -1$. If D is the matrix of the volumes of the vertices of the graph then its Laplacian satisfies $L = \Gamma D \Gamma^T$. Using this, it can be seen that

$$x^T L x = \sum_{i \neq j} w(i,j)(x_i - x_j)^2$$

The algebraic approach has been indispensable to the derivation of several graph theoretical results that are covered in at least three advanced monographs [Big94, CDS98,

RG97]. In the rest of this subsection we review some of the most relevant aspects to this dissertation. Consider the lazy **random walk** on the graph, where a particle at vertex i: (i) stays in i with probability, or (ii) follows edge e with probability $w(e)/2d(i)$. The matrix whose i^{th} row contains these transition probabilities is simply $1/2(I - D^{-1}L)$. This straightforward connection has been used extensively to discover and prove properties of random walks [Lov93]. A closely related connection can be established with **electrical networks** [DS00]. A graph can be viewed as an electrical network where each edge with weight $cap(i, j)$ corresponds to a resistance $r_{i,j} = 1/cap(i, j)$. The close relationship of the two models is highlighted by the fact that the *average commute time* between vertices i, j which is the expected time for a random walk starting from i to return to i after having visited j, is equal to $2Vol(V)R(i, j)$, where $R(i, j)$ is the *effective resistance* between i and j in the corresponding electrical network [Lov93]. Then, considering a vector x as **voltages** applied to the nodes of the network, Ax is the vector of **residual flows** on the vertices. Concretely, if L_i is the i^{th} row of the Laplacian, the residual flow at vertex i is given by

$$r_i = L_i x = \sum_{j:(j,i)\in E} cap(i, j)(x_i - x_j) \tag{2.2}$$

The product $x^T Lx$ is the **power dissipation** in the electrical network for the voltages given by x.

It can be easily derived that $\lambda_{max}(L) \leq 2 \max_v d(v)$ and $\lambda_{\max}(N) \leq 2$. The constant eigenvalues of the normalized Laplacian are almost trivial from a combinatorial point of view. However the opposite side of the spectrum is rich in combinatorial information about the given graph. Fiedler observed that the positive and negative components of the second eigenvector of L_G correspond to two connected components of vertices in G [Fie73]. His work eventually led to the spectral method for the computation of a sparse cut in a graph. If x is any unit norm vector with $x^T Nx = \alpha$, then an edge cut with sparsity α can be found as follows: Let X_i be the set of the largest i entries of x. The sparsest cut among the n 2-way cuts defined by X_i, for $i = 1, \ldots, n$ has sparsity at most α. The **Cheeger inequality** [Chu97] gives

$$\lambda_2(N_G) \geq \phi_G^2/2. \tag{2.3}$$

The spectral method for computing a sparse cut computes the eigenvector x_2 corresponding to λ_2. From the Cheeger inequality it follows that the cut computed from x_2 has sparsity at most $(2\phi_G)^{1/2}$. As we shall see, in practice a 2-approximate eigenvector of the second eigenvector, that is a vector x such that $x^T Nx/x^T x \leq 2\lambda_2$, is easier to compute.

2.3 Direct linear system solvers

Consider a system of equations with the matrix

$$\begin{pmatrix} 10 & 1 & 1 & 1 & 1 \\ 1 & 5 & 0 & 0 & 0 \\ 1 & 0 & 5 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 5 & 0 \\ 1 & 0 & 0 & 0 & 5 \end{pmatrix} \tag{2.4}$$

The matrix is **sparse**, it has only $O(n)$ non-zero elements. It can be described by the list of its non-zero entries. Most of the times, a programmer who would want to code up Gaussian elimination would be inclined to implement it in its usual form: "at the i^{th} step subtract a multiple of the i^{th} row from the rows below it so that all the elements of the i^{th} column below the diagonal are zeroed-out". After just one elimination step this is how the matrix looks in terms of its non-zero structure:

$$\begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix}$$

Obviously, something went wrong; although we started with a matrix with $O(n)$ non-zero entries, we ended up with a matrix that has $O(n^2)$ non-zero entries. This is the problem of **fill**; eliminating variables, causes entries which were zero to become non-zero. However, in the above example we can do better; renaming the variables (for example switching the place of x_1 and x_5) changes the matrix of the system to:

$$\begin{pmatrix} 5 & 0 & 0 & 0 & 1 \\ 0 & 5 & 0 & 0 & 1 \\ 0 & 0 & 5 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 5 & 1 \\ 1 & 1 & 1 & 1 & 10 \end{pmatrix} \tag{2.5}$$

Now, if we apply Gaussian elimination, it can be seen that the problem of fill disappears completely. The number of non-zero entries in the matrix never exceeds $O(n)$. There-

fore, it appears that the usual Gaussian elimination algorithm is not optimal. Before its application we need to compute a good *ordering* of the variables.

2.3.1 The graph theory connection

Although along its course Gaussian elimination may cancel a non-zero entry and restore a zero entry, this will clearly be a coincidence due to the specific values of the non-zero entries in A. It is not hard to see that if we apply the algorithm to almost every matrix with the non-zero structure of A, when an entry of the matrix becomes non-zero, it will stay non-zero until the termination of the algorithm.

The non-zero structure of a symmetric matrix can be captured naturally by G_A, the **graph of the matrix** A. The graph of the matrix has n vertices and vertices i, j are joined by an edge if and only if $A(i, j) \neq 0$. For example, G_A for our example is a star with $n-1$ leaves and one center node. The definition of a graph for a given matrix is quite appealing; it suggests the idea of using graph theoretic tools in our effort to compute a good ordering for the elimination. The slight problem in this approach is that the Gaussian elimination as shown above destroys the symmetry of the matrix. Fortunately, we can work around this problem by making use of special properties of positive matrices that give rise to the *Cholesky factorization*.

2.3.2 Cholesky factorization

From an algebraic point of view, Gaussian elimination can be used to drive the factorization of A in the form $A = LDU$, where L and U are lower and upper triangular matrices with 1 in the diagonal, and D is a diagonal matrix matrix. Once the LDU decomposition is computed, the upper and lower triangular matrices can be inverted easily, and thus the solution to $Ax = b$ can be computed without too much additional work.If A is symmetric, we furthermore have $U = L^T$. When A is positive definite, the decomposition $A = LDL^T$ enjoys special properties and its very simple rewriting to $A = (LD^{1/2})(LD^{1/2})^T$ is known as Cholesky factorization. In the rest of this thesis we will call the LDL^T factorization a Cholesky factorization. A full exposition and proofs for the Cholesky factorization can be found in [GL96].

Let A be a $n \times n$ be a positive definite matrix, and let I_m denote the $m \times m$ identity

matrix. We can write

$$\begin{aligned}
A &= \begin{pmatrix} d_1 & v_1^T \\ v_1 & B_1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ v_1/d_1 & I_{n-1} \end{pmatrix} \begin{pmatrix} d_1 & 0 \\ 0 & B_1 - (v_1 v_1^T)/d_1 \end{pmatrix} \begin{pmatrix} 1 & v_1^T/d_1 \\ 0 & I_{n-1} \end{pmatrix} \\
&= L_1 A_1 L_1^T, \\
A_1 &= \begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & v_2^T \\ 0 & v_2 & B_2 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & v_2/d_2 & I_{n-2} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & B_2 - (v_2 v_2^T)/d_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & v_2^T/d_2 \\ 0 & 0 & I_{n-2} \end{pmatrix}.
\end{aligned}$$

A consequence of the fact that A is positive definite is that $d_1 > 1$ and $B_1 - (v_1 v_1^T)/d_1$ is positive definite. Therefore the process may continue recursively until we get

$$A = L_1 \dots L_{n-m} \begin{pmatrix} D & 0 \\ 0 & Q \end{pmatrix} L_{n-m}^T \dots L_1^T.$$

where Q is an $m \times m$ positive definite matrix. If $m = n$ we recover the Cholesky factorization, whereas if $m < n$ we will call the product a **partial Cholesky factorization**.

It is very instructive to review the process using the graph theoretical connection. Let $G(A) = (V, E)$. Consider the first step

$$\begin{aligned}
A &= \begin{pmatrix} d_1 & v_1^T \\ v_1 & B_1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ v_1/d_1 & I_{n-1} \end{pmatrix} \begin{pmatrix} d_1 & 0 \\ 0 & B_1 - (v_1 v_1^T)/d_1 \end{pmatrix} \begin{pmatrix} 1 & v_1^T/d_1 \\ 0 & I_{n-1} \end{pmatrix}.
\end{aligned}$$

This step can be viewed as the elimination of the first vertex v_1 from the graph of A. Let $N(v_1)$ denote the set of neighbors of v_1 in $G(A)$. The number of non-zero entries in the lower triangular matrix L_1^T is equal to $|N(v_1)|$. We now focus on $G(B) = G(B_1 - (v_1 v_1^T)/d_1)$. This is a graph on $V - v_1$. It is well known and can be verified easily that this graph consists of the edges of the subgraph of $G(A)$ induced by $V - v_1$, plus the **complete graph** on the vertices of $N(v_1)$. Therefore, from a graph theoretical point of view, the fill in the matrix is just the extra edges that are added on $V - v_1$ among the neighbors of v_1. Going back to our example in equation 2.4, the elimination of the center vertex in a

star graph creates the complete graph on the leaves of the star. On the contrary, when we eliminate a leaf as in equation 2.5, we get another star with $n-2$ leaves.

Having seen the graph theoretical interpretation of a variable elimination, we are now ready to completely abandon the algebraic language and switch to graph theoretical language. We will use interchangeably A and $G(A)$. We will view each step of the Cholesky factorization process, as a vertex elimination that simply produces a new graph and a lower triangular matrix for the factorization. To summarize our discussion so far, we state a key lemma:

Lemma 2.3.1. *Eliminating a vertex v from a graph A creates a complete graph on the neighbors of v in A. In particular, if v is a vertex of degree 1 or 2, its elimination decreases the number of edges in the graph by at least 1.*

Assume now that we have a partial Cholesky factorization

$$A = L \begin{pmatrix} D & 0 \\ 0 & B \end{pmatrix} L^T$$

. The system then can be solved as:

$$x = L^{-T} \begin{pmatrix} D^{-1} & 0 \\ 0 & B^{-1} \end{pmatrix} L^{-1} b$$

The matrices L^{-1} and L^{-T} are not formed explicitly. In practice the vectors $L^{-1}u$ and $L^{-T}u$ are computed via backward and forward substitution, with a number of operations proportional to the number of non-zero entries in L [SS90].

2.3.3 Parallel Cholesky factorization

Assume that the edges of the graph can be partitioned by a vertex separator into disjoint sets. Algebraically, this means that the matrix A can be written as $\sum_i A_i$ with the matrices A_i having common non-zero entries only along the diagonal of A. Furthermore assume that we would like to construct the Cholesky factorization with respect to the elimination of vertices only in the interior of the A_i's. By Lemma 2.3.1, the elimination process for the vertices in the interior of A_i depends only the graph induced by the edges in A_i and the elimination order in A_i. Hence for all i, the elimination of the interior vertices of A_i gives a local Schur complement B_i which can be computed "locally", as a function of A_i. Algebraically, the global Schur complement B will be $\sum_i B_i$, and we can write

$L = \prod_i L_{A_i}$, where L_{A_i} corresponds to the elimination of the nodes in A_i, and can be constructed independently from the other L_{A_i}'s. Algorithmically, the vectors $L^{-1}u$ and $L^{-T}u$ are computed via backward and forward substitution that involves only locally the variables corresponding to the vertices of each A_i. Both in the computation of the Schur complement and in the substitution we need to compute sums on the vertex boundaries, where the summands come from the neighboring clusters of edges. The sums can be computed in parallel time $O(\log n)$, and the total work is proportional to the total number of non-zero entries in L.

2.3.4 Exploiting the graph theory connection

In view of Lemma 2.3.1, it can be seen that elimination of a vertex v of degree 3 and more from A may create fill, unless the neighbors of v are already joined in A. So, before we start worrying about fill, we can at least greedily eliminated vertices of degree 1 and 2 from the starting graph $A = (V, E)$, since no extra edges are introduced into the graph. Let us formally state a slight variance of this algorithm. Let S be subset of V.

`Eliminate`(A, S): Greedily apply the following rules when possible:
(a) If $v \notin S$ has degree 1 remove v and its adjacent edge.
(b) If $v \notin S$ has degree 2 remove v and connect its neighbors with an edge.

In fact it is not hard to see that `Eliminate` works perfectly for trees, and in fact rule (a) is enough.

Lemma 2.3.2. *When the graph of a matrix A is a tree, the solution to the system $Ax = b$ can be computed in $O(n)$, by greedy elimination of vertices of degree 1.*

It is interesting to ask how many vertices we can eliminate from a given graph before we get stuck with a graph where every vertex has degree at least 3. The following folklore Lemma due probably to Vaidya [Vai91] and used in several algorithms and articles (e.g. [Che01, ST04]), provides a bound.

Lemma 2.3.3. *Algorithm `Eliminate` returns a graph C with at most $4(|S| + |E| - |V| + 1)$ nodes. In addition if A is planar then C is also planar.*

After we are left with a graph where every vertex has degree at least 3, computing a good order becomes a more difficult problem. Computing the order that produces the

minimum fill-in is an NP-complete problem [Yan81]. Even if we settle with a polylogarithmic approximation for the fill the best known algorithms for computing a good order require time that exceeds kmn, where k is the optimal fill value and m is the number of edges in the graph [NSS98]. This time bound almost always exceeds the complexity of solving the system with other known methods. In fact, the best ordering does not provide any asymptotic improvement over an arbitrary ordering for almost every sparse matrix [Duf74, LRT79].

However, the situation is different when the graph is known *a priori* to have special structural properties, as it is the case with most applications. Consider the case of a two dimensional square grid with n vertices. Eliminating vertices at distant areas of the grid causes the introduction of only local and relatively isolated extra edges. Exploiting this locality was the central idea in the pioneering work of Alan George on nested dissection [Geo73], that showed that any positive definite system whose matrix is the square grid can be solved in time $O(n^{1.5})$. The square grid shares with every planar graph with n vertices the property that it can be split into two roughly equal sized parts by removing $\sqrt{n}$ vertices. In general a class of graphs is said to have a family of n^c separators, when every graph of the class can be divided in two roughly equal sized parts by removing n^c vertices. Lipton Rose and Tarjan observed that this is the key property needed for a good ordering and extended this work to graphs that have families of small vertex separators [LRT79]. They showed that any matrix whose graph that has a family of n^c separators can be solved in time $O(n^{1+c})$, provided that the tree of separators can be computed within the same time bound. As a result, using algorithms for computing the separator trees, planar graphs and graphs of bounded genus can be solved in time $O(n^{1.5})$ [LT79, GHT84]. Improvements are possible also for several other classes of graphs [GT87], for example d-dimensional grids. Due to more recent results, the general class of d-dimensional well shaped meshes can be solved in time $O(n^{1+(d-1)/d})$ [EMT93]. The nested dissection algorithms for these classes of graphs remain the best known algorithms to date.

The availability of parallel computers and large distributed systems has motivated research on parallel algorithms for solving the linear systems, and in particular on work-efficient parallel versions of the best known sequential algorithms. Pan and Reif introduced parallel nested dissection which achieved an $O(\log^3 n)$ time complexity, with total work at most $O(n^{1+c}\log^2 n)$, provided that the algorithm is given the tree of n^c-separators for the graph.

We close this section by mentioning that the computation of good orderings has been central in numerous theoretical and applied articles (e.g. [BMMR97, BMM99]), as well as in the development of robust linear system solvers such as the frontal and multi-frontal linear solvers for systems that may be indefinite and un-symmetric (e.g. [DR83]). In

addition, graph separators have been used to reduce the communication costs in parallel implementations for sparse matrix multiplication [GGKK94].

2.3.5 General direct solvers

As noted in the previous Section, the graph theoretical connection does not yield an improvement to the asymptotical complexity of Cholesky factorization for general positive definite matrices. Conjugate gradients is widely regarded as an iterative algorithm because it uses only matrix-vector multiplications, and it computes a converging sequence of approximate solutions. However, it is also a direct solver because it recovers the exact solution to the system after n steps [Dem97]. Each step has complexity $O(m)$, where m is the number of edges of the graph of the system, hence its total complexity is $O(mn)$. This is the best known algorithm for $m < n^{1.376}$. When $m > n^{1.376}$ the best algorithm (that works for general systems of equations) uses formulas that are provided through the Coppersmith-Winograd algorithm for matrix multiplication [CW90], the last paper in a sequence of Strassen-like approaches that was initiated in the celebrated work of Strassen [Str69].

2.4 Iterative linear system solvers

Iterative algorithms for the solution of linear systems are procedures that generate a sequence of approximate solutions x_t and corresponding errors $e_t = A^{-1}b - x_t$. We say that an iterative method converges if $\lim_{t\to 0} \|e_t\| = 0$. Typically, iterative algorithms target very large sparse matrices where the cost of direct methods is prohibitive, both in terms of time and space. Although it is not always clear whether iterative methods can reduce the time complexity, they at least can address the space complexity which is a very important problem because typically large memory usage translates to heavier use of very slow types of memory. Of course, the m non-zero entries of A provides obviously a minimal requirement for the time complexity. Iterative algorithms keep the space requirement low by keeping in the memory a small number of vectors and strive for fast convergence by using only matrix-vector multiplications with A, and vector additions. Although this looks like a rather small repertoire of available operations, it leads -in some instances at least- to asymptotically nearly optimal or optimal time complexity.

2.4.1 Richardson's iteration

Suppose we pick an arbitrary initial approximation x_0 to the solution of the system $Ax = b$. For the discussion in this subsection we need only assume that A is an arbitrary full-rank matrix with possibly complex eigenvalues. We would like to update x_0 with a better approximation x_1 using only computationally inexpensive (with respect to the input size) operations: vector additions and one matrix-vector multiplication with A. At the very least we must preserve x_0 if we were extremely lucky to start with $x_0 = A^{-1}b$. Perhaps the simplest iteration with these properties is known as **Richardson's iteration**:

$$x_t = (I - A)x_{t-1} + b. \tag{2.6}$$

Observe that the solution x satisfies $x = x_{t-1} + e_{t-1}$. Of course e_{t-1} is not available. What is readily available is the residual at time $t - 1$ which is defined as $r_{t-1} = b - Ax_{t-1}$. The residual can be seen as an easy to compute "approximate" form of error. A different derivation of Richardson's iteration is based on a **residual correction** approach. Form the new approximation as the sum combination of the current approximation and the residual:

$$x_t = x_{t-1} + r_{t-1} = x_{t-1} + b - Ax_{t-1} = (I - A)x_{t-1} + b.$$

So, this leads to another derivation of equation 2.6. The only stationary point of the iteration is the solution of the system. But does it always compute a better approximation x_t ? For this we need to express the error $e_t = x_t - A^{-1}b$ after the first iteration, in terms of the error $e_{t-1} = x_{t-1} - A^{-1}b$ in the beginning. A simple algebraic manipulation shows that $e_t = (I - A)e_{t-1}$. This implies that if we start with an initial error e_0 and apply the same iteration t times we get

$$e_t = (I - A)^t e_0. \tag{2.7}$$

To analyze the behavior of the error we will use the spectral decomposition of A. Let λ_i, for $i = 1, \ldots, n$ be the eigenvalues of A, with $|\lambda_i| \leq |\lambda_{i+1}|$, and $Ax_i = \lambda_i x_i$, where the vectors x_i are normalized. We have

$$\begin{aligned} e_0 &= \sum_{i=1} a_i x_i \Rightarrow \\ e_t &= \sum_{i=1} (1 - \lambda_i)^t a_i x_i \end{aligned} \tag{2.8}$$

Clearly if for all i we have $|1 - \lambda_i| < 1$, then all the coefficients in the expression of e_t converge to 0 as t increases, and we say that Richardson's iteration *converges*. If $|\lambda_i| > 1$ for some i, the method diverges. However, provided that we have an upper bound $c|\lambda_n|$ for $|\lambda_n|$, we can change the system to $Bx = |c\lambda_n|^{-1}b$, where $B = |c\lambda_n|^{-1}A$. Then, all

the eigenvalues of the new matrix B have magnitude less than 1 and Richardson's method converges.

How fast does Richardson's iteration converge? Let us formalize the question. Having fixed A, we define the norm $n_A : n_A(x) = \sum_i a_i^2$ when $x = \sum_i a_i x_i$. Clearly, the speed of convergence is determined by the eigenvalue of B of smallest magnitude, which is equal to $|\lambda_1|/|c\lambda_n|$. The number $\kappa(A) = |\lambda_n|/|\lambda_1|$ is known as the **spectral condition number** of A. It can be seen that even when λ_n is known exactly, $t = \kappa(A)\ln(1/\epsilon)$ iterations are needed so that $n_A(e_t) \leq \epsilon n_A(e_0)$.

2.4.2 Multigrid algorithms

Let us focus again on positive definite matrices, and more specifically on normalized Laplacians. Let E be a 3-regular unweighted expander with n vertices. Assume that we want to solve the system $0.5N_E x = b$. Formally, N_E has a null space - the constant vector. However we can restrict all our vectors orthogonal to the constant vector, and view N_E as a positive definite matrix with smallest eigenvalue equal to $\lambda_2(N_E)$. The maximum eigenvalue of N_E is 1, and by the Cheeger inequality, $\lambda_2(N_E)$ is a constant, independent of the size of the graph. Hence a constant number of iterations are enough to halve the error. This is impressive given that expanders are exactly the kind of graphs that are tough for Cholesky factorization for any ordering of the variables. On the other hand it is not hard to come up with a bad example. Let A_n be the normalized Laplacian of the cycle graph on n vertices. In this case we have $\lambda_2(A_n) = 1/n^2$, and Richardson's iteration requires $O(n^2)$ iterations which translate to an $O(n^3)$ complexity before the error gets reduced by a factor of 2. Observe that A_n can be solved in linear time by the Cholesky factorization, with respect to any ordering of the variables.

However we should not yet abandon Richardson's method. After all, we know that its application does reduce the error corresponding to constant eigenvalues - the **high frequency** of A_n. The idea then is to use a different algorithm for the elimination of the low frequency error. This idea is the main principle behind *multigrid* algorithms. We give a short introduction to the basic notions of multilevel methods. For a more thorough introductory exposition we refer the reader to the excellent tutorial by Briggs et. al. [BHM00].

Graph theoretically, very small sets of neighboring vertices in A_n are expanders and thus after some applications of the Richardson's iteration the error will not differ by much among neighboring vertices. Hopefully then the reduction of the high frequency error can be viewed as a local *smoothing* of the error. Let x_t be the approximate solution after t Richardson's iterations. Now consider the residual $r = b - A_n x_t$. The solution of the

system is equal to $A_n^{-1}b = x_t + A_n^{-1}r$. The observation that iteration 2.6 smoothes the error locally, leads to the idea of replacing A_n^{-1} by the "coarse" graph $A_{n/2}^{-1}$, and forming a new approximate solution as follows:

0. Let $r = b - Ax_t$;
1. Form a projection $r' = R_{project}^T(r)$, where $r' \in \mathbb{R}^{n/2}$;
2. Find $y' = A_{n/2}^{-1}r'$.
3. Lift y' to $y = R_{projeect}(y')$ where $y \in \mathbb{R}^n$.
4. Return $x_{t+1} = x_t + y$.

The hope is that the exact inversion of $A_{n/2}$ will reduce sufficiently the part of the error not dealt with by smoothing. In case $e_{t+1} = x_{t+1} - A_n^{-1}b$ contains high frequency error, the situation can be rectified easily by a few more steps of *post-smoothing*.

Without going into the details here let us not that one of the most elementary aspects of the multigrid analysis is the matrix that describes the error reduction associated with this correction step:

$$M = I - R_{project}A_{n/2}^{-1}R_{project}^T \tag{2.9}$$

We present a derivation of this matrix in Chapter 7.

Up to this point we have a *two-level* algorithm since we use only A_n and $A_{n/2}$. Of course, the exact computation of $A_{n/2}^{-1}r'$ is itself a difficult task. The natural solution is recursion; instead of solving exactly for $A_{n/2}$, apply the same algorithm to it. The definition of the multigrid algorithm is then the following:

$MG(A_n, b, x_0)$
1. Do t steps of $x_j = (I - A_n)x_{j-1} + b$;
2. Form a projection $r' = R_{project}^T(b - Ax_t)$, where $r' \in \mathbb{R}^{n/2}$;
3. Let $y_j = MG(A_{n/2}, r', y_{j-1})$;
4. Lift $y' = y_1$ or y_2 to $y = R_{project}(y')$ where $y \in \mathbb{R}^n$;
5. $x_{t+1} := x_t + y$.
6. Do t steps of $x_j = (I - A_n)x_{j-1} + b$

The structure of the recursive calls of MG resembles a "V" and the algorithm is also known as the V-cycle. Historically, multigrid methods were developed to deal with matrices corresponding to underlying differential operators, whose discretizations give natural hierarchies of 'grids' with certain repeated properties, or 'regularities'. Hence the name multigrid.

The first paper on multigrid was written in 1964 by Fedorenko [Fed64]. Then in 1977, Brandt wrote a seminal paper that popularized multigrid and made it practical [Bra77].

In the late 70s Hackbusch and Nicolaides gave the first proofs of optimal convergence for certain PDEs (e.g [Hac78, Nic78]). From then on, the field of multigrid exploded, resulting in hundreds of experimental and theoretical papers. Currently there is a vast literature on multigrid, including more than 3500 related references, and about 25 free software packages. The Copper Mountain Conferences on Multigrid Methods have been held biennially since 1983. For a more complete picture we refer to the several available books [Wes04, Bra93, TSO00, Sha03]. Ultimately, all the proofs of convergence that have appeared in the literature rely heavily upon the elliptic geometry of the underlying differential operators that allow the construction of self-similar grids, and the appropriate choice of the projection operator and the smoothing iteration.

Very often the classical multigrid approach is referred to as *Geometric multigrid* to make a distinction with *Algebraic multigrid* (AMG) which was introduced as an effort to generalize the principles of multigrid to general weighted graphs for which no geometric information/discretization is given a priori [BMR84]. While in geometric multigrid the two-level scheme is explicitly suggested by the choices in the discretization of the differential operators, the corresponding problem is a major problem in AMG. At a high level, the usual AMG approach consists of: (i) the choice of a subset of the variables that form the second level graph often called the "coarse" grid (ii) the assignment of each "fine" grid point to a small number of coarse grid points, (iii) the choice of interpolation/projection operators that transform vectors in the coarse space to vectors in the fine space, and vice-versa [Bra86, BHM00]. In general, the algorithms for performing these steps are mostly based in heuristics, with no guarantees on the running time and the size of the second level graph. Although the algorithm is quite successful in practice for SDD matrices arising in applications with a markedly scientific computing/discretization flavor, there is little theory and its convergence properties are not well understood [CFH$^+$00]. In particular, there are absolutely no guarantees for the complexity and convergence of the V-cycle for the Laplacian of an arbitrarily weighted square grid on the plane.

In Chapter 7 we show that the design of AMG algorithms for Laplacians can be recast as the construction of graph preconditioners with Steiner vertices. This observation makes AMG algorithms amenable to a combinatorial approach that provides natural graph-theoretical goals and solutions for the design of the two-level scheme. The analysis of the two-level scheme can in turn be reduced to the analysis of the condition number for the pair of the graph A and the Schur complement B of the Steiner preconditioner. We show that for Steiner preconditioners that are constructed from *edge separators*, $\kappa(A, B)$ is not a sufficiently strong property to guarantee the convergence of the multigrid V-cycle, precisely because of the tightness of the perturbation bounds of Chapter 6. We introduce a stronger notion of graph approximation, the condition number $\kappa(\hat{A}^2, \hat{B}^2)$, where $\hat{A}, \hat{B}$ are

normalized versions of A, B, and we show that it guarantees convergence of the V-cycle. Furthermore, driven by this new graph approximation measure, we propose Steiner preconditioners that are based on vertex separators on a properly modified linear system, and we give linear work parallel algorithms for their construction in the planar case.

2.4.3 Basic iterative methods

There are several iterative methods [Axe94]. In this subsection we list only the asymptotic convergence rates of methods that specialize to positive definite matrices. We state the convergence properties in term of the A-norm (see equation 2.1.13). The **steepest descent** algorithm requires $t = \kappa(A)\ln(1/\epsilon)$ iterations so that $\|e_t\|_A \leq \epsilon \|e_0\|_A$. It does not require the knowledge of an upper bound for $\lambda_{\max}(A)$. The **Conjugate Gradients (CG)** algorithm requires $t = \sqrt{\kappa(A)}\ln(2/\epsilon)$ so that $\|e_t\|_A \leq \epsilon \|e_0\|_A$. CG can be much faster when the eigenvalues of A fall in a small number of very tight clusters. In fact the worst case complexity of CG is derived by upper-bounding it with that of the **Chebyshev iteration**. Chebyshev iteration requires bounds that localize the eigenvalues of A whereas CG does not.

2.4.4 Preconditioning

As we saw in subsection 2.4.1 a simple multiplication by a scalar is enough to change the spectrum of the matrix so that Richardson's iteration converges. Of course multiplication by a scalar has just a scaling effect to the eigenvalues of the matrix. Multiplication by matrices can alter completely its spectrum and make it more favorable for the application of some iterative method. This is the idea of preconditioning; transforming the system $Ax = b$ to

$$B^{-1}Ax = B^{-1}b \tag{2.10}$$

where B is the **preconditioner**. Given that A is positive definite, the new matrix $B^{-1}A$ won't be in general symmetric, and this may be potentially a problem for the application of CG and the Chebyshev method. Fortunately, when B is positive definite, a little algebraic manipulation can transform these algorithms so that they implicitly operate on $B^{-1/2}AB^{-1/2}$, only with matrix-vector multiplications with A and B^{-1}. For the details we refer to [Axe94]. The convergence behavior of the new system in the A-norm is then determined by the **condition number** of the pair (A, B), defined as

$$\kappa(A, B) = \lambda_{\max}(A, B)\lambda_{\max}(B, A)$$

.

In our discussion in this dissertation we will be using the preconditioned Chebyshev method for analysis purposes. Following [ST06], we will view preconditioned Chebyshev as a function with the following specifications:

$$\boxed{\text{x = }\textbf{PrecondChebyshev}(A, b, f_B(\cdot), \tilde{\lambda}_{\min}(A, B), \tilde{\lambda}_{max}(A, B), t)}$$

where $f_B(z) = B^{-1}z$, $\tilde{\lambda}_{\min}(A, B)$, $\tilde{\lambda}_{max}(A, B)$ are approximations to the corresponding eigenvalues of (A, B) and t is the number of iterations. From our discussion so far, we get that when $t = \kappa^{1/2}(A, B)\ln(2/\epsilon)$ the error satisfies $\|e_t\|_A \leq \epsilon \|e_0\|_A$.

Obviously the complexity of the algorithm depends on the definition of B. For example, if $B = A$ the algorithm obviously converges in one step but the computation of $B^{-1}z$ is just our original problem. Thus the design of the preconditioner should strive to satisfy two contradicting goals: (i) The condition number $\kappa(A, B)$ must be small (ii) The matrix B must have a relatively inexpensive partial Cholesky factorization.

In contrast to the direct methods where the sparsity pattern of A can always be used to derive a good elimination order, the construction of a good preconditioner is an issue that in general depends subtly on the given matrix. Several preconditioners that depend on the matrix in straightforward generic ways have been proposed. For example:

1. $B = D$ where D is the Laplacian of A, gives the **Jacobi method**. Letting B contain blocks along the diagonal of A gives the more general block Jacobi algorithm.

2. $B = D+L$ where L is the lower triangular part of A gives the **Gauss-Seidel** method, used only with iterations that don't require the preconditioner to be symmetric.

3. $B = (D + L)D^{-1}(D + L^T)$ is an instance of SSOR also known as **symmetric successive overelaxation**.

Although these preconditioners may work very well for certain matrices, they give no general guarantees. As an example let A be the Laplacian of the wagon-wheel graph consisting of a star and a cycle on n nodes. It can be verified that $\kappa(A) = \Theta(n)$. On the other hand, the eigenvalues of $D^{-1}A$ are those of the normalized Laplacian. The wagon-wheel is an expander hence the eigenvalues of $D^{-1}A$ are constant so $\kappa(A, D) = O(1)$. On the contrary Jacobi's method for the Laplacian of the 2-dimensional square grid does not yield any improvement since the smallest eigenvalue of the normalized Laplacian is $O(1/n)$, asymptotically equal to the smallest eigenvalue of the Laplacian.

2.4.5 Combinatorial Preconditioners for SDD matrices

Perhaps the first systematic approach to the construction of preconditioners for a fairly general class of matrices is due to Vaidya [Vai91, Che01]. Vaidya, inspired by the one-to-one correspondence of Laplacians and graphs, proposed preconditioning the Laplacian of a given graph with the Laplacian of a spanning subgraph. If A, B are Laplacians and D is a positive diagonal matrix an easy application of the splitting Lemma 2.1.8 shows that $\kappa(A + D, B + D) \leq \kappa(A, B)$. Hence Vaidya's approach applies to the more general class of symmetric diagonally dominant matrices with negative entries. Gremban showed that the solution of a system with a SDD matrix with positive off-diagonal entries can be reduced to the solution of a SDD system with only twice the size as the original and with non-positive off-diagonal entries [Gre96]. Hence Vaidya's preconditioners apply to the general class of SDD matrices.

Initially, Vaidya showed that taking the preconditioner B to be the **maximum weight spanning tree (MST)** gives $\kappa(A, B) \leq nm$, where m is the number of edges in the graph. This was far from trivial, because it showed that the preconditioning of Laplacians is possible *independently from the graph weights*. He then proposed an algorithm for adding edges to the tree and he proved that it yields an $O(n^{1.75})$ time algorithm for any bounded-degree weighte graphs and a $O(n^{1.2})$ algorithm for weighted planar graphs. Joshi [Jos97] and Reif [Rei98] observed that in the partial Cholesky factorization

$$B = L \begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix} L^T$$

where D is a diagonal, the matrix C is a Laplacian if B is a Laplacian. In other words, *the class of Laplacians is closed under elimination of vertices*. In particular, adjusting the greedy elimination of degree 1 and 2 for Laplacians, gives the following algorithm:

$B = \texttt{Eliminate}(\texttt{A}, \texttt{S} \subseteq \texttt{V})$: Greedily apply the following rules when possible: (a) If $w \notin S$ has degree 1 remove w and its adjacent edge from the graph A. (b) If $w \notin S$ has degree 2 and is connected to vertices u and v, remove w and connect its neighbors with an edge of weight $(w^{-1}(u, w) + w^{-1}(v, w))^{-1}$.

In view of Lemma 2.3.3 adding a sublinear number vertices to the spanning tree still gives a graph B that has many degree 1 and 2 vertices. After their elimination the graph C will be smaller than B, but it might still be quite big for a direct method. However because of the fact that C is a graph it is possible to use recursion. Joshi [Jos97] analyzed recursive algorithms for simple model problems while Reif analyzed a constant depth recursive

algorithm and improved the bound for constant degree planar graphs to $O(n^{1+\beta})$ for any $\beta > 0$ [Rei98] .

The theory behind the application of Vaidya's approach to matrices with non-positive off diagonals is presented in [BGH$^+$06]. An algebraic extension of Vaidya's techniques was given by Boman and Hendrickson [BH03], and based on this extension they observed that the *low-stretch* spanning trees of Alon, Karp, Peleg and West [AKPW95] has condition number at most $O(m + n2^{O(\sqrt{\log n \log\log n})})$. This reduced the complexity of the solver to $O(m^{1.5+o(1)})$. Spielman and Teng [ST03] demonstrated a way of carefully adding edges to the low-stretch spanning trees. This yielded a method that requires $O(m^{1.31})$ time. Then, by (i) improving the algorithm for adding edges, (ii) giving an $O(mpolylog(n))$ *sparsification* algorithm and (iii) presenting a careful analysis of the recursive preconditioned Chebyshev method with a super-constant number of levels, they showed that general graphs can be solved in time $O(m2^{O(\sqrt{\log n \log\log n})})$ [ST04, ST06]. Finally, by replacing the trees of [AKPW95] with lower-stretch spanning trees, Elkin, Emek, Spielman and Teng improved the performance of the algorithm to $O(n \log^2 \log\log n)$ for planar graphs and to $O(mpolylog(n))$ for general graphs [EEST05].

In a separate thread of work, Gremban and Miller [Gre96] considered a different kind of graph-based preconditioner. They introduced additional vertices called *Steiner vertices* and they demonstrated that a graph preconditioner need not be of the same size as the graph represented by A. Gremban and Miller presented and analyzed *support tree preconditioners* for regular d-dimensional unweighted grids. Their tree B for the d-dimensional grid satisfies $\kappa(A, B) = O(dn^{1/d} \log n)$. Miller and Richter showed that the condition number of any spanning subgraph of the square grid is $\Omega(n^{1-e})$ for all $e > 0$, thus proving the superiority of the Steiner tree preconditioners for this graph [MR04]. Maggs et. al. developed new tools for analyzing general support trees [MMP$^+$05]. They also showed that Räcke's hierarchical decomposition of graphs [RÖ2, BKR03] gives support trees that guarantee an $O(n \log^4 n)$ condition number. In this dissertation we present additions to the theory of Steiner support preconditioners, that extend the analysis of [MMP$^+$05] to more general Steiner support graphs that are derived from the Steiner trees, and show that Steiner preconditioners also yield linear time algorithms when used in recursive solvers, for families of graphs that are known a priori to have certain structural properties.

2.4.6 Support theory - The role of the Splitting Lemma

The progress in the analysis as well as in the design of combinatorial preconditioners has been built around the fact that Laplacians are *closed under addition*. The idea is to *split* the graph A and the preconditioner B into smaller graphs $A = \sum_i A_i$ and $B = \sum_i B_i$, so

that the support $\sigma_{\max}(A_i, B_i)$ (see definition 2.1.7) : (i) is easy to analyze, (ii) has a good bound. Then the bound for $\lambda_{\max}(A, B)$ follows from the Splitting Lemma (Lemma 2.1.8).

Historically, the construction of subgraph preconditioners via the computation of a tree and its subsequent enrichment with a few edges was driven by the fact that the splitting is analyzable and actually dictated by the tree. Let us be more concrete; assuming that B is a spanning tree of a given graph A, the difficult part is to analyze $\lambda_{\max}(A, B)$ because we trivially have $x^T Ax > x^T Bx$. In the splitting, the graphs A_i are the edges of A and B_i must be the unique path in B that goes between the endpoints of A_i, because any other subgraph of B has infinite support with A_i. In this simple case the support $\sigma(A_i, B_i)$ turns out to be easy to analyze. It can be verified that it is equal to the ratio of the resistance of A_i over the effective resistance between the endpoints of B_i. So, if B_i is a relatively long path, it must consist of heavy (relative to A_i) edges in order for $\sigma(A_i, B_i)$ to have a good bound. This led Vaidya to propose the MST preconditioner, and to the eventual replacement of the MST by a low-stretch tree.

While the low-stretch trees are still indispensable to the nearly linear algorithm of Spielman and Teng for general graphs, the fixation with the construction of a monolithic global preconditioner has been the main barrier to obtaining optimal parallel algorithms at least for the class of planar graphs. The novel idea in our work is to bypass the construction of the global low stretch tree for the given graph, by exploiting the combinatorial structure of the underlying unweighted graph. As we discussed in Section 3 every planar graph has small vertex separators that partitions the graph into constant size components. We compute the partition and then, a proper "miniature" preconditioner is constructed independently for each of these pieces. The global preconditioner will be the aggregation of the miniature preconditioners. Its quality is bounded above via the Support Lemma, by the quality of the worst among the miniature preconditioners. We give the details in Chapter 4.

Chapter 3

Planar Graph Partitioning

It is known that multi-way planar vertex separators with small boundaries can be constructed in $O(n \log n)$ time [Fre87, KST01]. This upper bound is sufficient in applications where the construction of the separator is not the dominant complexity term. However, in the case of the solution of planar Laplacians, the developments presented in Chapter 4 show the possibility of an optimal linear time algorithm, provided that the multiway separator can be constructed in linear time. In this Chapter we resolve this question, by presenting a linear work, $O(\log n)$ parallel time algorithm for the computation of the multiway separator. We present the algorithm in the Concurrent Read Exclusive Write - Parallel Random Access Memory (CREW-PRAM) model. In this model, processors are allowed to read but not to write simultaneously the same memory address [Pap94]. The algorithm adapts and improves an algorithm of Gazit and Miller [GM87].

Let $A = (V, E)$ be a graph, and W be a vertex separator that decomposes the edges into disjoint sets $E = \bigcup_{i=1}^{m} E_i$. Let $A_i = (V_i, E_i)$ be the graph induced on E_i, and let $W_i = W \cap V_i$. The total boundary cost is defined as $\sum_{i=1}^{m} |W_i|$. This Chapter presents a proof for the following Theorem.

Theorem 3.0.1. *Every planar graph with n nodes has a vertex separator W with total boundary cost $O(n/\sqrt{k})$, that decomposes the edge set into disjoint clusters of size $O(k)$. The separator can be constructed in the CREW PRAM model with $O(nk \log^2 k)$ work in $O(k \log n)$ parallel time, or in $O(kn)$ sequential time.*

Our algorithm computes a set S of $O(n/\sqrt{k})$ edges, that -on the planar embedding of the graph- can be thought as boundaries delimiting components of size $O(k)$. Each edge of S is the boundary of at most two such components. By assigning each edge of S

arbitrarily to one of its neighboring components we get a decomposition of the edges E into disjoint components of size $O(k)$. The vertices incident to S will be the separator W. Then, each node of S is incident to a number of components equal to the number of edges incident to it in S. Hence, the total cost of the boundary will be $O(n/\sqrt{k})$. The algorithm is based on an algorithm of Gazit and Miller [GM87]. It runs in $O(k \log n)$ parallel time, doing at most $O(nk \log^2 k)$ work.

Throughout this section we let $\bar{G}$ be a triangulation of G. Given the embedding of G, the triangulation can be computed easily with linear work in $O(\log n)$ time. Thus every edge in $\bar{G}$ is either an edge in G or an added edge. The separator will be the boundary between a partition of the faces of $\bar{G}$, consisting of $O(n/\sqrt{k})$ edges.

There are two natural graphs to define on the set of faces $\bar{F}$ of $\bar{G}$. The first is where we connect two faces if they share an edge, the **geometric dual**, denoted by $\bar{G}^*$. In the second, the **face intersection graph**, we connect two faces if they share a vertex. Note that the face intersection graph is not in general planar, while the dual is planar. We say that a set of faces in $\bar{F}$ are **edge/vertex** connected if the corresponding induced graph in the geometric dual/face intersection graph is connected.

3.1 Neighborhoods and their cores

We define the **vertex distance** $dist(f, f')$ between two faces f and f' to be one less than the minimum number of faces on a vertex connected path from f to f'. Since the faces are triangular, $dist(f, f')$ is equal to the length of the shortest path from a vertex of f to a vertex of f', plus one. Thus two distinct faces that share a vertex are at vertex distance one. A d-radius vertex connected ball centered at a face $f \in \bar{F}$, denote $B_d(f)$, is the set of all faces at distance at most d from f. That is, $B_d(f) = \{f' \in \bar{F} \mid dist(f, f') \leq d\}$. By induction on the radius of the ball, one can show that a ball forms a set of edge connected faces. We are now ready to give the definition of a k-neighborhood, and some of its consequences.

Definition 3.1.1. *The k-**neighborhood** of a face $f \in \bar{F}$ $N_k(f)$ will consist of k faces defined as follows:*

1. *The ball $B_d(f)$ where d is the maximum d such $|B_d(f)| \leq k$.*
2. *The faces at distance $d + 1$ from f are picked so that they form an edge connected set of faces, and $N_k(f)$ remains edge connected and of size k.*

We call faces at a given distance from f a **layer** and those at distance $d+1$ the **partial layer**. We define $d+1$ to be the **radius** of $N_k(f)$. By definition, the boundary of the last full layer, is a simple cycle. Since the partial layer is edge connected to the last full layer, the boundary of $N_k(f)$ is also a simple cycle.

For each face we construct its k-neighborhood. The neighborhood of a face f that is incident to a node v of degree at least k, will have only a partial layer. The partial layer can be constructed by taking the first k edges going in a clockwise fashion around v. In order to simplify our presentation, if a face is incident to more than one nodes of degree more than k, we will construct one k-neighborhood per each such node, as described above. So, a given face may generate up to three neighborhoods.

Lemma 3.1.2. *The number of neighborhoods containing any given face is* $O(k^{\log k+2})$.

Proof. We seek to bound the size of the set $\mathcal{C}$ of faces whose neighborhoods contain a given face f'. The neighborhoods are edge connected. If $f' \in N$, there is an edge connected path of faces from f' to the center of N. There are at most $6k$ neighborhoods of radius $r = 1$ that may contain f'. Every neighborhood of radius $r \geq 2$ that contains f' includes in its full layers at least one of $18k$ given faces that surround f'. So, from now on, we may assume that the neighborhoods are full balls.

We claim that $\mathcal{C}$ is an edge connected set of faces. To see why, let $f \in \mathcal{C}$, with $N(f) = B_r(f)$. Let h be the edge-incident face on the path from f to f'. We must have $f' \in B_{r-1}(h)$. Let $I(f)$ be the set of faces at distance 1 from f. We have $B_r(f) = \bigcup_{g \in I(f)} B_{r-1}(g)$. Since $h \in I(f)$, this implies that the radius of $N(h)$ is at least $r-1$. Hence $f' \in N(h)$, and $h \in \mathcal{C}$.

We will find a set $\mathcal{B}$ of $(2k)^{\log k+1}$ neighborhoods that cover all the faces in $\mathcal{C}$. To form $\mathcal{B}$ we will be removing, in rounds, sets of neighborhoods from $\mathcal{C}$. We start with $N(f') = \mathcal{B}_0$. Assume that in the t^{th} round we removed a set $\mathcal{B}_t$. We will let $\mathcal{B}_{t+1}$, be the neighborhoods of the faces that have not been covered in previous rounds, and are edge-incident to the faces in $\mathcal{B}_t$. Hence $|\mathcal{B}_{t+1}| \leq 2k|\mathcal{B}_t|$. Let r_t be the minimum radius over the neighborhoods in B_t. To go from $f \in \mathcal{B}_t$ to f' the path must go through r_{t-1} layers of a neighborhood N in B_{t-1}, before it reaches the center of N. By an inductive argument, this gives that $r_t \geq \sum_{i=0}^{t-1} r_i \geq 2^{t-1}$. This implies that after $d \leq \log k + 1$ rounds, the process must stop because r_d becomes greater than k, meaning that all neighborhoods in $\mathcal{B}_d$ have radius greater than k, which is the maximum possible by definition. So, $|\mathcal{C}| \leq 3\sum_{i=1}^{d} |\mathcal{B}_t| = O(k^{\log k+2})$. □

The critical fact is that each k-neighborhood $N_k(f)$ has a set C_f of **core** faces.

Lemma 3.1.3. *Let $N_k(f)$ be a neighborhood of radius r. There exists a ball, $B = B_{r'}(f)$ such that $2(r - r') + |\partial B| \leq \sqrt{2k} + 4$. We call $B_{r'}(f)$ the* **core** *of $N_k(f)$.*

Proof. The proof follows by a standard pigeon hole argument used by Lipton and Tarjan [LT79]. We give the proof for completeness. Let b_i be the size of the boundary of ball $B_i(f)$, and f_i be the number of faces in the i^{th} level of $N_k(f)$. We will discard the partial layer in $N_k(f)$ and one full level, and we will show that the core lies inside $B_{r-2}(f)$. More precisely, we will show that $2((r-2) - r') + |\partial B| \leq 2\sqrt{k} + 1$. Since all the layers are full we can apply Lemma 5 in [Mil86b]. Thus there exists a r' such that

$$2((r-2) - r') + |\partial B| \leq 2\sqrt{b} \tag{3.1}$$

where $b = \sum_{i=0}^{r-2} b_i$. Using the fact that $\bar{G}$ is triangulated we know that $b_0 = 3$, $f_0 = 1$, and $b_i + b_{i-1} = f_i$ for $0 < i < r$. Thus:

$$\begin{aligned} k - 1 &\geq f_0 + \cdots + f_{r-1} \\ &= f_0 + b_0 + 2b_1 + \cdots + 2b_{r-2} + b_{r-1} \\ &= 1 - 3 + b_{r-1} + 2b_0 + 2b_1 + \cdots + 2b_{r-2} \\ &\geq 2b - 1 \end{aligned} \tag{3.2}$$

Here we used the fact that $b_r \geq 1$. Substituting above, we get

$$2(r - r') + |\partial B| \leq 2\sqrt{b} + 4 \leq \sqrt{2k} + 4 \tag{3.3}$$

□

Lemma 3.1.4. *If $N_k(f_1)$ and $N_k(f_2)$ have at least one vertex in common and P is any shortest path in $\bar{G}$ from the boundary of f_1 to the boundary of f_2, then the exposed part of P, that is the number of edges exterior to $C_{f_1} \cup C_{f_2}$ is at most $\sqrt{2k} + 4$.*

3.2 An outline of the algorithm

With the introduction of the neighborhoods and their cores, we are ready to restate our goal for the rest of this section. We aim to find a set $\mathcal{P}$ of $O(n/k)$ paths or **incisions**, with the following properties: (i) the removal of $\mathcal{P}$ disconnects the graph into pieces of size $O(k)$, (ii) the two endpoints of each incision $P \in \mathcal{P}$ are faces whose neighborhoods touch, so that Lemma 3.1.4 applies to P. Then, for every incision P with end faces f_1, f_2,

we will include in the final separator $\mathcal{S}$: (i) the boundaries of the cores C_{f_1} and C_{f_2}, and (ii) the exposed part of P. One way to think of this, is that we first find the incisions, and then we add the cores of their end points on top of them. Finally, we return to the graph the interior of all the cores. It then becomes clear that the final separator decomposes the graph into pieces of size $O(k)$. Furthermore, by Lemma 3.1.3 the number of edges added in $\mathcal{S}$ per incision, is at most $2(\sqrt{2k}+4)$. Hence, the total number of edges in the final separator is $O(n/\sqrt{k})$.

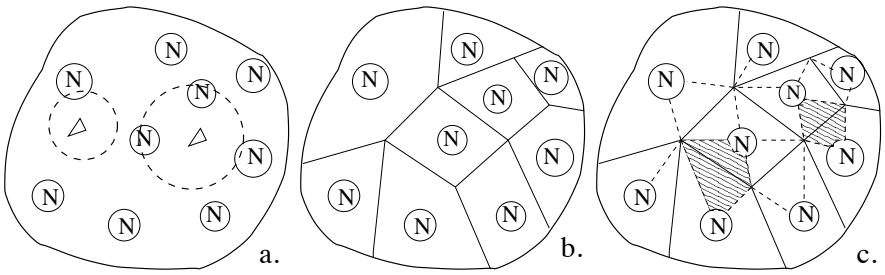

Figure 3.1: Steps of the algorithm.

We now give a short outline of the algorithm. The first step is to obtain maximal a set I of at most n/k face-disjoint neighborhoods in $\bar{G}$. We will call this the set of **independent neighborhoods**. The maximality of the set of independent neighborhoods will provide a good "covering" of the graph, in the sense that the neighborhood of every face **exterior** to I, intersects at least one neighborhood in I. This step is shown schematically in Figure 3.1a and it is described formally in section 3.3. In the second step, we assign each exterior face to one of the neighborhoods in I, in order to decompose the graph into edge-connected **Voronoi regions** of faces, each consisting of the faces assigned to one neighborhood. This step is depicted in Figure 3.1b and described in Section 3.4. The edges between the Voronoi regions form a planar graph that will be called the **Voronoi boundary graph**. The nodes in the Voronoi boundary graph with degree greater than 2 will be called **Voronoi nodes**. The next step will be to further decompose the graph into **Voronoi-Pair regions**, by finding paths between the neighborhoods and the surrounding Voronoi nodes. Two of the Voronoi-Pair regions are highlighted in Figure 3.1c. We give the details in Section 3.5. Finally, we separately split each Voronoi-Pair region, as described in Section 3.6.

3.3 Computing the set of independent neighborhoods

We say that two neighborhoods are **independent** if they share no faces of $\bar{F}$. Our goal will be to compute a maximal set I of independent neighborhoods. It is easy to compute I in $O(kn)$ sequential time. The purpose of this section is to show that I can be computed in $O(k \log n)$ parallel time, doing $O(nk \log^2 k)$ work in the CREW PRAM model.

For the rest of this section let us denote with $|G|$ the number of edges of a graph G. We define the containment graph B_0 to be the bipartite graph with the left side nodes corresponding to neighborhoods, and the right side nodes corresponding to faces. Any given neighborhood is joined with the k faces it contains. By construction, $|B_0| \leq 3kn$. We also define the neighborhood conflict graph $N(B_0)$, by letting nodes correspond to neighborhoods, and edges joining neighborhoods that intersect. By Lemma 3.1.2, every neighborhood intersects with most $O(k^{\log k})$ neighborhoods. Thus $|N(B_0)| = O(k^{\log k} n)$.

We will use a modification of Luby's algorithm [Lub86]. Let us first briefly describe the algorithm. Assume that the input graph has n nodes. The algorithm consists of a number of rounds. The algorithm maintains an (initially empty) independent set I of nodes, which have been removed from the graph along with their neighbors. In every round: (i) Each node in the graph independently picks a random number in $(1, n^4)$. (ii) If a node has a bigger number than its neighbors, it joins I. (iii) The nodes that join the I and their neighbors remove themselves from the graph. The process continues until the graph is empty. Luby showed that with high probability one round of the algorithm reduces the number of edges in the graph by a constant fraction. Thus the algorithm terminates with a Maximum Independent Set (MIS) after $O(\log n)$ rounds.

We now describe a simulation of the t^{th} round of Luby's algorithm. Recall that we start with the set of neighborhoods that have not been removed from the graph, and their faces. We can as above define the current corresponding containment graph B_t and the current neighborhood conflict graph $N(B_t)$. The simulation will be done in k steps. At any step: (i) Each neighborhood is active or inactive. Initially, each neighborhood is active. If a neighborhood becomes inactive, it stays inactive for the rest of the round. (ii) Each face is owned by one neighborhood. Initially, one of the (up to) three neighborhoods that have a given center face owns it. The other two become inactive. (iii) Each face f keeps the value v_f of the neighborhood that owns it, and each neighborhood keeps a list L_N of the faces that it has owned during the previous steps.

Each neighborhood N pics a random number v_N in $(1, n^4)$, and computes a breadth first search (BFS) spanning tree of its geometric dual. The tree will be connected since

the neighborhood is edge connected. At each step, each active neighborhood N picks one face f' from L_N, that is edge connected with a face $f \notin L_N$, and $f \in N$. This is always possible since every neighborhood is edge connected, and it can be done in constant time using the BFS for N. Then N checks if it still owns f'. If not, N becomes inactive. If yes, N sends to f the value v_N as a request to own f, and adds f to L_N. Note that at any point there will be only three neighborhoods that request from f to own it, so this step can be performed in constant time, without a concurrent write. So, f receives at most 3 neighborhood values, compares them with v_f and keeps the largest, which becomes the new v_f. After the k steps are completed, every active neighborhood N reads the values of its faces, and if any face has value bigger than v_N, N becomes inactive. We are left with a set of still active neighborhoods each of which owns all its faces. Then, every one of these neighborhoods joins the I and marks its faces for deletion. All these writes are independent. Then, every neighborhood reads the values from its faces, and if it finds a face that has been marked for deletion, it removes itself from the containment graph B_t, and so it doesn't take part in the subsequent rounds of the algorithm. It is easy to see that the total work of the round is $O(|B_t|)$, and the parallel time complexity is $O(k)$.

The correctness of the algorithm follows from the fact that all the neighborhoods that remain active until the end are independent by construction. A neighborhood can become inactive only if it intersects a neighborhood with higher index. So, if a neighborhood has the biggest index among all the neighborhoods it intersects, it will stay active, and will join I. Thus the algorithm adds to I a superset of the neighborhoods that Luby's algorithm would add if run on $N_i(B)$. So with high probability we get a constant reduction $1/c$ of the number of edges of $N(B_t)$. Recall that $|N(B_0)| = O(k^{\log k} n)$, so for a proper $d = O(\log^2 k)$, $|N_0(B)|/c^{dk} = O(n)$. Also, it is clear that $|B_t| < |N(B_t)|$, and $|B_0| \leq 3kn$. Hence, the total work is

$$\sum_{t=0} |B_t| = \sum_{t=0}^{dk} |B_t| + \sum_{t=dk+1} |B_t| \leq \sum_{t=1}^{dk} |B_0| + \sum_{t=dk+1} |N(B_0)|/c^{dk} c^{t-dk} = O(k \log^2 kn).$$

3.4 Decomposition into Voronoi Regions

The goal of this section is to decompose the graph into edge connected Voronoi regions, each corresponding to one of the neighborhoods in I. At a high level, the natural approach is to find the nearest neighborhood of each exterior face f, and assign f to it. However, an exterior face may have several nearest independent neighborhoods. Simply breaking ties does not guarantee the edge connectedness of the Voronoi regions. We shall instead decompose faces that have more than one nearest neighborhood into more triangular faces,

and then assign these new faces to neighborhoods.

Let f be an exterior face. Let ∂N denote the faces on the boundary of a neighborhood N. We define $\textit{dist}(f, N) = \min_{a \in \partial N} dist(f, a)$, and $\textit{dist}(f) = \min_{N \in I} \textit{dist}(f, N)$.

Lemma 3.4.1. *Let f be an exterior face of radius r. Then $r \geq dist(f)$. Also, if $N(a) \in I$ is such that $dist(f, N(a)) = dist(f)$ then $N(a)$ and $N(f)$ share at least one vertex. Finally, if v is any vertex of f, every path that starts at v and has length at most $dist(f) - 1$, is contained in $N(f)$.*

Proof. Since f is not in I, there must be a face f with $N(a) \in I$, such that $N(f) \cap N(a) \neq \emptyset$. Let f' be face in the intersection. Then $\textit{dist}(f) \leq \textit{dist}(f, f') \leq r$. If $dist(f, N(a)) = l \leq r - 1$, then $N(a)$ must contain a face in the l^{th} layer of $N(f)$ and this face is included in $N(f)$ by definition. If $dist(f, N(a)) = r$ and $N(f) \cap N(a) = \emptyset$, then $N(a)$ must contain a face which touches a face in the $(r-1)^{th}$ layer of f. For the last claim, note that $dist(f) - 1 \leq r - 1$. So, all the faces that share edges with the given path are at distance at most $r - 1$ from f. Hence they are included in $N(f)$, by definition. □

We now describe the algorithm. In what follows, every exterior face f will compute a labeling of each of its vertices, of the form $d[a]$, where d will be a distance, and a the index of a neighborhood in I. The labeling will be local, and so no concurrent writes are needed.

1. Recall that $\bar{G}$ is embedded and thus every vertex knows a clockwise arrangement of its edges. Given a root vertex v and an incident face f of reference, the leftmost path between v and any vertex w is well defined. One can easily compute a "leftmost" BFS tree, that provides the leftmost shortest paths starting from v. For each neighborhood $N(f)$, and every vertex v on the boundary of f, we compute the unique leftmost and rightmost BFS trees, rooted on v, with respect to f.

2. Each neighborhood $N(a) \in I$ marks all its faces with the index of a.

3. If a vertex v is on the boundary of some $N \in I$, it marks itself with 0 and submits clockwise the marks to its unmarked surrounding faces, so that the faces that receive the same mark are contiguous. This can be done in $O(\log n)$ time with $O(n)$ total work. In this way, every exterior face f receives up to 3 marks through its vertices. If f receives a through vertex v, it labels v with $0[a]$. Finally if f has received at least one mark, it labels with 1 each vertex that has not been marked with a 0.

4. By Lemma 3.4.1, to find the nearest neighborhood of an exterior face f, it is enough to consider the nodes in $N(f)$ that are marked with 0. First, we label each vertex v of f with the distance of the 0 vertex nearest to v, plus one. This is by definition equal to $dist(f)$. Let us call the vertices labelled with $dist(f)$ *critical* for f. For each critical vertex v of f, we find the **preferred** path P, defined as the leftmost path that (i) starts in v, (ii) reaches a vertex w in a neighborhood $N \in I$, (iii) has length $dist(f) - 1$. Lemma 3.4.1 implies that P is contained in $N(f)$, and thus it can be found in $O(k)$ time, by using the BFS computed in the first step. The face that lies anticlockwise (with respect to w) of the last edge of P has already labelled v with $0[a]$, for some a. Then, f labels v with $dist(f)[a]$.

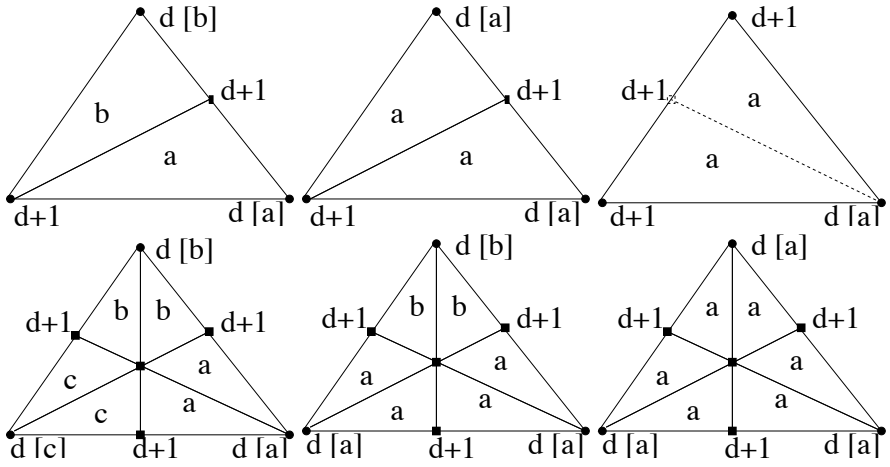

Figure 3.2: Breaking exterior faces.

5. Note that the distance labels computed for three vertices by the same face can differ by at most 1. Then, one can verify that the exterior vertices can be classified to six different cases with respect to the type of labels that they have computed for their vertices. These cases are shown in Figure 3.2. The **base case** is when the exterior face has only one critical vertex. In each other case we introduce extra nodes and edges (shown as boxes in Figure 3.2), so that every new face becomes a base case, and is marked with the corresponding nearest neighborhood. After splitting all the non-base faces (generating base faces in $\bar{G}'$), we split the base faces of $\bar{G}$ so that $\bar{G}'$ triangulated. This can be done without concurrent writes by having the faces communicate through the edges. We end up with a graph $\bar{G}'$, where every exterior face is triangular and has only one critical vertex.

All the faces assigned to a given neighborhood in $N(a) \in I$ will be called the **Voronoi Region** of a. We claim that the above construction produces Voronoi regions that are edge connected. Before we proceed to prove this claim, we need to prove the following key lemma.

Lemma 3.4.2. *All the faces that share a vertex v compute the same distance label for v.*

Proof. Suppose f_1 and f_2 share a vertex v that is critical for both f_1 and f_2. Then, f_1 and f_2 label v with $dist(f_1)$ and $dist(f_2)$ respectively. We need to show that $dist(f_1) = dist(f_2)$. Assume without loss of generality, that $dist(f_1) > dist(f_2)$. Then, there is a neighborhood $N \in I$ such that $d(f_1, N) = dist(f_2)$, which gives $dist(f_1) \leq d(f_1, N) = dist(f_2) < dist(f_1)$, a contradiction.Consider now the case when v is critical for f_1 but not for f_2. Then, f_1 labels v with $dist(f_1)$ and f_2 with $dist(f_2) + 1$. We need to show that $dist(f_1) = dist(f_2) + 1$. Note that $dist(f_1) - 1 \leq dist(f_2) \leq dist(f_1)$. So, assume for the sake of contradiction that $dist(f_2) = dist(f_1)$. We know that there is a path P of length $dist(f_1) - 1 = dist(f_2) - 1$ from v to a vertex marked with 0. By Lemma 3.4.1, P is included in $N(f_2)$, and thus f_2 must have labeled v with $dist(f_2)$, a contradiction. The remaining case is when two faces f_1 and f_2 touch on a vertex v that is not critical for neither face. Then f_1 and f_2 label v with $dist(f_1) + 1$ and $dist(f_2) + 1$ respectively. We need to show that $dist(f_1) = dist(f_2)$. Without loss of generality, let us assume that $dist(f_2) \geq dist(f_1)$. Then $dist(f_1) \leq dist(f_2) \leq dist(f_1) + 1$. There is a path P of length $dist(f_1)$ from v to a node marked with 0. If $dist(f_2) = dist(f_1) + 1$, Lemma 3.4.1 implies that P is contained in $N(f_2)$. Hence, f_2 must compute the label $dist(f_2)$ for v, a contradiction. Thus $dist(f_2) = dist(f_1)$. □

The above lemma ensures that the last step of the algorithm is always possible. To see why, observe that every non-base case face splits into triangular faces. The graph will be triangulated if and only if an edge is split by both the faces that share it. A face splits its edges that join vertices with equal labels. So, two adjacent faces have both subdivided their common edge in Step 5, unless one of them is a base case face, which can be subdivided in the last step.

Lemma 3.4.3. *The Voronoi regions are edge connected.*

Proof. By construction each neighborhood is edge connected. So, it will suffice to show that for every exterior face $f' \in \bar{G}'$ that belongs to the Voronoi region associated with $N(a)$, there is an edge connected path from f' to a face of $N(a)$. Let v be the critical

vertex of f', and f be the parent face of f' in $\bar{G}$. It must be the case that v was labeled with $dist(f)[a]$ by f in Step 4.

If $dist(f) = 1$, then v is on the boundary of $N(a)$. Step 2 ensures that there is an edge connected sequence of exterior faces surrounding v, that all marked v with $1[a]$. The face on the one end of the sequence shares an edge with $N(a)$. By the way we split the faces of $\bar{G}$, all the faces of $\bar{G}'$ that are generated inside the faces in the sequence, are labeled with a. This provides the edge connected path from f' to $N(a)$.

Now assume $dist(f) > 1$. Let P the preferred path. By construction, the face g on the left of the last edge of P has marked w with $0[a]$. Now assume that v is not in the last edge of P, and let v_1 be the vertex after v in P. We will consider the face $f_1 \in \bar{G}$ on the left of P that includes the edge (v, v_1), and the faces of $\bar{G}$ between f and f_1 that touch v, as shown in Figure 3.3. We show that these faces label v and v_1 with $dist(f)[a]$ and $dist(f_1)[a]$ respectively.

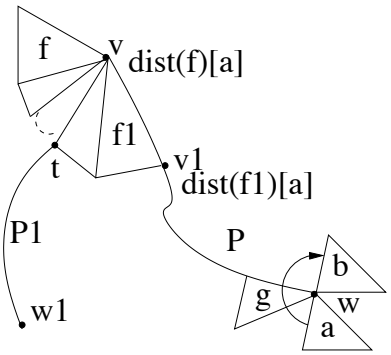

Figure 3.3: Getting one step closer to $N(a)$.

Recall that Lemma 3.4.2 shows that the distance labels are independent of the faces. We first show that the labels of all vertices on the arc between f and f_1 must be at least equal to $dist(f)$. Assume for the sake of contradiction that one of these vertices, say t, is labeled with $dist(f) - 1$. This means that there is a path P_1 of length $dist(f) - 2$ from t to a vertex marked with 0. The path $(v, t) + P_1$ has length $dist(f) - 1$. Thus P is not the preferred path. This is a contradiction. We know already that v is critical for f. Since all the nodes of the faces between f and f_1, excluding f_1, are labeled with at least $dist(f)$, v is critical for them as well. Therefore, each of these faces uses independently exactly the same definition to compute the label of v in Step 4, and so the label is consistently $[a]$.

It is easy to see that $dist(f_1) = dist(f)-1$. Since all the other vertices of f_1 are labeled with $dist(f)$, v_1 must be labeled with $dist(f_1)$. The neighborhood label computed for v_1

by f_1 is computed by considering the last edge of the leftmost path of length $dist(f) - 2$ starting from v_1. It is clear that this path is the segment of P after v_1, and thus the label is $[a]$.

By applying this argument inductively, it follows that the set of all the faces F on the left of P, mark the vertices of P with $[a]$. Finally consider all the faces of $\bar{G}'$ that were generated by splitting the faces of F. First note that, by the way we split the faces of $\bar{G}$, these faces form an edge connected path from $f' \in \bar{G}'$ to the face $g' \in \bar{G}'$ that was generated inside g. Since $dist(g') = 1$, we know that there is an edge connect path from it to $N(a)$. The concatenation of the two paths forms an edge connected path from f' to $N(a)$. □

Lemma 3.4.4. *The set of preferred paths that reach a given $N \in I$ can be used to form a BFS spanning tree of the Voronoi region of N. We call this the* **preferred BFS** *tree of the Voronoi region. Every node can find its ancestor and children in the tree in $O(\log n)$ time with $O(n)$ work.*

Proof. The proof of Lemma 3.4.3 implies that if a vertex v is critical for a face f, all the vertices on the preferred path P that goes from v back to the boundary of $N(a)$, are critical for the faces that share vertices with P. Furthermore, by construction, P cannot be crossed by any other preferred path from a critical vertex to $N(a)$. Thus the paths from the critical vertices back to the boundary of $N(a)$, together with a BFS tree of N form a BFS tree. All the faces have at least one critical vertex, and every vertex that has not been marked as critical by any face, can attach itself to the tree through one of its incident faces. □

Lemma 3.4.5. *Each Voronoi region contains $O(k^{\log k})$ faces.*

Proof. If f is a face that was assigned to a given Voronoi region V_a, $N(f)$ must intersect at least one face $b \in B$, where B is the set of faces that are incident to the boundary of $N(a)$. By Lemma 3.1.2, it is enough to bound the size of B. The faces of B contained in $N(a)$ are less than k, and so we concentrate on the faces on the exterior of $N(a)$. There are up to k faces that share an edge with the boundary of $N(a)$. Now consider all the faces which are incident only to a single vertex v on the boundary of $N(a)$. There may exist a set $M \subseteq I$, that contains faces incident to v. By construction, the faces incident to v that are assigned to $N(a)$ are exactly those between $N(a)$ and the first neighborhood $N' \in M$, walking in a clockwise fashion around v. By assumption, those faces are not contained in any neighborhood in I. Their number can't be more than k, because by construction,

there is a k-neighborhood N'' that includes k of them. The assumption implies that N'' does not intersect any neighborhood in I, which contradicts the maximality of I. Using the fact that there are at most k vertices on the boundary of $N(a)$, this implies that there are at most $O(k^2)$ faces in B. □

3.5 Decomposition into Voronoi-Pair Regions

To simplify our notation, we will be denoting $\bar{G}'$ by $\bar{G}$. We have decomposed the graph into at most n/k Voronoi regions. Their boundaries are edges of $\bar{G}$. Despite the fact that these regions are edge-connected sets of faces, their boundaries may be not connected. In general, every connected region can be decomposed into a collection of simple *boundary cycles*, where the faces exterior to one cycle are edge-disjoint to those of another cycle. See [Mil86b] for a more complete discussion. Let $\mathcal{C}$ denote the set of boundary cycles of all the Voronoi regions. Any pair of boundary cycles in $\mathcal{C}$, corresponding to different Voronoi regions, can share a path, a single vertex, or no vertices at all. We say that a cycle in $\mathcal{C}$ is *non-trivial* if it shares a path with at least one other cycle in $\mathcal{C}$. The vertices where non-trivial cycles intersect have degree at least 3. We call these vertices the **Voronoi nodes**. Thinking of the simple paths between the Voronoi nodes as edges, we get a planar graph which we call the **Voronoi boundary graph**, denoted by G_I. The graph G_I will not be in general connected when the regions have disconnected boundaries. We can think of G_I as a set of connected components, where each but one connected component lies inside one face of another connected component. To see this formally, pick an arbitrary "outer" face f_o of $\bar{G}$. To simplify our discussion we assume without loss of generality that the boundary of the region that contains f_o is connected. Every region V_g has a unique *external* boundary cycle that lies closer to f_0. The faces enclosed by the boundary of each non-trivial internal cycle boundary of V_g form a connected component of $\bar{G}$. This boundary is the outer face of a connected component G_c of G_I. Each of the other faces of G_c correspond to the external boundary cycle of exactly one Voronoi region. It can be seen that the number of faces of G_I is equal to the number of Voronoi regions that have a non-trivial external boundary.

A topological picture of a Voronoi region with a disconnected boundary is shown in Figure 3.4. Searching faces out from f, the boundary of V_f is initially connected, until it reaches a saddle point, where it disconnects into two or more connected simple cycles. There are paths from f to the saddle points that form a collection of simple cycles and decompose V_f into Voronoi subregions with simple cycle boundaries. Consider any given subregion V_{f_A}. Any point on the boundary of V_{f_A} can be reached via a shortest path from

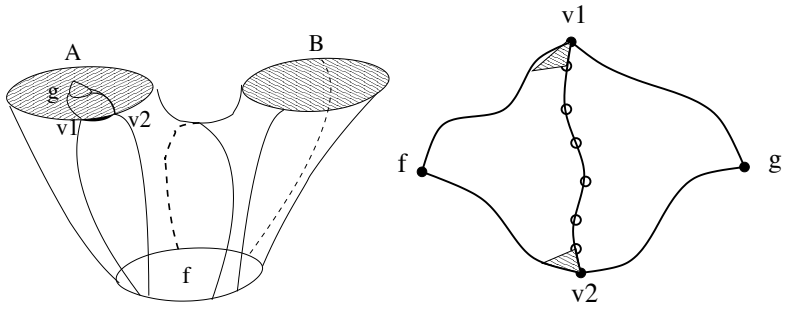

Figure 3.4: A Voronoi region and a Voronoi-Pair region.

f, that lies in V_{f_A}. Provided that we are given $k \geq 3$ vertices on the boundary of V_{f_A}, we can decompose V_{f_A} into k regions. The boundary of each of these smaller regions consists of one path on the boundary of V_{f_A}, and two shortest paths from its endpoints back to f. So, any segment along the boundary between two different Voronoi regions V_f, V_g, is reachable from both regions through shortest paths that lie inside the two subregions of V_f, V_g that share the given cycle, as depicted in Figure 3.4. This forms what we call a **Voronoi-Pair region**.

Based on the above discussion we construct the set $\mathcal{P}$ of incisions and the final separator $\mathcal{S}$, as described in Section 3.2. First, for each Voronoi region V_f we add shortest paths from f to the saddle points. This decomposes V_f into connected components with simple boundaries. Then, we pick three arbitrary vertices on every trivial cycle in $\mathcal{C}$. Let V_1 be the set of those vertices, and V_2 be the Voronoi nodes. Finally, for each Voronoi region V_f we add to $\mathcal{P}$ the shortest paths from f to each point of its boundary which is in $V_1 \cup V_2$. There are at least two such points on each boundary cycle, and each Voronoi subregion is decomposed into half-Voronoi pairs. Those are coupled with half-Voronoi pairs inside the adjacent region V_g, and thus the graph is decomposed into Voronoi-Pair regions.

Lemma 3.5.1. *The number of paths added to $\mathcal{P}$ is at most $6n/k$.*

Proof. Let α be the number of trivial external boundary cycles, and β be the number of non-trivial external cycles. We have $\alpha + \beta \leq n/k$. Let f, v, e, p be the number faces, vertices, edges, and connected components of G_I. We have $\beta = f$. The number of paths to the saddle points is at most $2p + 2\alpha$. Fix a connect component G_c of G_I. Let $f_{i,c}$ be the sizes of the faces of G_I. The total number of paths in $\mathcal{P}$ that are incident to G_c is $\sum_i f_{i,c} = 2e_c$. The number of paths to the points in V_1 is at most 3α. Hence, $|\mathcal{P}| \leq 5\alpha + 2p + 2\sum_c e_c = 5\alpha + 2p + 2e$. From Euler's formula, we have $\beta = 1 + p + e - v$.

Since $6v \leq 4e$, we have $6\beta = 6 + 6p + 6e - 6v \geq 6 + 6p + 2e > 2p + 2e$. So, $|\mathcal{P}| \leq 5\alpha + 6\beta \leq 6n/k$. $\square$

At the end of the previous section, every edge knows on which boundary it lies, and can compute its neighboring edges on it. Then, every boundary component between two Voronoi regions can decide if it is a trivial cycle or a segment with two endpoints. If it is a trivial cycle, it arbitrarily breaks itself in three segments. For the computation of the shortest paths between say, f and v_1, v_2 in Figure 3.4, we will use the preferred BFS tree of the Voronoi region of f. By construction, Lemma 3.1.4 applies to those paths. So, each path in $\mathcal{P}$, and the corresponding exposed part in $\mathcal{S}$ can be computed easily, and will be marked. It is not hard to see that all the required computations can be done in $O(\log n)$ time with $O(n)$ work.

3.6 Splitting a Voronoi Pair

Let $\mathcal{V}$ denote the set of Voronoi-Pair regions. By Lemma 3.4.5, the size of each $V \in \mathcal{V}$ is bounded by $O(k^{\log k})$. We can run Frederickson's algorithm [Fre87] on the geometric dual of each V, to add to the separator $O(|V|)/\sqrt{k}$ edges that disconnect V into pieces of size $O(k)$. The total number of edges added to $\mathcal{S}$ will be $\sum_{V \in \mathcal{V}} O(|V|)/\sqrt{k} = O(n/\sqrt{k})$. The total work will be $\sum_{V \in \mathcal{V}} O(|V| \log |V|) \leq n \log^2 k$. The algorithm can be run independently on each V, so the parallel time is $O(k^{\log k})$.

Alternatively, we can decompose the Voronoi pairs without invoking another separator algorithm. We give a sketch of the algorithm. Let V_f and V_g be the two Voronoi regions in the pair, and T_f, T_g be their preferred BFS trees. Given a segment between two vertices w_1, w_2 of the boundary, we define the weight of $[w_1, w_2]$ to be the total number of the nodes contained between the paths from w_1, w_2 to their common ancestors, in T_f and T_g respectively. We will decompose the boundary into non-overlapping segments, such that: (i) every segment consisting of one edge has weight larger than $2k$, (ii) every segment of weight less than k lies between two segments of weight larger than k, (iii) all other segments have weight between k and $2k$. Let V_3 be the set of the endpoints of these segments. We add to $\mathcal{P}$ the shortest paths from the vertices in V_3 to f and g. Since the diameter of the trees is $O(k)$, this decomposition can be done in $O(k + \log n)$ time with linear work. The total number of paths added to $\mathcal{P}$ is $O(n/k)$, by construction. We are left with the segments consisting of only one edge, whose weight can be up to $O(k^{\log k})$. Let M be the component defined by one such segment. We separately focus on each half of M. As implied by the proof of Lemma 3.4.3, along with the preferred BFS T_M, we have implicitly computed a preferred spanning tree T_M^* of the geometric dual of M. The paths

of faces in T_M^* lie along paths of T_M, by construction. We will use parallel tree contraction, to find the k-critical nodes of T_M^* in $O(k)$ time, with $O(|T_M^*|)$ work (see [RMMM93] for definitions and details). The number of critical nodes is $O(|M|/k)$. We will add to $\mathcal{S}$ the faces corresponding to the critical nodes. This will decompose M into $O(|M|/k)$ pieces (called in [RMMM93] the k-bridges) of size at most $O(k)$. The vertices contained in each of these bridges are delimited by three paths in T_M. We will add these paths to $\mathcal{P}$. The total number of paths added to $\mathcal{P}$ in this step is $O(n/k)$ and the total work is $O(kn)$.

Chapter 4

Planar Preconditioner and Solver

This Chapter presents a linear work parallel iterative algorithm for solving linear systems involving Laplacians of planar graphs. Concretely, se show the following Theorem.

Theorem 4.0.1. *If $Ax = b$, where A is the Laplacian of any weighted planar graph with n vertices, there is an algorithm that produces a vector $\bar{x}$ such that $||x - \bar{x}||_A \leq \epsilon \|x\|_A$, in $O(n^{1/6+c} \log(1/\epsilon))$ parallel time, doing $O(n \log(1/\epsilon))$ work, where c is any positive constant.*

The result also applies to symmetric diagonally dominant matrices of planar structure via the reduction of Gremban [Gre96]. The best previously known algorithm has complexity $O(n \log^2 \log \log n)$ and it is not known how to parallelize it [EEST05]. One of the key ingredients of the solver is the algorithm for partitioning the planar graph into pieces of size at most k. In order to achieve the sublinear complexity, the parallel solver must operate on pieces with size bigger than those needed in the sequential version. Because of the complexity associated with the partitioning algorithm, there is a factor of $O(k)$ loss in the total work of the parallel algorithm over that of the sequential algorithm, where k is the smallest piece size for which the sequential algorithm achieves its guarantee. We give the details in Section 4.2.2.

4.1 The solver

Given the the system $Ax = b$, our goal is to compute a vector $\bar{x}$ such that the error $\bar{x} - x$ satisfies $\|\bar{x} - A^{-1}b\|_A \leq \epsilon \|A^{-1}b\|_A$. Formally, we describe an algorithm with the following template.

$\boxed{\bar{x} = \textbf{Solve}(A, b, \epsilon, \mathcal{H}(A))}$

The solver uses an hierarchy $\mathcal{H}(A) = \{A_i, B_i\}$, $i = 1, \ldots, r$ of graphs and their preconditioners. A_i and B_i are two graphs with the same number of vertices and $A_{i+1} = \texttt{Eliminate}(B_i, S_i)$ where S_i is a (potentially empty) set of vertices in B_i. We let

$$B_i = L_i \begin{pmatrix} D_i & 0 \\ 0 & A_{i+1} \end{pmatrix} L_i^T$$

be the partial Cholesky factorization of B_i. Also, we let Π_i denote the $dim(A_{i+1}) \times dim(A_i)$ matrix for which for all x, $\Pi_i x$ consists of the last $dim(A_{i+1})$ coordinates of x. Similarly let Φ_i denote the $(dim(A_i) - dim(A_{i+1})) \times dim(A_i)$ matrix for which for all x, $\Phi_i x$ consists of the first $dim(A_i) - dim(A_{i+1})$ coordinates of x.

Without loss of generality we will assume that for all i we have $\lambda_{\min}(A_i, B_i) = 1$. Let $\kappa = \max_i \kappa(A_i, B_i)$. We define the *hierarchy condition* as $\tau_{\mathcal{H}} = 5\sqrt{\kappa} \ln \kappa$, where $\kappa = \max_i \kappa(A_i, B_i)$ and the *size reduction factor* of $\mathcal{H}$ as $\mu_{\mathcal{H}} = \max_i |A_i|/|A_{i+1}|$.

4.1.1 Two-level preconditioned Chebyshev

Following our discussion in section 2.4.4, the preconditioned Chebyshev iteration for A_1 with preconditioner B_1 gives the following definition for **Solve**, which satisfies the error reduction requirement.

$\boxed{\begin{array}{l} \textbf{Solve}(A, b, \epsilon, \{A_1, B_1\}): \\ \text{Return } \bar{x} = \textbf{PrecondChebyshev}(A_1, b, f_{B_1}(\cdot), 1, \kappa, \sqrt{\kappa} \ln(2\epsilon^{-1})) \end{array}}$

where $f_{B_1}(z) = B_1^{-1} z$. The partial Cholesky factorization of B_1 can be exploited to compute $B_1^{-1} z$ as follows:

$$\begin{aligned} f_{B_1}(z) &= L_1^{-T} \begin{pmatrix} D_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} L_1^{-1} z \\ &= L_1^{-T}((\Phi_1^T D_1^{-1} \Phi_1 + \Pi_1^T A_2^{-1} \Pi_1) L_1^{-1} z. \end{aligned}$$

4.1.2 Recursive Preconditioned Chebyshev

Computing $A_2^{-1} z$ in the one-shot preconditioned Chebyshev algorithm is an expensive operation. This leads to the idea of computing an approximate solution to the system

$A_2 w = z$. The natural idea is to use recursively the Preconditioned Chebyshev iteration, in combination with the hierarchy $\mathcal{H} = \{A_i, B_i\}$. Of course the fact that the solution is now approximate, has an impact on the number of iterations required for convergence. However the impact is not severe; in [ST04] it was shown that the following recursive algorithm obtains the required approximation:

Solve$(A_i, b, \epsilon, \mathcal{H})$:
If $i = r$ then return $A_r^{-1} b$ else
return $\bar{x}$ = **PrecondChebyshev**$(A_i, b, f_{B_i}(\cdot), 1, \kappa, 5\sqrt{\kappa} \ln \kappa \ln(2\epsilon^{-1}))$

where

$$f_{B_i}(z) \quad = \quad L_i^{-T} \Phi^T D_1^{-1} \Phi L_i^{-1} z + L_i^{-T} \Pi_i^T \textbf{Solve}(A_{i+1}, \Pi_i L_i^{-1} z, 2/e)).$$

4.1.3 The complexity of the solver

Theorem 4.1.1. *Let $\mathcal{H}(A)$ be a hierarchy of graphs with condition $\tau_{\mathcal{H}}$ and size reduction factor $\mu_{\mathcal{H}}$. If the hierarchy satisfies*

$$\boxed{\tau_{\mathcal{H}}/\mu_H = (hierarchy\ condition)/(size\ reduction\ factor) < 1/2}$$

then the complexity of **Solve**(A, b, ϵ) *is $O(\tau_{\mathcal{H}} |A| \ln \epsilon^{-1})$.*

Proof. Let $\mathcal{H} = \{A_i, B_i\}$ for $i = 1, \ldots, r$. We will let $|A_r|$ be a constant, so that the corresponding systems are solved in constant time. By an easy induction, the total number of calls to **Solve** with input A_i for $i > 1$, is τ^i. For each call of **Solve** at level i, the amount of work is $O(\tau_{\mathcal{H}} |A_i|) = O(\tau_{\mathcal{H}} |A|/m^i)$. The total amount of work is $O(\tau_{\mathcal{H}} |A| \sum_i (\tau_{\mathcal{H}}/\mu_{\mathcal{H}})^i) = O(\tau_{\mathcal{H}} |A|)$. The proof is completed by noting that the number of calls to **Solve** with input A_1 is $\tau_{\mathcal{H}} \ln \epsilon^{-1}$. □

4.2 Planar preconditioner

The following theorem is an adaption of theorem **Ultra-Sparsify** of [ST04], based on the construction of the lower-stretch spanning tree in [EEST05].

Theorem 4.2.1. [Monolithic Ultra-Sparsify] *Let A be a planar graph with n nodes and k be an integer. One can find a subgraph B of A, with $n-1+mO(\log^2 n \log\log n)$ edges, such that $\kappa(A,B) \leq n/m$. B can be constructed in time $O(n\log^2 n)$.*

We prove the following stronger version.

Theorem 4.2.2. [Miniaturization Ultra-Sparsify] *Every planar graph A with n nodes has a subgraph B such that: (i) $\kappa(A,B) \leq \sqrt{k}$, (ii) B can be reduced via greedy Gaussian elimination of degree 1 and 2 vertices to a planar graph C with $O(n\log^3 k/\sqrt{k})$ nodes. Given the decomposition of Theorem 3.0.1, the embedded graphs B, C can be constructed with $O(n\log^2 k)$ work, in $O(k\log n)$ parallel time.*

Proof. Assume we are given the partition of Theorem 3.0.1. Let $\mathcal{A} = \{A_i\}$ be the components of the partition, and $W_i = A_i \cap W$. We have $\sum_i |W_i| = O(n/\sqrt{k})$, and thus $\sum_i |A_i| \leq 2n$.

Every edge of A is contained in at least one A_i, and in at most two; if it is contained in two, each cluster gets half of its weight. In this way, we get $A = \sum_i A_i$. We let B_i be the subgraph of A_i constructed by setting $|A_i|/m = \sqrt{k}$ in Theorem 4.2.1. We have $|B_i| = |A_i| - 1 + |A_i|O(\log^3 k/\sqrt{k})$, and $\kappa(A_i, B_i) = \sqrt{k}$. The preconditioner will be $B = \sum_i B_i$. By Lemma 2.1.8, we get $\kappa(A,B) = \sqrt{k}$. To obtain the partial Cholesky factorization $B = L[I,0;0;C]L^T$, we will be greedily removing degree one and two nodes in the interior of each A_i independently. Concretely, let $C_i = \texttt{Eliminate}(B_i, W_i)$ so that $C = \sum_i C_i$. The algorithm `Eliminate` is given in Section 2.4.5. By Lemma 2.3.3 we have $|C_i| \leq 4(|W_i| + |A_i|\log^3 k/\sqrt{k})$, which gives $|C| \leq \sum_i |C_i| = O(n\log^3 k/\sqrt{k})$.

Each B_i can be constructed independently in time $O(|A_i|\log^2 k)$ using Theorem 4.2.1. Hence, the total work for the construction of B is $\sum_i |A_i|\log^2 k = O(n\log^2 k)$. Furthermore, as discussed in Section 2.3.3, since there are no edges between the graphs A_i, the matrix L can be written as a product $L = \prod_i L_i$, where each L_i can be computed independently from the partial Cholesky factorization of A_i in time $O(k)$. This finishes the proof. □

4.2.1 Sequential complexity

The hierarchy is constructed by applying recursively Theorem 4.2.2. The theorem allow us to pick a value k that satisfies $\tau_{\mathcal{H}}/\mu_{\mathcal{H}} = 1/2$ for a constant t. The time to construct the hierarchy is $T = \sum_i O((k+\log^2 k)n/\mu_{\mathcal{H}}^i) = O(kn)$. In comparison, using the monolithic

preconditioner provided by Theorem 4.2.1, the least τ_H for which $\tau_{\mathcal{H}}/\mu_{\mathcal{H}} < 1/2$ can be achieved is in the order of $O(\log^2 n \log\log n)$.

4.2.2 Parallel Complexity

Let us now turn our attention to the potential for parallelism in algorithm **Solve**. By Theorems 3.0.1 and 4.2.2, the hierarchy of graphs can be constructed in $O(k \log^2 n)$ time with $O(nk \log^2 k)$ work. At any point of time, the total memory needed by the algorithm is $O(n)$, since for each i we need to store a graph of size $O(n/t^i)$ and a constant number of vectors of size n/t^i. One Chebyshev iteration consists only of a constant number of sparse matrix-vector multiplications and vector additions. Using n processors, the vector operations can be performed in time $O(1)$, and the matrix-vector multiplication in time $O(\log n)$ with work linear in the size of the vector. Both the sequential and the parallel algorithms make the same number of Chebyshev iterations, and thus the total parallel work is proportional to the total sequential work, for a fixed value of k.

The Chebyshev iterations have to be performed sequentially, so the dominating factor in the time complexity of the parallel algorithm is the total number of Chebyshev iterations which is dominated by the $O(t^r)$ iterations done at the bottom of the hierarchy. Let $m = t^c$. Given that $|A_r|$ is constant, we have $r \leq \log_m n$, and $t^r = O(n^{1/c})$. The algorithm of Spielman and Teng can achieve a c arbitrarily close to 2, though at the expense of the total work done by the algorithm. For example, ignoring $\log\log n$ terms, if we set $n/m = \log^8 n$ in Theorem 4.2.1, we get $t = \log^4 n$ and $m = \log^6 n$, thus $c = 3/2$. Observe that the parallel time complexity is up to a $polylog(n)$ factor equal to $n^{1/c}$ even when we use nearly $n^{1-1/c}$ processors. Theorem 4.2.2 also guarantees that c can be taken arbitrarily close to 2, while the total work remains $O(n)$ with only a larger hidden constant.

We can improve the number of Chebyshev iterations while keeping the amount of work linear, by stopping the recursion at a higher level. For simplicity, in the following discussion we omit inverse polylogarithmic factors in the size of A_r and polylogarithmic factors in the parallel time complexity. Let $|A_r| = n^\alpha$. We have $r = (1-\alpha)\log_k n$, and $t^r = n^{(1-\alpha)/c}$. To solve the systems in A_r we will use the parallel nested dissection algorithm of Pan and Reif [PR93]. The algorithm requires as input a tree of small vertex separators for A_r. This can be constructed one time, with $o(n)$ work, and in $n^{(1-\alpha)/c}$ time using Klein's algorithm [Kle93]. Then, the algorithm obtains a one-time factorization of A_r in $polylog(n)$ time, with $O(n^{3\alpha/2})$ work, which is linear if $a = 2/3$. Then, every system in A_r can be solved in $polylog(n)$ time, with $O(n^\alpha)$ work. The total amount of work for solving the systems in A_r is $O(n^{(1-\alpha)/c} n^\alpha) = o(n)$. Hence the parallel time complexity approaches $O(n^{1/6})$ as c approaches 2, and the algorithm can use only $O(n^{5/6})$ processors.

4.2.3 Implementation and practicality notes

We believe that besides the theoretical improvement, our method can lead to more practical implementations. An appealing characteristic of the miniaturization approach is the fact that it disconnects the problem of the existence of a good preconditioner from its construction. For example, in this paper, we use the preconditioners of Spielman and Teng for the construction of the mini preconditioners. However, without giving the details here, let us note that we can substitute them entirely with the Steiner support trees introduced in [Gre96] and analyzed in [MMP$^+$05], affecting only the hidden constant in the total work of the algorithm. Steiner trees are provably better for many natural families of graphs [MR04, MMP$^+$05]. A major obstacle in their applicability as preconditioners was that the algorithm for their construction is polynomial in the size of the graph. This is no longer a problem.

The increased hidden constant in the construction of the preconditioner may actually be desirable. In most applications, one is interested in solving many linear systems with a given Laplacian. The preconditioners depend only on the given graph, hence they are constructed a single time. In those situations, it makes sense to spend more time on the construction of the preconditioners. This is because their quality affects the running time for every system that is solved; to guarantee fast convergence, the solver must do a certain number of iterations. Otherwise the convergence can be arbitrarily slow. Apart from the extra time for the design of the miniature preconditioner, one can also spend extra time for measuring its quality. With a global preconditioner, one has to assume the worst case theoretical guarantee for the quality of the preconditioner. This guarantee may be too pessimistic, but there is no way to decide quickly if this is indeed the case. In our approach, the actual quality can be measured easily, and the corresponding parameters in the solver can be adjusted accordingly. Testing the quality of the preconditioner is also useful when a fast algorithm for constructing the preconditioner is good on typical instances, but may occasionally fail, as it is the case with algorithms for constructing Steiner trees. Failure instances can be detected, and the more expensive accurate algorithm will be run only on them.

Finally, we note that the idea of using small separators has also been used in nested dissection [LRT79], which uses the full tree of separators of the given graph. Our algorithm cannot avoid computing a number of decompositions, even in the case that the given graph has a directly available tree of separators. The reason is that after the preconditioner undergoes the reduction to the smaller graph which does not inherit the tree of separators from the original graph. So, the use of the decomposition algorithm seems to be necessary even when the original system corresponds to a weighted square grid.

Chapter 5

Edge separators and Steiner preconditioners

Given a graph A with n vertices, a *Steiner support graph* S is a graph with n vertices corresponding to the vertices of A (called the A-vertices) and m extra or *Steiner* vertices. Gremban and Miller showed that Steiner graphs can be used as preconditioners [Gre96]. The analysis of their quality can be reduced to the analysis of the generalized eigenvalues of the pair (A, B) where B is the Schur complement with respect to the Steiner vertices of S. We will call B the *effective preconditioner*.

Gremban used the fact that $\sigma(A, S) = \sigma(A, B)$ (proposition 6.1 in [BH03]) to give easy bounds on $\sigma(A, B)$. In the other direction, bounding the support $\sigma(B, A)$ is a difficult task because not only B is dense, but in general it doesn't have a closed analytic expression. For very regular graphs like the unweighted d-dimensional grid, Gremban actually calculated closed analytic expressions and proved bounds on $\sigma(B, A)$. More generally, until the paper of Maggs et. al [MMP$^+$05] it was not known whether there is a good Steiner tree preconditioner. However, their analysis concerns only Steiner trees. In this Chapter we present a way for analyzing the support for more general Steiner graphs. For certain cases of trees, the new bound strengthens the bounds of [MMP$^+$05]. The analysis is quite simple, and has an impact on the practical design of solvers for graphs which *a priori* have certain nice properties.

The presentation is based on the following characterization of $\sigma(B, A)$, shown in [MMP$^+$05].

Lemma 5.0.3. *If S is a Steiner graph for A and B_S is Schur complement with respect to*

the elimination of the Steiner vertices of S, we have

$$\sigma(B_S, A) = \max_x \min_y \left(\begin{pmatrix} x \\ y \end{pmatrix}^T S \begin{pmatrix} x \\ y \end{pmatrix} \right) / x^T A x$$

where $y \in \mathbb{R}^m$.

Lemma 5.0.4. [Steiner support transitivity] *Let S', S be Steiner graphs for A, with the same number of vertices. Also, let $B_{S'}, B_S$ be the Schur complements with respect to the elimination of the Steiner vertices of S', S. We have*

$$\sigma(B_S, A) \leq \sigma(S, S')\sigma(B'_S, A).$$

Proof. Lemma 5.0.3 implies that for all vectors $x \in \mathbb{R}^n$ there is a vector $y_x \in \mathbb{R}^m$ such

$$(x|y_x)^T S'(x|y_x) \leq \sigma(B'_S, A)(x^T A x).$$

By the definition of $\sigma(S, S')$ this implies that for all vectors x, we have

$$(x|y_x)^T S(x|y_x) \leq \sigma(S, S')\sigma(B'_S, A)(x^T A x).$$

Then, Lemma 5.0.3 implies directly the bound on $\sigma(B_S, A)$. □

In the following, to simplify our notation, whenever it is understood that S is a Steiner graph of A, we will denote $\sigma(B_S, A)$ by $\sigma(S, A)$.

5.1 An illustrative example

Let A_{2n} be the cycle graph with $2n$ vertices, and let the support graph S be the graph consisting of the *quotient* cycle graph A_n attached to the $2n$ leaves as shown in Figure 5.1. We are interested in bounding $\sigma(S, A_{2n})$. The technique that we present here is easy but it hasn't appeared elsewhere. One can observe that $\sigma(S, A_{2n}) = \sigma(S + A_{2n}, A_{2n}) - 1$. As an intermediate step in bounding $\sigma(S + A_{2n}, A_{2n})$, we will bound $\sigma(S + A_{2n}, S + A_{2n} - A_n)$. The graphs $S + A_{2n}$ and $S + A_{2n} - A_n$ are shown in Figure 5.2. Every edge of the quotient A_n can be supported by the obvious shortest path in $S + A_{2n} - A_n$ which has effective resistance 2. The supporting paths are disjoint, hence $\sigma(A_n, S + A_{2n} - A_n) \leq 2$ and

$$\begin{aligned} \sigma(S + A_{2n}, S + A_{2n} - A_n) &= \sigma((S + A_{2n} - A_n) + A_n, S + A_{2n} - A_n) \\ &\leq \sigma(S + A_{2n} - A_n, S + A_{2n} - A_n) + S(A_n, S + A_{2n} - A_n) \\ &\leq 3. \end{aligned}$$

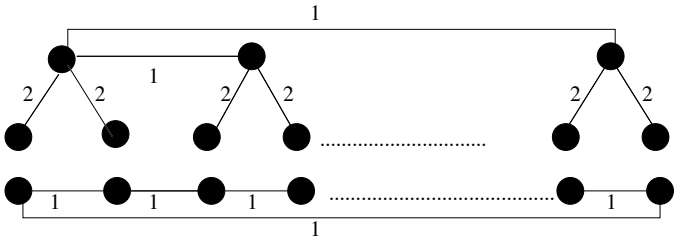

Figure 5.1: A support analysis example.

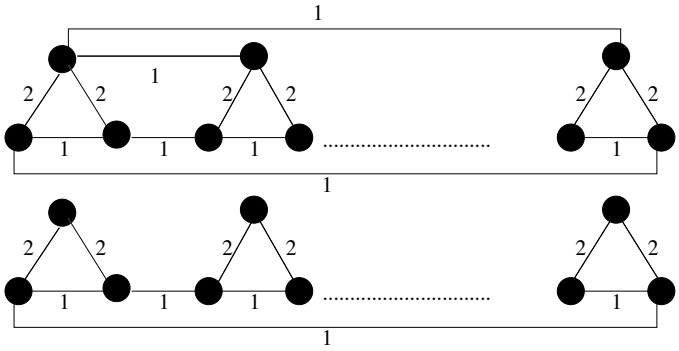

Figure 5.2: The graphs $S + A_{2n}$ and $S + A_{2n} - A_n$.

Now computing the effective preconditioner with respect to $S + A_{2n} - A_n$ is easy, and we directly get $\sigma(S + A_{2n} - A_n, A_{2n}) \leq 2$. Combining the above inequalities and using the Steiner support transitivity, we have

$$\begin{aligned} \sigma(S, A_{2n}) &= \sigma(S + A_{2n}, A_{2n}) - 1 \\ &\leq \sigma(S + A_{2n}, S + A_{2n} - A_n)\sigma(S + A_{2n} - A_n, A_{2n}) - 1 \leq 5. \end{aligned}$$

5.2 Laminar decompositions and Steiner graphs

A **laminar decomposition** of a given graph $G = (V, E, w)$ is a collection $\mathcal{H} = H_0, \ldots H_l$ of l partitions of the vertices of G into disjoint clusters, with the property that each cluster in H_i is a proper subset of a cluster in the H_{i-1}. We will refer to H_i as the i^{th} level of the decomposition. By definition, every cluster at level $i - 1$ is partitioned to its children clusters in level i. We let $H_0 = V$ and level l contain the vertices of G as singletons.

A laminar decomposition H naturally defines a tree $T_H = (V_T, E_T, w_T)$. The i^{th} level of T_H consists of vertices corresponding (one-to-one) to the subsets in the i^{th} level of H. For each vertex t of the tree we denote by $V_t \subseteq V$ the set of vertices it corresponds to. A vertex t is connected to its parent vertex in T_H with an edge of weight $out(V_t)$. All the Steiner trees that we present in this Chapter as well as the trees of [MMP+05] follow this definition, and we will call them *laminar Steiner trees*.

Given a graph $G = (V, E, w)$, a set $S \subseteq V$ and a laminar decomposition H for S, we define the **local Cheeger constant** of $S \subset V$ as the ratio

$$\gamma(S) = \min_{T \subseteq S} \frac{cap(S - T, T)}{out(T)}. \tag{5.1}$$

and the its restriction to H by

$$\gamma_H(S) = \min_{T \in H} \frac{cap(S - T, T)}{out(T)}. \tag{5.2}$$

The local Cheeger constant was introduced and studied recently by Chung in [Chu07], but has also been used in [BKR03] where it is called the **precondition property**. We will denote the local Cheeger constant of a given set S by $\gamma(S)$.

We now describe a way for constructing more general Steiner graphs.

Definition 5.2.1. [Quotient and Steiner graph] *Let P be an edge cut, i.e. a partitioning of the vertices of the graph A into disjoint sets V_i, $i = 1, \ldots, m$. Let A_i be the graph induced by the vertices in V_i. We let H_i be a laminar decomposition of A_i and T_i be the corresponding laminar Steiner tree. We define the* **quotient graph** *Q on the set of the roots of the trees T_i, by letting $w(r_i, r_j) = cap(V_i, V_j)$. We define the* **Steiner graph** *with respect to P, as $S_P = Q + \sum_{i=1}^{m} (\gamma_{H_i}(A_i))^{-1} T_i$.*

Theorem 5.2.2. *Let P be an edge cut, and S be the Steiner graph with respect to P. If $h = \max_i height(T_i)$ we have $\sigma(A, S) \leq 2h + 1$ and*

$$\sigma(S, A) \leq (2h + 2)(1 + \max_i \left((\gamma_{H_i}(P_i))^{-1} \sigma(T_i, A_i) \right)).$$

Proof. The key observation to bound the support numbers is what we will call the *sufficient capacity* property: every non-root vertex t in S is connected to the upper level with capacity at least equal to $out(V_t)$, because every laminar Steiner tree is multiplied by $(\gamma_{H_i}(P_i))^{-1}$. To bound $\sigma(A, S)$ we embed A onto S by routing each edge e of A

via the shortest (with respect to the number of hops) path $p(e)$ in S that goes between the endpoints of e. The length of the path is at most $2h+1$. The sufficient capacity property ensures that the congestion of each edge of S in the embedding is at most 1. To bound $\sigma(S,A)$ we will apply the technique we illustrated in Section 5.1. We observe that $\sigma(S,A) = \sigma(S+A,A) - 1$. Consider again an edge e whose associated path $p(e)$ in S uses the edge (r_i, r_j) in Q. We route $w(e)$ units from (r_i, r_j) through $p(e)$. Doing this for every edge e of A defines an embedding of Q into $S + A - Q$. The dilation of the embedding is $2h+1$ and the sufficient capacity property ensures that the congestion is 1. This proves that $\sigma(S+A, S+A-Q) \leq 2h+2$. Finally, $\sigma(S+A-Q,A) = \sigma(S-Q,A)+1$. The graph $S-Q$ consists of the disjoint trees $(\gamma(P_i))^{-1}T_i$, hence $\sigma(S-Q,A) \leq \max_i \sigma((\gamma_{H_i}(P_i))^{-1}T_i, A_i) = \max_i(\gamma_{H_i}(P_i))^{-1}\sigma(T_i,A_i)$. Combining these bounds, and using the Steiner support transitivity, we get

$$\begin{aligned}
\sigma(S,A) &= \sigma(S+A,A) - 1 \\
&\leq \sigma(S+A, S+A-Q)\sigma(S+A-Q,A) - 1 \\
&\leq (2h+2)(1 + \max_i(\gamma_{H_i}(P_i))^{-1}\sigma(T_i,A_i)) - 1.
\end{aligned}$$

□

Remark. The intention behind the multiplication of T_i by $(\gamma_{H_i}(P_i))^{-1}$ is to ensure the sufficient capacity property. In practice we only need to separately scale the edges of S so that each edge has congestion exactly 1. This won't have an impact on the bound for the maximum generalized eigenvalue but it may be have practical consequences especially when the Steiner graph is used in preconditioned Conjugate Gradients.

5.3 Steiner graphs and linear time solvers for uniform d-dimensional model grids

We apply the theory developed in the previous section to uniform d-dimensional meshes. Our following discussion for the $n \times n$ square grid (denoted by A) extends easily to higher dimensions. We let the edge separator P be the natural partitioning of the vertices into $k \times k$ squares. For each $k \times k$ square A_i, we let T_i be the star graph with edge weights equal to 4, We construct the Steiner graph S_k as described in the previous section. It is clear that $\sigma(A, S_k) \leq 3$. In order to analyze $\sigma(T_i, A_i)$ we state the following general theorem.

Theorem 5.3.1. *Let A be a graph whose vertices have volumes $a_1 \leq a_2 \ldots \leq a_n$. Then if*

S is the star graph with n edges with weights $a_1 \leq a_2 \ldots \leq a_n$, we have $\sigma(S, A) \leq 4/\phi_A^2$, where ϕ_A is the conductance of A.

Proof. By definition we have $\sigma(S, A) = \sigma(B, A)$ where B is the Schur complement with respect to the elimination of the root of D. The edges weights for B are given by $b_{i,j} = a_i a_j / \sum_k a_k$ [Gre96]. For the volume b_i of the vertex i in B, we have

$$b_i = a_i \frac{\sum_{j \neq i} a_k}{\sum a_k} = a_i \left(1 - \frac{a_i}{\sum a_k}\right) \geq a_i/2$$

So, if D_G denotes the diagonal of the Laplacian G, we have $x^T D_B x \geq x^T D_A x/2$.

$$\begin{aligned} \sigma(B, A) &= \max_x \frac{x^T B x}{x^T A x} = \max_x \frac{x^T D_A x}{x^T A x} \frac{x^T B x}{x^T D_A x} \\ &\leq 2 \max_x \frac{x^T D_A x}{x^T A x} \frac{x^T B x}{x^T D_B x} \leq 2\lambda_{max}(D_B^{-1} B) \lambda_{\min}^{-1}(D_A^{-1} A) \end{aligned}$$

We have $\lambda_{\min}(D_A^{-1} A) \geq \phi_A^2/2$ by Cheeger inequality, and $\lambda_{\max}(D_A^{-1} A) x \leq 2$. This completes the proof. □

Using now the fact that the conductance of the $k \times k$ square grid is $1/k$, we have $\sigma(T_i, A_i) = O(k^2)$.

Let us now describe briefly the application of the Steiner graphs in the recursive preconditioned Chebyshev algorithm we described in Section 4.1.2. When using S_k as a preconditioner for A, the algorithm requires the solution of a system with S_k. After obtaining a partial Cholesky factorization of S_k by eliminating the leaves of S_k we recursively call the Chebyshev algorithm to solve a system in the quotient Q. We note that Q is the $n/k \times n/k$ grid with weights equal to k. The process can be continued recursively to give a hierarchy $\mathcal{H} = \{A_i, B_i\}$ of graphs and preconditioners where each A_i is a square grid and $|A_i|/|A_{i+1}| \leq k^2$ while $\kappa(A_i, B_i) = O(k^2)$. We can properly scale the graphs in $\mathcal{H}$ so that $\lambda_{\min}(A, B) = 1$, and plug them in the algorithm **Solve** of section 4.1.2. Theorem 4.1.1 bounds the running time of **Solve** by $O(kn)$ for a sufficiently large constant k.

The theory of Section 5.2 and the construction of this section are applicable to any family of graphs that possess similar regularities and self-similarities that are known a priori, or more generally are decomposable into small clusters that have a bounded local Cheeger constant. Much of the research in geometric multigrid has been devoted to extending its analysis to families of grids that deviate from the model d-dimensional grid. Our analysis provides linear time algorithms for several of these families, and strict running time

bounds for cases that are quite possibly problematic for the geometric multigrid theory, such as grids with holes or missing corners. We note that we can speed up the solver by a constant, if we use as a preconditioner for each A_i not the star graph, but the natural quadtree that was analyzed by Gremban [Gre96], which satisfies the much stronger bound $\sigma(T_i, A_i) = O(k)$. More generally, one should always use as a building block of the preconditioner the trees analyzed in [MMP+05] which always guarantee a condition number at most proportional to the number of vertices of A_i. This can be crucial for convergence, for example in the case of the line: preconditioning segments of the line of size k with the star graph, gives a hierarchy $\mathcal{H}$ that does not satisfy the requirement of Theorem 4.1.1, because $\kappa(A_i, B_i) = O(k^2)$ and $|A_i|/|A_{i+1}| = k$. In contrast, preconditioning the segments with the binary tree gives a hierarchy with $\kappa(A_i, B_i) = O(k)$ and $|A_i|/|A_{i+1}| = k$. For less regular graphs, constructing better tree preconditioners also increases the flexibility for absorbing larger local Cheeger constants and varying weights in the quotient part of the Steiner graph.

5.4 Additions to the theory of Support trees

In this Chapter we provide additional tools for analyzing the trees of [BKR03] that were demonstrated to be effective preconditioners in [MMP+05].

5.4.1 Laminar decompositions with guarantees

Assume we are given a laminar decomposition $\mathcal{H} = \{\mathcal{H}_i\}$, for $i = 0, \ldots, l$, of a graph G. The level 0 set is V while level l contains singletons. We say that an edge e of G is cut on level $l \geq 1$ if both endpoints of e are contained in the same $l-1$ level cluster, but in different l level clusters. For any set X, we define the weight function $w_l(X)$ as

$$w_l(X) = \sum_{\substack{e \in X \times V \\ level(e) \leq l}} c(e).$$

Informally $w_l(X)$ is the capacity leaving X at level l. By definition, if H is a level l cluster, we have $w_l(H) = out(H)$. Let H be a level i cluster that is furthermore divided into level $i+1$ sub-clusters $H_1, \ldots, H_k$. Let $G[H]$ denote the graph induced by the vertices in H. Now contract the nodes in each cluster of $G[H]$ to form a graph X_H with k vertices, and edge weights equal to the total capacity of the edges between the clusters of $G[H]$. We call X_H the contraction of $G[H]$ and we denote by $\phi(X_H)$ its conductance. We define the

contraction conductance of the laminar decomposition $\mathcal{H}$ as

$$\phi_{\mathcal{H}} = \min_{H \in \mathcal{H}} \phi(X_H).$$

We define a Concurrent Multicommodity Flow (CMCF) problem for every cluster H at level $i < l$. The problem consists of one commodity $d(u, v)$ for each pair of vertices u and v of H. The demand for $d(u, v)$ is given by

$$dem(u, v) := \frac{w_{i+1}(u) \cdot w_{i+1}(v)}{w_{i+1}(H)}. \tag{5.3}$$

Note that $dem(u, v)$ for level l, depends on the weight function w_{l+1}. The *throughput fraction* of a solution to a CMCF problem is the minimum over all commodities, of the fraction of the commodity's demand that is met by the solution. A cut of the subgraph induced by a cluster H is a partition of H in to subsets A and $B = H - A$. The *sparsity* of the CMCF problem on H is defined as $\phi = \min_A \frac{cap(A,B)}{dem(A,B)}$ where $dem(A, B)$ is the demand of the CMCF problem that is separated by the cut. Räcke et.al. [BKR03] described an algorithm which -using an oracle that approximates a sparsest cut of given CMCF problem within σ- constructs a laminar decomposition $\mathcal{H}$ with the following guarantees:

- Each cluster $H \in \mathcal{H}$ satisfies the *throughput property*: the throughput for the CMCF problem in H is $\Omega(1/(\sigma^2 \log n))$.
- Each cluster $H \in \mathcal{H}$ satisfies the *precondition property*

$$\min_{T \subseteq H, |T| \leq 3/4|H|} \frac{cap(H - T, T)}{out(T)} = \Omega(1/(\sigma \log n)).$$

- $\mathcal{H}$ consists of $O(\log n)$ levels.

We will denote by $\mathcal{H}_{BKR}$ the laminar decomposition of [BKR03]. We are now ready to prove a basic property of $\mathcal{H}_{BKR}$.

Theorem 5.4.1. *The contraction conductance of $\mathcal{H}_{BKR}$ is $\Omega(1/(\sigma^3 \log^2 n))$.*

Proof. Let H be a level i cluster that is furthermore divided into level $i+1$ sub-clusters $H_1, \ldots, H_k$. Let X be the contraction of $G[H]$. Let $c(H_i)$ denote the total incident capacity of the node corresponding to H_i in X. Since H satisfies the precondition property, we have

$$c(H_i) = cap(H_i, H - H_i) \geq \Omega(\frac{1}{\sigma \log n}) w_{l+1}(H_i)$$

and

$$vol(X) = \sum_i c(H_i) \geq \Omega(\frac{1}{\sigma \log n}) \sum_i w_{l+1}(H_i) \geq \Omega(\frac{1}{\sigma \log n}) w_{l+1}(H).$$

Consider an arbitrary partition of X into sets A and $B = X - A$. We have

$$\begin{aligned} dem(A,B) &= \sum_{u \in A, v \in B} \frac{w_{l+1}(u) \cdot w_{l+1}(v)}{w_{l+1}(H)} \\ &= \frac{\left(\sum_{u \in A} w_{l+1}(u)\right)\left(\sum_{v \in B} w_{l+1}(v)\right)}{w_{l+1}(H)} \\ &\geq \frac{vol(A)vol(B)}{\sigma \log n \cdot vol(X)} \end{aligned}$$

Using the last inequality, we get

$$\phi(X) \geq \frac{cap(A,B)}{min\{vol(A), vol(B)\}} \geq \frac{cap(A,B)}{2vol(A)vol(B)/vol(X)} \geq \frac{1}{2\sigma \log n} \frac{cap(A,B)}{dem(A,B)}.$$

By the throughput property, we also have

$$\frac{cap(A,B)}{dem(A,B)} \geq q_{\min} = \Omega(1/(\sigma^2 \log n)).$$

Hence, $\phi(X) = \Omega(1/(\sigma^3 \log^2 n))$. $\square$

5.4.2 A new bound for laminar Steiner trees

The general $O(n \log^3 n)$ bound of [MMP+05] for the support number $\sigma(T, A)$ of the graph A and the laminar Steiner tree T for $\mathcal{H}_{BKR}$ is up to the log factors optimal for general graphs. However there are cases where this upper bound does not capture the actual value of $\sigma(T, A)$. For example, assuming $\sigma = 1$, if the laminar decomposition contains only one level the graph is an expander. Then, Theorem 5.3.1 shows that $\sigma(T, A)$ is constant. In this section we give a bound on $\sigma(T, A)$ in terms of the depth d of the tree, the contraction conductance $\phi_{\mathcal{H}}$ and the restricted local Cheeger constant $\gamma_{\mathcal{H}}$. Thus the following Theorem directly implies a better bound for laminar Steiner trees corresponding to a bounded depth $\mathcal{H}_{BKR}$.

Theorem 5.4.2. *Let $\mathcal{H} = \{H_1, \ldots, H_d\}$ be a laminar decomposition with contraction conductance $\phi_{\mathcal{H}}$, restricted local Cheeger constant $\gamma_{\mathcal{H}}$ and depth d. Let T be the associated laminar Steiner tree. We have $\sigma(T, G) \leq (1 + 16/(\gamma_{\mathcal{H}} \phi_{\mathcal{H}}^2))^d$.*

Proof. We construct a sequence of graphs $G_0, \ldots, G_d$. G_i is constructed by contracting the clusters at level i, so $G_d = G$ and G_0 is a single vertex. The laminar support tree T has d levels and by construction the weight of the edge connecting a level i vertex to its parent is equal to the total incident weight of the corresponding vertex in G_i. Let T_i denote the edges of T that are below level i. By Lemma 5.0.3, we have

$$\sigma(T, G) \leq \sigma(T + \sum_{j=1}^{d-1} G_j, G).$$

We bound the latter quantity by using the technique of Section 5.1 d times, for each pair of consecutive levels in the hierarchy. More formally, we use induction to show that for all $i < d$, we have

$$\sigma(T_i + \sum_{j=i}^{d} G_j, G) \leq (1 + 16/(\gamma_{\mathcal{H}} \phi_{\mathcal{H}}^2))^{d-i}.$$

We first consider the base case $j = d - 1$, where we need to bound $\sigma(T_{d-1} + G_{d-1})$. Let $H_{d-1} = V_1, \ldots, V_k$ be the partitioning of the vertices at level $d - 1$. By the assumption for the contraction conductance, for all i, the graph A_i induced by V_i has Cheeger constant bounded below by $\phi_{\mathcal{H}}$, and local Cheeger constant bounded below by $\gamma_{\mathcal{H}}$. Let S_i denote the star graph with edge weights equal to the volumes of the vertices in A_i. By definition, we have

$$T_{d-1} + G_{d-1} \preceq G_{d-1} + \gamma_{\mathcal{H}}^{-1} \sum_i S_i.$$

By the Cheeger inequality we have $\sigma(S_i, A_i) \leq 4/\phi_{\mathcal{H}}^2$ and thus applying Theorem 5.2.2 gives $\sigma(T_{d-1} + G_{d-1}, G) \leq 16/(\gamma_{\mathcal{H}} \phi_{\mathcal{H}}^2)$. This completes the proof for the base case.

For the induction, we note that -similarly to the base case- applying Theorem 5.2.2 gives

$$\sigma(T_i - T_{i+1} + G_i, G_{i+1}) \leq 16/(\gamma_{\mathcal{H}} \phi_{\mathcal{H}}^2).$$

By using this and the Steiner support transitivity, we have

$$\begin{aligned} \sigma(T_i + \sum_{j=i}^{d} G_i, G) &= \sigma(T_i - T_{i+1} + G_i + T_{i+1} + \sum_{j=i+1}^{d} G_i, G) \\ &\leq (1 + 16/(\gamma_{\mathcal{H}} \phi_{\mathcal{H}}^2)) \sigma(T_{i+1} + \sum_{j=i+1}^{d} G_i, G). \end{aligned}$$

The claim follows by the inductive hypothesis. □

5.5 Planar multiway edge separators

Let P be a partition of the vertices of a graph $G = (V, E, w)$ into disjoint sets V_i, $i = 1, \ldots, k$ and let G_i denote the graph induced by the vertices in V_i. We call n/k the *vertex reduction factor* of P. We call P a (ϕ, γ)-decomposition if the conductance of each G_i is bounded below by ϕ and for each vertex $v \in V_i$, $cap(v, V_i)/vol(v) \geq \gamma$. We allow some of the V_i's to be singletons in which case we define the conductance of G_i to be equal to 2. We call P a $(\phi^{loc}, \gamma^{avg})$ decomposition if $\sum_i \sum_{v \in V_i} cap(v, V_i)/vol(v) \geq \gamma$ and the conductance of each G_i is bounded below by ϕ.

Spielman and Teng considered the problem of computing a (ϕ, γ^{avg})-decomposition for unweighted graphs [ST04]. They note that there is no nearly linear time algorithm for computing such a decomposition with good guarantees. Kannan et.al. analyzed the recursive application of any given algorithm for approximating the sparsest cut [KVV04]. While they showed that it obtains a good approximation to the optimal $(\phi^{loc}, \gamma^{avg})$ - decomposition of a given graph, they didn't prove guarantees that are independent from the instance. The laminar decomposition $\mathcal{H}_{BKR}$ gives a sequence of graphs G_i and partitions P_i, such that G_{i+1} is the contraction of the vertices of G_i with respect to the clusters in P_i. The discussion in the previous section implies that every P_i is a $(1/(\sigma^3 \log^2 n), 1/(\sigma \log n))$-decomposition for G_i, where σ is the approximation factor obtained by an algorithm for the computation of the sparsest cut. The vertex reduction factor is constant in average, but there are no guarantees for the reduction factor between G_1 and G_2.

In this section we consider a variant of (ϕ, γ)-decompositions. For each G_i we introduce a vertex on each edge leaving G_i. If W_i is the set of newly introduced vertices for G_i, we say that P is $[\phi, \rho]$-decomposition if the *closure* graph G_i^o induced by the vertices in $V_i \cup W_i$ has conductance bounded below by ϕ and the vertex reduction factor of P is at least ρ. By definition, G_i^o is G_i with additional degree one vertices hanging off of it. Therefore, any edge cut in G_i induces a sparser cut in G_i^o, and thus the conductance of G_i must be lower bounded by ϕ. Also note that if G_i contains two vertices v_1, v_2 such that $cap(v_i, V_i)/vol(v_i) \leq \phi$, the conductance of G_i^o is less than ϕ; this can be seen by considering the edge cut consisting of the edges incident to v_1, v_2 in G_i. Hence there can be no more than one vertex violating the γ constraint, if $\gamma < \phi$. So a $[\phi, \rho]$-decomposition is "almost" a (ϕ, γ) decomposition with the additional guarantee for the vertex reduction factor. In Chapter 7 we will see that $[\phi, \rho]$-decompositions are useful in multigrid algorithms. Typically, we will be interested in minimizing the product $\phi\rho$. We first consider the problem for trees.

Theorem 5.5.1. *Trees have a $[1/2, 6/5]$-decomposition that can be computed with linear work in $O(\log n)$ parallel time.*

Proof. If the tree contains 2 or 3 vertices the decomposition consists of only one cluster. The basic step of the algorithm is to compute the 3-critical vertices of the given tree T, using the algorithm of Miller and Reif. For the definitions we refer the reader to [RMMM93]. This step can be done with linear work in $O(\log n)$ parallel time. Let us now describe the decomposition P of T into disjoint sets of vertices. Assuming that T has n vertices, the number of 3 critical vertices is at most $2n/3$. Although we will allow critical vertices to be singletons in P, we will not allow non-critical vertices to be singletons. This implies that after the contraction of the clusters the tree will have at most $2n/3 + n/6$ vertices, which gives (asymptotically) the reduction factor. We start by forming a cluster per critical vertex, each containing initially only the critical vertex. The 3-critical vertices

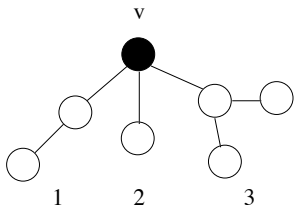

Figure 5.3: External 3-bridge with possible attachments.

decompose the edges of T into connected 3-bridges of two types. *External* 3-bridges contain only one critical vertex and *internal* 3-bridges contain two critical vertices. Each external 3-bridge is formed by the critical vertex v which is the shared root of a number of trees T_i. The 3 possible cases of T_i are depicted in Figure 5.3, where the black vertex is the critical vertex. In cases 1,3 we form clusters with the non-critical vertices and we add them to P. The closure of these clusters has conductance 1. We also add to the cluster of v its attached leaves (case 2).

An *internal* 3-bridge that contains two critical vertices contains at most 2 non-critical vertices. In Figure 5.4 we give the three possible 3-bridges of this type. In case 2, we form a cluster for the two non-critical vertices, the conductance of its closure is obviously 1. In case 3, we assign the non-critical vertex v to the cluster of the adjacent critical node which has the heaviest connection to v. Finally for case 2 we have the following subcases: (i) if $e_2 < e_1$ and $e_2 < e_3$, we assign v_1, v_2 to the clusters of their adjacent critical vertices, otherwise (ii) we form a cluster with v_1, v_2 and we add it to P. The closure of the cluster has conductance at least $1/2$.

We are left with the clusters of the critical vertices which we add to P. By the construction in the previous step, the closure of each cluster has the critical vertex v as a root shared by a number of edges and at most two paths of the form (v, u_1, u_2), where

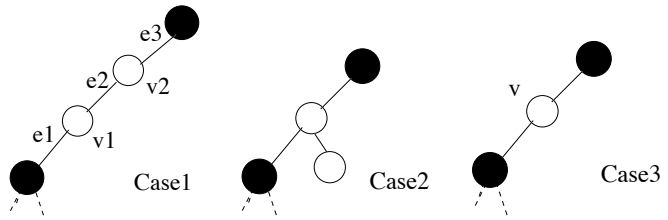

Figure 5.4: The possible internal 3-bridges.

$w(v, u_1) \geq w(u_1, u_2)$. It is then easy to see that the conductance of the closure is at least $1/2$. □

Theorem 5.5.2. *Planar graphs have an $[\phi, \rho]$-decomposition such that $\phi\rho$ is constant. The decomposition can be constructed with linear work in $O(\log n)$ parallel time.*

Proof. Let $A = (V, E, w)$ be any planar graph, and B be the subgraph constructed in Theorem 4.2.2. The graph B contains a subset W of $O(n \log^3 k/k)$ vertices that cannot be eliminated by the greedy Gaussian elimination of degree 1 and 2 vertices.

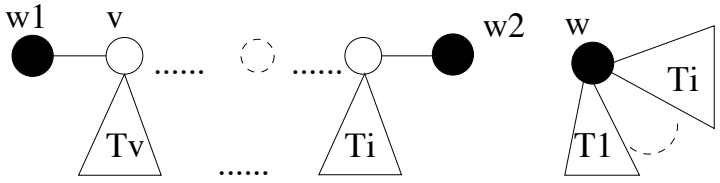

Figure 5.5: The organization of B-vertices that are greed-eliminated.

The vertices in $V - W$ either (i) lie on a path between two vertices $w_1, w_2 \in W$, or (ii) they belong to trees that are attached to the rest of B through a vertex in $w \in W$ or through a vertex $v \in V - W$ of the first kind. This is illustrated in Figure 5.5.

In the following we describe an algorithm to construct P, a $[\phi, \rho]$-decomposition of B, with $\phi > 1/4$ and ρ constant. We first construct an edge cut $\mathcal{C}$. Consider the path p between $w_1, w_2 \in W$ including w_1 and w_2. Let e be an arbitrary edge of smallest weight among the edges of p. We include e in $\mathcal{C}$. This decomposes V into disjoint trees each containing a unique vertex $w \in W$.

We will decompose each tree T_w independently. We describe the process for a given T_w. The removal of w disconnects T_w into a set of single vertices R and a number of non-trivial trees T_i with roots t_i. We form the cluster $w \cup R$ and we add it to P. The closure

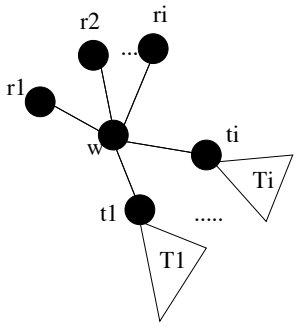

Figure 5.6: Computing a tree decomposition.

graph of $w \cup R$ is a star, so its conductance is 1. Now let $T_i' = T_i + (t_i, w)$ and compute P_i, the $[1/2, 6/5]$-decomposition of each T_i'. Each P_i includes exactly one cluster containing w. We remove w from its cluster and we add the cluster to P, along with the rest of the clusters of P_i. By construction, all clusters that are added to P are vertex disjoint. If the cluster of w in some P_i contains only two vertices, then T_i' must have at least 4 vertices, and P_i has at least 2 non-singleton clusters. This shows that we have a constant reduction in the number of vertices of T_i. In the worst case the vertices of W remain as singletons in P, but since $|W|$ is a constant fraction of n, the vertex reduction factor of P is constant.

It remains to show that the closure of the clusters in P have conductance at least $1/4$. The clusters that are not incident to an edge in $\mathcal{C}$ satisfy the constraint by construction. However we have boundary clusters each of which contains exactly one vertex which is incident to some edge in $\mathcal{C}$.

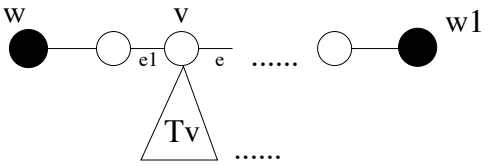

Figure 5.7: The boundary cluster

Assume that a cluster U contains a vertex v which is a adjacent to $e \in \mathcal{C}$, and let T_U be the tree induced by U. Recall that e is the lightest edge on a path between w and some

$w_1 \in W$. This scenario is depicted in Figure 5.7. Let T'_U denote the closure of T_U restricted in T_w. By construction the conductance of T'_U is at least $1/2$. We also have $T^o_U = T'_U + e$. Note that T'_U contains e_1. Hence the volume of v in T^o_U is at most two times its volume in T'_U. Hence adding e in T'_U can decrease its conductance by at most a factor of 2.

We finally claim that P is a $[1/4k, \rho]$-decomposition for A. Let $A[V_i], B[V_i]$ be the graphs induced by the cluster $V_i \in P$ in A and B respectively. Now note that $\lambda_{\max}(A, B) \leq k$. Let e_v be the vector which has a single non-zero entry corresponding to the vertex v. We have $e_v^T A e_v = vol_A(v)$, and similarly for B. It follows that the volume of v in A is at most k times its volume in B. Taking the closure $B[V_i]^o$ only adds extra leaves in $A[V_i]^o$. Since the conductance of $B[V_i]^o$ is at least 1/4, we get that the conductance of $A[V_i]^o$ is at least $1/4k$.

The graph B can be constructed with linear work in $O(\log n)$ time. The segments between the vertices of W have size at most k^2. Their decomposition can be done in parallel and with linear work per segment. □

Remarks. The essential ingredients in the proof of the above theorem is the size of W, and the fact that the volume of the vertices in A and B are within some fixed amount. For example, instead of using the preconditioner B of Theorem 4.2.2, we could have used a weaker construction B where each miniature preconditioner is just the Maximum Spanning Tree of the corresponding component. This would decrease the ϕ by a factor of k. Also, we don't need the strong vertex separators of Theorem 3.0.1, and we can substitute them with separators whose boundary size is within some constant of the interior size, so that $|W|$ is a constant factor of n. This can affect the parameter ρ by at most a constant factor.

Chapter 6

Spectral inequalities for multiway cuts

What is the effect on the spectrum of the Laplacian A a graph if we multiply its edge weights by a constant factor? Although we can in principle write the perturbed Laplacian B as $B = A + E$ where E is the perturbation matrix, applying the classical additive spectral perturbation theory [SS90] does not yield interesting bounds on the spectrum of B. The reason is that the additive theory has been developed mainly to deal with "small" ϵ-perturbations to the matrix; the eigenvalues of E can be big comparing to those of A. Clearly, we need a different notion of "relative" perturbation.

In this Chapter we study the relationship between the eigenvalues and eigenspaces of two positive definite matrices in terms of the generalized eigenvalues of the pair (A, B). We give a slight generalization of a bound of Mathias and Veselić [MV98] on the angles between their eigenspaces. We also prove a new simple but in certain cases stronger bound. We then focus on Laplacians and we show that the bounds are optimal up to a constant. We use this to demonstrate that the condition number of the pair (A^2, B^2) can be arbitrarily bad comparing to the condition number of (A, B).

We next apply the perturbation bounds to the pair of a graph A and the Schur complement of a Steiner graph B which is derived from a multi-way edge separator of A, that disconnects the graphs into vertex disjoint expanders that are more strongly connected to their interior relative to their exterior. This yields spectral inequalities that characterize the distance of any low frequency eigenvectors to a subspace of vectors with a nice description directly derivable from the partitioning of the vertices defining the edge separator.

6.1 Relative perturbation theory for Laplacians

6.1.1 Related work

There have been several papers on eigenvalue and eigenvector perturbations bounds that involve a perturbation of the matrix which is bounded in some relative sense (see for example [BD90, EI95, MV98, Li98, Li99] and the references therein). As noted in [MV98], the proofs are usually complicated and are expressed in terms of the 'relative gap' between the eigenvalues, i.e. a relative distance of the unperturbed eigenvalue to the rest of the spectrum. In this type of bounds one does not care about eigenvectors corresponding to distant eigenvalues. In contrast, the work of Mathias and Veselić [MV98] provides such bounds. We give a generalization of their bounds, as well as a new bound which is tighter for larger perturbations.

6.1.2 Perturbation bounds

Let A, B be positive definite matrices. We let $\lambda_1 \leq \ldots \leq \lambda_n$ denote the eigenvalues of A and $\mu_1 \leq \ldots \leq \mu_n$ denote the eigenvalues of B. Let $\kappa_{\max}$ and $\kappa_{\min}$ denote $\lambda_{\max}(A, B)$ and $\lambda_{\min}(A, B)$. We therefore have $\lambda_{\max}(B, A) = 1/\kappa_{\min}$ and $\lambda_{\min}(B, A) = 1/\kappa_{\max}$. The following eigenvalue bound is well known and easy to prove. We include it here for completeness.

Theorem 6.1.1. *We have $\lambda_i \leq \kappa_{\max}\mu_i$ and $\lambda_i \geq \kappa_{\min}\mu_i$.*

The estimates of Mathias and Veselić [MV98] on the relationship between the eigenspaces of A and B are stated in term of a quantity η, which is defined as $\left\|B^{-1/2}(A-B)B^{-1/2}\right\|_2$. They assume that $\eta < 1$, in which case we have the inequalities

$$\begin{aligned} \left\|B^{-1/2}(A-B)B^{-1/2}\right\|_2 &\leq \eta \\ \left\|(I + B^{-1/2}(A-B)B^{-1/2})^{-1}\right\|_2 &\leq (1-\eta)^{-1}. \end{aligned}$$

These inequalities are used in their proofs. Note however that using Lemma 2.1.6 we get

$$\Lambda(A, B) = \Lambda(B + A - B, B) = \Lambda(I + B^{-1/2}(A-B)B^{-1/2}, I)$$

from which it follows that

$$\begin{aligned} \left\|B^{-1/2}(A-B)B^{-1/2}\right\|_2 &\leq \max\{\kappa_{\max} - 1, 1 - \kappa_{min}\} \\ \left\|(I + B^{-1/2}(A-B)B^{-1/2})^{-1}\right\|_2 &= \kappa_{\min}^{-1} \end{aligned}$$

Substituting these new inequalities directly into the proof of [MV98], yields the following theorem.

Theorem 6.1.2. *Let $AX = \Lambda X$ and $BY = MY$ be the eigenvalue decompositions of A and B. That is Λ, M are diagonal matrices containing the eigenvalues and X, Y are unitary matrices whose columns are the corresponding eigenspaces of A and B respectively. Let $S = X^*Y$. Then for any j and for any set $\mathcal{S}$ not containing j we have*

$$\left(\sum_{i\in\mathcal{S}} |s_{ij}|^2\right)^{1/2} \leq \max_{i\in\mathcal{S}} \frac{\lambda_i^{1/2}\mu_j^{1/2}}{|\lambda_i - \mu_j|} \frac{\max\{\kappa_{\max} - 1, 1 - \kappa_{\min}\}}{\kappa_{\min}^{1/2}}$$

and in particular,

$$|s_{ij}| \leq \frac{\lambda_i^{1/2}\mu_j^{1/2}}{|\lambda_i - \mu_j|} \frac{\max\{\kappa_{\max} - 1, 1 - \kappa_{\min}\}}{\kappa_{\min}^{1/2}}.$$

As shown in [MV98] the bound is asymptotically optimal when $\kappa_{\max}, \kappa_{\min} \to 1$, hence it is not possible to improve it for "small" relative perturbations. However, we now show that it is possible to obtain a bound which is stronger when $\kappa_{\max}$ and $1/\kappa_{\min}$ are relatively large.

Theorem 6.1.3. *Let $\mathcal{X}, \mathcal{Y}$ be invariant subspaces of A and B respectively. Let the columns of X and Y be the normalized eigenvectors that span $\mathcal{X}$ and $\mathcal{Y}$ respectively. We have $AX = X\Lambda_X$, $BY = YM_Y$, where Λ_X, M_Y are diagonal matrices containing the corresponding eigenvalues. Let $y \in \mathcal{Y}$ and $x \in \mathcal{X}$ be unit vectors. Suppose $\min_t (\Lambda_X)_{t,t} = \lambda_i$, $\max_t (M_Y)_{t,t} = \mu_j$, and $\min_t (M_Y)_{t,t} = \mu_i$, $\max_t (\Lambda_X)_{t,t} = \lambda_j$. Then, we have*

$$(x^T y)^2 \leq \min\{\kappa_{\max}\frac{\mu_j}{\lambda_i}, \frac{1}{\kappa_{\min}}\frac{\lambda_j}{\mu_i}\}$$

Proof. Let y be an arbitrary unit vector in $\mathcal{Y}$, with $y = u+v$, where $u \in \mathcal{X}$ and $v \in \mathcal{X}_\perp$, with $\|u\|_2^2 + \|v\|_2^2 = 1$. By using the A-orthogonality of u, v, and positive definiteness, we have

$$y^T Ay = u^T Au + v^T Av \geq u^T Au \geq \|u\|^2 \lambda_i$$

By definition, we have $y^T By \leq \mu_j$, and using Lemma 2.1.6 we have

$$\kappa_{\max} \geq \frac{y^T Ay}{y^T By} \geq \frac{\|u\|^2 \lambda_i}{\mu_j}. \tag{6.1}$$

Now let x' denote $u/\|u\|_2$. It is easy to see that

$$x' = \arg\max_{x \in \mathcal{X}} x^T y$$

and that $\|u\|_2^2 = (x'^T y)^2$. Combining this with equation 6.1 proves the first inequality. The second inequality follows from the first by interchanging the roles of A and B and noting that $\lambda_{\max}(B, A) = 1/\lambda_{\min}(A, B)$. $\square$

6.2 Optimality of the bounds

In this Section we study the optimality of the bound in 6.1.3 with respect to the ratio λ_i/μ_j. In particular, we want to establish optimality for *varying* values of the ratio when the generalized eigenvalues $\kappa_{\min}, \kappa_{\max}$ are *fixed*. Consider the following two positive definite matrices:

$$\begin{aligned} A &= \begin{pmatrix} 1/n & 0 \\ 0 & 1 \end{pmatrix} \qquad (6.2) \\ B &= (1+1/n)^{-1} \begin{pmatrix} 1 & -1/\sqrt{n} \\ 1/\sqrt{n} & 1 \end{pmatrix} \begin{pmatrix} 1/n & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1/\sqrt{n} \\ 1/\sqrt{n} & 1 \end{pmatrix}^T \end{aligned}$$

It is not hard to verify that $\lambda_{\max}(A, B), \lambda_{\max}(B, A) < 4$. On the other hand, the vectors $x = [1, 0]^T$ and $y = (1 + 1/n)^{1/2}[-1/\sqrt{n}, 1]^T$ are normalized eigenvectors of A and B respectively, corresponding to eigenvalues $1/n$ and 1. The bound of 6.1.3 gives $|x_i^T y_j| \leq O(1/\sqrt{n})$, which is within a constant of the actual inner product. So, for general positive matrices the bound is optimal up to a constant with respect to the ratio of the eigenvalues. Does this remain true if we restrict the pair (A, B) to be Laplacians? We show that the answer is positive.

6.2.1 Graph definitions - the pair (A,B)

We will demonstrate the optimality of the bound with a pair of Laplacians (A, B) which we define here. Let A_n be the cycle of n nodes, where n is a multiple of 4. The vertices in A_n are ordered in the natural way; consecutive vertices take consecutive numbers. Let R be the $n \times (n/2)$ matrix, with all its elements equal to zero, with the exception $R_{2j-1,j} = R_{2j,j} = 1$ for each $1 \leq j \leq n/2$. It is not hard to verify that $Q = R^T A R$ is the Laplacian of $C_{n/2}$. We now define a Laplacian S on $3n/2$ vertices

$$S = \begin{pmatrix} 2I & -2R \\ -2R^T & Q + 4I \end{pmatrix}.$$

The graph S is the support graph constructed in section 5.1. The Schur complement of S with respect to $Q + 4I$ is -as noted in Section 2.4.5- a Laplacian and it is given by $B = 2I - 2R(Q + 4I)^{-1}(2R)^T$. This will be the second graph B. It is interesting to note that the matrix B has no zero entries. From the discussion in section 5.1 we have

$$\lambda_{\max}(A, B) \leq 3 \text{ and } \lambda_{\max}(B, A) \leq 5. \tag{6.3}$$

6.2.2 Eigenvalues and eigenspaces of A, B

Eigenvectors and eigenvalues of B. It is easy to see that 2 is an eigenvalue of B with multiplicity $n/2$. The corresponding eigenspace is $\mathcal{N}(R^T)$.

Eigenvectors and eigenvalues of A. Let n be a multiple of 4, and C_n be the Laplacian of the cycle of size n, with eigenvalues $\lambda_0 \leq \ldots \leq \lambda_{n-1}$. Let $\omega = \exp(2\pi i/n)$ be the n^{th} primitive root of unity. The Fourier transform matrix is given by $F_{jk} = w^{jk}$, for $j, k \in [0, n-1]$. It is known and easy to show that F diagonalizes any circulant matrix of size n (for instance, see [Big94]). In particular, F diagonalizes C_n. The simple eigenvalues of C_n are $0, 4$ with corresponding eigenvectors $F_0, F_{n/2}$. All the other eigenvalues have multiplicity 2, and the j^{th} pair corresponds to the eigenspace generated by F_j, F_{n-j}. In the rest of this section we will let $\mu_j = \lambda_{2j-1} = \lambda_{2j}$. It can be verified that

$$\mu_j = 2 - \omega^j - \omega^{-j} = 2 - 2Re\left(\exp(2\pi i/n)\right) = 2 - 2\cos(2\pi j/n) = 4\sin^2(\pi j/n)$$

Let $S_j = (F_{n-j} + F_j)/2$ and $S_{n-j} = i(F_{n-j} - F_j)/2$, for $1 \leq j \leq n/2 - 1$. The following expressions can be easily verified.

$$\begin{aligned} S_j(k) &= \cos(2\pi kj/n) \qquad 0 \leq k \leq n-1 \\ S_{n-j}(k) &= \sin(2\pi kj/n) \end{aligned}$$

Using the orthogonality of the F_j's, we have $S_j^T S_{n-j} = 0$, and thus

$$span(\{F_j, F_{n-j}\}) = span\left(\{S_j, S_{n-j}\}\right)$$

Hence the vectors S_j, S_{n-j} are eigenvectors of A.

Expressing eigenspaces of B in terms of those of A. Let $\mathcal{H} = Null(R)$ be the subspace of $\mathbb{R}^n$ consisting of vectors w such that $w(2k) = -w(2k+1)$ for $1 \leq k \leq n/2$. Obviously, the dimension of H is $n/2$. Let $2H_{n/2} = F_{n/2}$, and

$$\begin{aligned} 2H_j &= \mu_j^{1/2} S_j - \mu_{n/2-j}^{1/2} S_{n/2+j} \qquad 1 \leq j \leq n/4 \\ 2H_{n/4+j} &= \mu_j^{1/2} S_{n-j} - \mu_{n/2-j}^{1/2} S_{n/2-j} \qquad 1 \leq j < n/4 \end{aligned}$$

We claim that the vectors H_j, $0 \leq j \leq n/2$ form an orthogonal basis for $\mathcal{H}$. Orthogonality follows from the orthogonality of the S_j's. It remains to show that for each j, t we have $H_j(2t) = -H_j(2t+1)$. It is easy to check this for $j = n/2$. Now, note that $\mu_j = \sin(\pi j/n)$ and $\mu_{n/2-j}^{1/2} = \sin(\pi/2 - \pi j/n) = \cos(\pi j/n)$. For $1 \leq j \leq n/4$, and using the trigonometric identity $\sin(a+b) = \sin a \cos b + \cos a \sin b$ we have

$$\begin{aligned} H_j(2t) &= \sin(\frac{\pi j}{n})\cos(\frac{4\pi jt}{n}) - \cos(\frac{\pi j}{n})\sin(\frac{4\pi(n/2-j)t}{n}) \\ &= \sin(\frac{\pi j}{n})\cos(\frac{4\pi jt}{n}) + \cos(\frac{\pi j}{n})\sin(\frac{4\pi jt}{n}) \\ &= \sin(\frac{4\pi jt + \pi j}{n}) \end{aligned}$$

and

$$\begin{aligned} H_j(2t+1) &= \sin(\frac{\pi j}{n})\cos(\frac{2\pi j}{n}2t + \frac{2\pi j}{n}) - \cos(\frac{\pi j}{n})\sin(\frac{4\pi(n/2-j)t}{n} + \frac{2\pi(n/2-j)}{n}) \\ &= \sin(\frac{\pi j}{n})\cos(\frac{2\pi j}{n}2t + \frac{2\pi j}{n}) - \cos(\frac{\pi j}{n})\sin(\pi - \frac{2\pi j}{n}2t - \frac{2\pi j}{n}) \\ &= \sin(\frac{\pi j}{n})\cos(-\frac{2\pi j}{n}2t - \frac{2\pi j}{n}) + \cos(\frac{\pi j}{n})\sin(-\frac{2\pi j}{n}2t - \frac{2\pi j}{n}) \\ &= \sin(-\frac{4\pi jt + \pi j}{n}) \end{aligned}$$

The same property can be shown similarly for the vectors H_j for $n/4 \leq j \leq n/2$. We are now ready to state the main result of this section.

Theorem 6.2.1. *Theorem 6.1.3 is optimal up to a factor of 5.*

Proof. We take the Laplacians A and B as defined above. We may without loss of generality assume that the vectors S_j have unit norm. Let $x = S_j$ and $y = 2H_i / \|2H_i\|_2$. The vectors x, y are eigenvectors corresponding to eigenvalues μ_j and 2 respectively. We have

$$(x^T y)^2 = \mu_i / \|2H_1\|_2^2 = \frac{2}{\|2H_j\|_2^2}\frac{\mu_i}{2}.$$

Using equation 6.3, Theorem 6.1.3 gives

$$(x^T y)^2 \leq \lambda_{\max}(B, A)\frac{\mu_i}{2} \leq 5\frac{\mu_i}{2}$$

The Theorem follows by noting that $\|2H_j\|_2^2 < 2$. $\square$

6.2.3 The eigenvalues of (A^2, B^2) - and some questions

Although one can use the analytic expressions for the eigenvalues and eigenvectors of A^2, B^2 to verify directly that $\lambda_{\max}(A^2, B^2) = \Theta(n^2)$, the fact that the maximum generalized eigenvalue of (A^2, B^2) is unbounded can be viewed as a consequence of the optimality of the bound for (A, B), as the following corollary shows.

Corollary 6.2.2. *We have* $\lambda_{\max}(A^2, B^2) = \Omega(n^2)$.

Proof. As in the proof of Theorem 6.2.1, let $x = S_1$ and $y = 2H_1 / \|2H_1\|_2$. The vectors x, y are eigenvectors corresponding to eigenvalues $\mu_1 = O(1/n^2)$ and $1/2$ respectively. Theorem 6.1.3 applied to the pair (A^2, B^2) gives

$$(x^T y)^2 \leq \lambda(B^2, A^2) 4{\mu_i}^2$$

However, we already know that $(x^T y)^2 \geq \mu_i/2$. This implies that $\lambda(B^2, A^2) \geq \mu_i^{-1}/8$. □

It is interesting to observe the difference between the example in equation 6.2 and the proof of Theorem 6.2.1. The optimality of the bound for general positive matrices can be shown via a family of 2×2 times controlling the size of the smallest eigenvalue $1/n$. We haven't been able to do the same for Laplacians, and because of that we used graphs of increasing size to show the optimality.

We pose two related questions. Let A, B be graphs on n. Let D_A be the diagonal matrix containing the vertex volumes of A, and let $\tilde{A} = D_A^{-1/2} A D_A^{-1/2}$, $\tilde{B} = D_A^{-1/2} B D_A^{-1/2}$.

Question 1: Is the bound of Theorem 6.1.3 for the pair $(\tilde{A}, \tilde{B})$ optimal for all possible values of the ratio λ_i/μ_j for a given n ?

Question 2: Is the maximum eigenvalue of the pair $(\tilde{A}, \tilde{B})$ independent from $\lambda_{\min}(\tilde{A})$ and only a function of n and $\kappa(\tilde{A}, \tilde{B})$?

In Chapter 7, we will see that these questions are related to the V-cycle multigrid algorithms.

6.3 Spectral inequalities for multiway cuts

Let $\mathcal{P}$ be an edge separator, i.e. a partitioning of a given graph A into m disjoint clusters of vertices $P_i, i = 1 \ldots, m$. In this section we denote the normalized Laplacian of A by

$\hat{A}$. To state the inequality, we need to define several auxiliary graphs and matrices that are derived from A and $\mathcal{P}$.

(i). The cluster membership matrix R. Column j of R corresponds to cluster j and $R(i,j) = 1$ if and only if node i of A is contained in cluster j.

(ii). The quotient graph Q. Q has one node per cluster of A, and the weight $Q(i,j)$ is equal to the total capacity between clusters i and j in G. Algebraically, it is not hard to verify that

$$Q = R^T A R. \tag{6.4}$$

(iii). The Steiner graph S. We construct S as follows. If cluster i contains m_i nodes, we attach m_i leaves to the i^{th} node of Q. The weights of the new edges are equal to the degrees of the corresponding nodes in G. So, S has n leaves and m internal vertices. Algebraically,

$$S = \begin{pmatrix} D & -V \\ -V^T & Q + D_Q \end{pmatrix}$$

where D are the volumes of the vertices of A, $V = DR$, and $D_Q = R^T D R$.

(iv). The approximation graph B. We define B as the Schur complement of S with respect to the elimination of the internal vertices.

$$B = D - V(Q + D_Q)^{-1} V^T.$$

(v). The normalized approximation $\hat{B}$. We let

$$\hat{B} = D^{-1/2} B D^{-1/2} = I - D^{1/2} R (Q + D_Q)^{-1} R^T D^{1/2}. \tag{6.5}$$

We are now ready to give a spectral characterization for (ϕ, γ)-decompositions that we defined and discussed in Section 5.5. For completeness, we re-define the decomposition with a slightly different terminology here. Let P be an edge separator that disconnects the vertex set V into disjoint sets V_i. For all i and $v \in P_i$, we denote by $out(v)[in(v)]$ the total weight of the edges incident to v that leave [stay inside] P_i. We define the *local separation* $\gamma_{\mathcal{P}}$ of $\mathcal{P}$, as the least number γ for which all $v \in G$ satisfy $\gamma in(v) \geq out(v)$. Let A_i denote the graph induced by the vertices in P_i and let ϕ_i be its conductance. We define the *local conductance* as $\phi_{\mathcal{P}} = \min_i \phi_i$.

Theorem 6.3.1. *Let $\mathcal{P}$ be a an edge separator with local conductance $\phi_{\mathcal{P}}$ and local separation $\gamma_{\mathcal{P}}$. Let y be any vector in $\mathcal{N}(R^T D^{1/2})$, and x be any unit vector which is a linear combination of vectors of $\hat{A}$ corresponding to eigenvalues smaller than λ_i. We have*

$$(x^T y)^2 \leq 16 \gamma \phi^2 \lambda_i.$$

Hence there is a unit vector $z \in \mathcal{R}(D^{1/2}R)$ such that

$$(x^T z)^2 \geq 1 - 16\gamma\phi^2\lambda_i.$$

Proof. An application of Theorem 5.2.2 in combination with Theorem 5.3.1 to the pair of graphs (A, B) gives $\lambda_{\max}(B, A) \leq 16\gamma\phi^2$. By Lemma 2.1.6 we have $\lambda_{\max}(B, A) = \lambda_{\max}(\hat{B}, \hat{A})$. Note that $\mathcal{N}(R^T D^{1/2})$ is an eigenspace of $\hat{B}$ with eigenvalue 1. Then, applying Theorem 6.1.3 to $(\hat{B}, \hat{A})$ gives

$$(x^T y)^2 \leq \lambda_{\max}(B, A)\lambda_i \leq 16\gamma\phi^2\lambda_i$$

Note now that if y is the projection of x into $\mathcal{N}(R^T D^{1/2})$ and z is its projection into $\mathcal{R}(D^{1/2}R)$, we have $x = y + z$, with $y^T z = 0$. From this, we get $\|z\|^2 = (x^T z)^2$ and $\|y\|^2 = (x^T y)^2$. Since $\|z\|^2 + \|y\|^2 = 1$ the second claim follows. □

Chapter 7

Multigrid algorithms: A combinatorial approach

The algorithms of Spielman and Teng [ST04], as well as our planar solver presented in Chapter 4 are based on the recursive preconditioned Chebyshev algorithm. The choice of the Chebyshev iteration over Richardson's iteration is motivated by the much faster convergence of the former with respect to the condition number of the system. Given the superior properties of the Chebyshev iteration it seems that the analysis of the recursive Richardson's iteration could at best be left as an interesting exercise. We show that it is much more.

Throughout this Chapter for all symmetric matrices B we will define B^{-1} to be any matrix C that satisfies $CBx = BCx = x$ for all $x \in \mathcal{R}(B)$. In Section 6.3 we undertook an analysis of matrices that are involved in the two-level scheme; the Steiner graph S, the quotient Q, and the preconditioner B, along with its normalization $\hat{B}_+$. However, to analyze Richardson's method preconditioned by B we need to understand the properties of the matrix $I - B^{-1}A$, which leads us to seek an expression for B^{-1}. Using the notation of Section 6.3, Gremban [Gre96] observed that if x and y satisfy

$$S \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

then x satisfies $Bx = b$. The Cholesky factorization of S gives

$$\begin{pmatrix} D & -V \\ -V^T & Q + D_Q \end{pmatrix} = \begin{pmatrix} I & 0 \\ -R^T & I \end{pmatrix} \begin{pmatrix} D & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} I & -R \\ 0 & I \end{pmatrix} \Rightarrow \tag{7.1}$$

$$\begin{pmatrix} D & -V \\ -V^T & Q + D_Q \end{pmatrix}^{-1} = \begin{pmatrix} I & R \\ 0 & I \end{pmatrix} \begin{pmatrix} D^{-1} & 0 \\ 0 & Q^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ R^T & I \end{pmatrix} \tag{7.2}$$

Using the expression for the inverse of S we get that for all b we have

$$x = (D^{-1} + RQ^{-1}R^T)b.$$

Since for all b we have $x = B^{-1}b$, we get

$$B^{-1} = D^{-1} + RQ^{-1}R^T.$$

We then observe that for the normalization $\hat{B}_+ = D^{-1/2}BD^{-1/2}$ we have

$$\hat{B}_+^{-1} = D^{1/2}B^{-1}D^{1/2} = I + D^{1/2}RQ^{-1}R^TD^{1/2}.$$

The expression for the inverse of the normalized preconditioner comes strikingly close to one of the fundamental operators in multigrid analysis, the two level operator, a form of which we already have seen in equation 2.9: $M = I - R_{project}A_{n/2}^{-1}R_{project}^T$. This observation, along with the fact that $\kappa(A, B) = \kappa(\hat{A}, \hat{B}_+)$ where $\hat{A}$ is the normalized Laplacian, suggests a combinatorial approach for constructing as well for analyzing algebraic multigrid for Laplacians; the construction of the second level graph is reduced to the construction of a Steiner preconditioner, which makes the analysis of the two-level operator amenable to support theory tools.

We obtain an exact characterization of the two-level error in terms of the condition number $\kappa(A, B)$. We show that for Steiner preconditioners that are constructed from *edge separators*, a bounded $\kappa(A, B)$ is not a sufficiently strong property to guarantee the convergence of the multigrid V-cycle, precisely because of the tightness of the perturbation bounds of Chapter 6. On the positive side, we use the progress in the analysis of the two-level error to show that fast convergence is possible for more complicated multigrid cycles for certain model problems, such as those we considered in Chapter 5. We then introduce a stronger notion of graph approximation, the condition number $\kappa(\hat{A}^2, \hat{B}^2)$, where $\hat{A}, \hat{B}$ are normalized versions of A, B, and we show that it guarantees convergence of the V-cycle. Furthermore, driven by this new graph approximation measure, we propose Steiner preconditioners that are based on *vertex separators* on a properly modified linear system, and have the stronger condition number bounded at least in a local sense.

7.1 ResidualCorrection: A general framework

In this section we review general tools that will be helpful in the analysis of multigrid. Our review is based on the presentation of [TSO00].

Our goal is to solve the linear system $Ax = b$, where A is a symmetric matrix. In this section we will assume that we are given another symmetric matrix B^{-1}, of which we can think as an "approximation" of A^{-1}. We consider the following general iteration for approximating the solution of the system.

$$\boxed{\begin{array}{l}\mathbf{ResidualCorrection}(A, M, b, \nu, x_1) := \\ r = b - Ax_m;\ x_{m+1} = x_m + B^{-1}r;\ m = 1, \ldots, \nu - 1.\end{array}}$$

This iteration obtains a new approximation by correcting the current iterate by $B^{-1}r$ where r is the residual. When $B = I$ the iteration is equivalent to Richardson's iteration. Alternatively, if $M = I - B^{-1}A$, we equivalently have

$$\boxed{\begin{array}{l}\mathbf{ResidualCorrection}(A, M, b, \nu, x_1) := \\ x_{m+1} = Mx_m + B^{-1}b;\ m = 1, \ldots, \nu - 1.\end{array}}$$

We will call M is the *iteration operator*. It can be seen that the effect of each iteration on the error e_m is given by the following equation, which actually provides yet another alternative definition.

$$\boxed{\begin{array}{l}\mathbf{ResidualCorrection}(A, M, b, \nu, x_1) := \\ e_{m+1} = Me_m;\ m = 1, \ldots, \nu - 1.\end{array}}$$

For the composition of instances of **ResidualCorrection** (**RC** for short), we have

$$\begin{aligned}\mathbf{RC}(A, M, b, 1, \mathbf{RC}(A, M, b, \nu, x_1)) &= \mathbf{RC}(A, M^\nu, b, 1, x_1) \\ \mathbf{RC}(A, M_2, b, 1, \mathbf{RC}(A, M_1, b, 1, x_1)) &= \mathbf{RC}(A, M_2M_1, b, 1, x_1)\end{aligned}$$

Note that if $x_1 = 0$, we have $x_2 = (I - M)A^{-1}b$. Using the composition properties of **RC** we also have $x_{\nu+1} = (I - M^\nu)A^{-1}b$. When we know that $x_1 = 0$, we will call $(I - M^\nu)A$ the *approximate inverse* of A. The rate of convergence of **RC** is characterized by the spectral radius $\rho(M)$ of M, which is the maximum over the absolute values of the eigenvalues of M.

We conclude this Section with a Lemma useful for the calculation of the operator of a given iteration.

Lemma 7.1.1. *If an instance of* **RC** *is such that for all b the iteration satisfies $x_2 = (I - M)A^{-1}b$ when $x_1 = 0$, the matrix M is the iteration operator.*

Proof. Let M' be the iteration operator. By definition, we have $e_2 = M'e_1$ for all error vectors e_1. When $x_1 = 0$, we have $e_1 = A^{-1}b$. Since $x_2 = (I - M)A^{-1}b$ for all b, we have $e_2 = Me_1$ for all e_1. Hence $M' = M$. □

7.1.1 Simple transformations are ResidualCorrection

Obviously a direct solver can be seen as **RC** with $B = A$. Let $A = G^T C G$ where G is an invertible matrix. Let M be the iterator operator of an instance of **RC** for C. Consider the following iteration for the solution of the system $Ax = b$.

$$x_{new} = G^{-1} \cdot \mathbf{RC}(C, M, G^{-T}b, 1, Gx_{old})$$

By definition, we know that there is matrix B^{-1} such that

$$\begin{aligned} x_{new} &= G^{-1} \cdot \mathbf{RC}(C, M, G^{-T}b, 1, Gx_{old}) \\ &= x_{old} + G^{-1}B^{-1}(G^{-T}b - CGx_{old}) \\ &= x_{old} + G^{-1}B^{-1}(G^{-T}b - G^{-T}AG^{-1}Gx_{old}) \\ &= x_{old} + G^{-1}B^{-1}G^{-T}(b - Ax_{old}). \end{aligned}$$

The last equality shows that the proposed iteration for A is also an instance of **RC**. Using the properties of **RC**, it is not hard to see that if $x_{old} = 0$, we have $x_{new} = G^{-1}(I - M)C^{-1}G^{-T}b = (I - G^{-1}MG)A^{-1}b$. By Lemma 7.1.1, the iteration operator is $M' = G^{-1}MG$. Since M' is a similarity transformation of M we have $\rho(M') = \rho(M)$.

We now consider what we will call *partial Cholesky expansion* . Let

$$C = G^T \begin{pmatrix} A & 0 \\ 0 & D \end{pmatrix} G$$

where D is a diagonal matrix, G is invertible and A, C are positive definite. Assume that we have a **RC** procedure for C, with iterator operator M, such that MC^{-1} is symmetric. Let the dimensions of A, C be n and m respectively. Let Π be the $m \times n$ matrix which has the $n \times n$ identity in its top left corner, and zero everywhere else. Note that $\Pi^T\Pi = I_n$. Consider the following iteration for the solution of the system $Ax = b$.

$$x_{new} = \Pi^T G \cdot \mathbf{RC}(C, M, G^T\Pi b, 1, G^{-1}\Pi x_{old})$$

By definition, we know that there is matrix B^{-1} such that

$$\begin{aligned} x_{new} &= \Pi^T G \cdot \mathbf{RC}(C, M, G^T\Pi b, 1, G^{-1}\Pi x_{old}) \\ &= x_{old} + \Pi^T G B^{-1}(G^T\Pi b - CG^{-1}\Pi x_{old}) \\ &= x_{old} + \Pi^T G B^{-1}G^T\Pi(b - Ax_{old}). \end{aligned}$$

The last equality shows that the proposed iteration for A is also an instance of **RC**. In the case where $x_{old} = 0$, we have

$$\begin{aligned} x_{new} &= \Pi^T G(I-M)G^{-1} \begin{pmatrix} A^{-1} & 0 \\ 0 & D^{-1} \end{pmatrix} \Pi b \\ &= \Pi^T I_m \begin{pmatrix} A^{-1}b \\ 0 \end{pmatrix} - \Pi^T GMG^{-1} \begin{pmatrix} A^{-1}b \\ 0 \end{pmatrix} \\ &= (I - \Pi^T GMG^{-1}\Pi)A^{-1}b \end{aligned}$$

Hence, by Lemma 7.1.1 the operator for the proposed iteration is $M' = \Pi^T GMG^{-1}\Pi$. We have

$$M'A^{-1} = \Pi^T GMC^{-1}CG^{-1}\Pi A^{-1} = \Pi^T GMC^{-1}G^T \begin{pmatrix} A & 0 \\ 0 & I \end{pmatrix} \Pi A^{-1} = \Pi^T GMC^{-1}G^T\Pi.$$

By assumption MC^{-1} is symmetric, so $M'A^{-1}$ is symmetric. Using also the last equation, we have

$$\begin{aligned} \rho(M') &= \max|\lambda(M'A^{-1}, A^{-1})| = \max_x \left| \frac{x^T M'A^{-1}x}{x^T A^{-1}x} \right| \\ &= \max_x \left| \frac{x^T \Pi^T GMC^{-1}G^T \Pi x}{x^T \Pi^T GB^{-1}G^T \Pi x} \right| \\ &\leq \max_y \left| \frac{y^T MC^{-1}y}{y^T C^{-1}y} \right| = \rho(M). \end{aligned}$$

7.2 The multigrid algorithm

Given the the system $Ax = b$ the graph $A = (V, E, w)$, our goal is to describe an instance of **RC** with an iteration operator M and study the spectral radius of M.

7.2.1 The hierarchy of graphs

The algorithm will operate on a hierarchy of graph Laplacians $\mathcal{H}(A) = \{A_i, A_i^o, S_i, B_i\}$, $i = 1, \ldots, r$, where each two consecutive levels satisfy the following requirements:

1. The graph A_i^o is constructed by substituting each edge $e \in E^o \subseteq E$ by two edges of weight $2w(e)$. Hence A_i^o is a partial Cholesky expansion of A_i, where

$$A_i^o = G^T \begin{pmatrix} A_i & 0 \\ 0 & D \end{pmatrix} G$$

for some diagonal D. We also let Π be the associated projection matrix defined in section 7.1.1. In general E^o may be an empty set, in which case $A_i^o = A_i$.

2. S_i is a Steiner graph for A_i^o, with a set of $n+m$ vertices, n vertices corresponding to the vertices of A_i^o and $m < n$ Steiner vertices. B_i is the Schur complement of S_i with respect to the elimination of the Steiner vertices.

3. The graph A_{i+1} has $m < n$ vertices, and $A_{i+1} = R^T A_i^o R$, where R is the $n \times m$ **restriction** matrix. We call A_{i+1} the **quotient** graph of A_i. We also require that A_{i+1} is the Schur complement of A_i^o with respect to the elimination of the non-Steiner vertices of A_i^o and that $B_i^{-1} = D^{-1} + RA_{i+1}^{-1}R^T$ where D is the diagonal of A_i.

We will denote by $\mu_{\mathcal{H}}$ the *size reduction factor* $\max_i |A_i|/|A_{i+1}|$, and by $\tau_{\mathcal{H}} = (4\kappa \ln \kappa)^{1/2}$, $\kappa_{\mathcal{H}} = \max_i \lambda_{\max}(A_i, B_i)$ the *hierarchy condition*.

For any fixed i, we will use the following matrices and notation: $D = 2diag(A_i^o)$, the **normalized Laplacian** $\hat{A} = D^{-1/2} A_i^o D^{-1/2}$, the **quotient** $Q = A_{i+1}$, the **normalized preconditioner** $\hat{B}_+$ with $\hat{B}_+^{-1} = 2I + D^{1/2} R Q^{-1} R^T D^{1/2}$ and $\hat{B}^{-1} = D^{1/2} R Q^{-1} R^T D^{1/2}$. A key role in the multigrid analysis is played by the eigenvalue decomposition of $\hat{B}^{-1}\hat{A}$.

Lemma 7.2.1. *The eigenvalues of* $I - \hat{B}^{-1}\hat{A}$ *are*
(i) 1, *with corresponding eigenspace* $\mathcal{N}(R^T D^{1/2} \hat{A})$, *and*
(ii) 0, *with corresponding eigenspace* $\mathcal{R}(D^{1/2} R)$.

Proof. By the fundamental theorem of linear algebra, $\mathcal{N}(R^T D^{1/2})$ and $\mathcal{R}(D^{1/2} R)$ are orthogonal and span $\mathbb{R}^n$. That is, $(q, u) = 0$ whenever $q \in \mathcal{R}(D^{1/2} R)$, and $R^T D^{1/2} u = 0$. Equivalently, $(q, \hat{A}u) = 0$ whenever $q \in \mathcal{R}(D^{1/2} R)$ and $R^T D^{1/2} \hat{A} u = 0$. This means that $\mathcal{R}(D^{1/2} R)$ and $\mathcal{N}(R^T D^{1/2} \hat{A})$ are $\hat{A}$-orthogonal, and thus they also span $\mathbb{R}^n$. Part (i), holds by definition. For part (ii), let $y = D^{1/2} R w$. By using the algebraic definition of the quotient we have

$$\begin{aligned} \hat{B}^{-1}\hat{A} y &= D^{1/2} R Q^{-1} R^T D^{1/2} D^{-1/2} A D^{-1/2} D^{1/2} R w \\ &= D^{1/2} R Q^{-1} R^T A R w \\ &= D^{1/2} R w = y. \end{aligned}$$

□

Corollary 7.2.2. *The eigenvalues of* $I - \hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2}$ *are*
(i) 1, *with corresponding eigenspace* $\mathcal{N}(R^T D^{1/2} \hat{A}^{1/2})$, *and*
(ii) 0, *with corresponding eigenspace* $\mathcal{R}(\hat{A}^{1/2} D^{1/2} R)$.

7.2.2 The two-level scheme

We first state a two-level scheme.

	$\mathbf{MG_k}(A_i, b, x, \mathcal{H}) :=$
0.	If $i = r$ return $x = A_i^{-1}b$;
1.	Let $b' = G^T \Pi b$;
2.	Return $x = \Pi^T G \mathbf{MG_k^o}(A_i^o, b', G^{-1}\Pi x, \mathcal{H})$;

where,

	$\mathbf{MG_k^o}(A_i^o, b, x, \mathcal{H}) :=$
1.	$Q = A_{i+1} = R^T A_i^o R$;
	$D = 2diagonal(A^o)$;
	$\hat{A} = D^{-1/2} A_i^o D^{-1/2}$;
	$\hat{B}^{-1} = D^{1/2} R Q^{-1} R^T D^{1/2}$;
2.	$z = D^{1/2}x$;
3.	Repeat t times $z := (I - \hat{A})z + D^{-1/2}b$;
4.	$r := D^{-1/2}b - \hat{A}z$;
5.	$z := z + \hat{B}^{-1}r$;
6.	Repeat t times $z := (I - \hat{A})z + D^{-1/2}b$;
7.	Return $x = D^{-1/2}z$.

The reader can think of t as a small function of the dimension of A. Steps $3 - 6$ in $\mathbf{MG_k^o}$ solve the normalized system $\hat{A}z = D^{-1/2}b$ and consist of three different instances of **RC** for $\hat{A}$. Steps 3 and 6 are Richardson's iterations, while steps $3 - 4$, implement a step preconditioned with $\hat{B}$. Steps 2 and 7 can be seen to be a simple change of variables, as discussed in the previous section. Using the properties of composition of instances of **RC** we can fully describe the iteration matrix of the two-level $\mathbf{MG_k^o}$.

$$M_{k,k+1}^o = D^{-1/2}(I - \hat{A})^t (I - \hat{B}^{-1}\hat{A})(I - \hat{A})^t D^{1/2}. \tag{7.3}$$

7.2.3 Recursion

Step 5 in the two level $\mathbf{MG_k^o}$ is in general expensive, and the natural idea is to try recursion. In particular, we will replace step 5 with

5.	Let $w = 0$;
	Repeat τ times $w = \mathbf{MG_{k+1}}(A_{k+1}, R^T D^{1/2} r, w, \mathcal{H})$;
	$z := z + cD^{1/2}Rw$;

Theorem 7.2.3. $\mathbf{MG_k}$ *and* $\mathbf{MG_k^o}$ *are instances of* **RC** *with iteration operators* M_k *and* M_k^o *respectively, such that* $M_k A_k^{-1}$ *and* $M_k^o {A_k^o}^{-1}$ *are symmetric, and* $\rho(M_k) \leq \rho(M_k^o)$.

Proof. We aim now to prove that $\mathbf{MG_k}$ is an instance of **RC**, for which it is enough to show that there is an iteration operator M_k. We use induction. The inductive hypothesis is that (i) $\mathbf{MG_{k+1}}$ is an instance of **RC**, hence it has an iteration operator M_{k+1}, and (ii) $M_{k+1}A_{k+1}^{-1}$ is symmetric. At level r of the recursion, we solve the system exactly and thus $M_r = 0$. Consider now the two-level operator for $\mathbf{MG_k^o}$ given in equation 7.3. The key observation is that we do not have the exact inversion $A_{k+1}^{-1}R^T D^{1/2} r$, but rather the approximate inverse, which by the inductive hypothesis is given by

$$(I - M_{k+1}^{\tau})A_{k+1}^{-1}R^T D^{1/2} r$$

Then, by substituting in equation 7.3 we have

$$\begin{aligned} M_k^o &= D^{-1/2}(I-\hat{A})^t(I - D^{1/2}R(I - M_{k+1}^{\tau})A_{k+1}^{-1}R^T D^{1/2}\hat{A})(I-\hat{A})^t D^{1/2} \\ &= M_{k,k+1}^o + D^{-1/2}(I-\hat{A})^t D^{1/2} R M_{k+1}^{\tau} A_{k+1}^{-1} R^T D^{1/2}\hat{A}(I-\hat{A})^t D^{1/2}. \quad (7.4) \end{aligned}$$

We have ${A_k^o}^{-1} = D^{-1/2}\hat{A}^{-1}D^{-1/2}$. Using this and the inductive hypothesis that $M_{k+1}A_{k+1}^{-1}$, it is easy to see that $M_k^o {A_k^o}^{-1}$ is also symmetric. The definition of $\mathbf{MG_k}$ is a partial Cholesky expansion instance as described in section 7.1, hence it is an instance of **RC** and it has an iteration operator M_k. Given that $M_k^o {A_k^o}^{-1}$ is symmetric, the discussion in section 7.1 shows directly that $M_k A_k^{-1}$ is also symmetric, and that $\rho(M_k) \leq \rho(M_k^o)$. □

Remarks. When $\tau = 1$ the algorithm is known as the V-cycle, and when $\tau = 2$ as the W-cycle. The V-cycle is quite interesting from a complexity perspective, for two reasons: (i) The parallel complexity of the algorithm is $O(r \log n)$, (ii) More importantly, the total work is at most $O(tr|A|)$ where r is the number of levels. As we will see in more detail later, for all other values of τ, the complexity of the algorithm is uncontrollable unless the size of the graphs in the hierarchy decreases sufficiently fast. The need for this geometric decrease is the major problem in the nearly-linear time algorithms of Spielman and Teng.

7.3 Multigrid convergence analysis

The complexity analysis of the recursive preconditioned Chebyshev method, presented in Chapter 4, is based on a uniform upper bound on $\kappa(\hat{A}, \hat{B}_+)$ for every two levels, along with a sufficiently fast geometric decrease in the size of the graphs in the hierarchy. We

show that the *weak* uniform upper bound on $\kappa(\hat{A}, \hat{B}_+)$ is *not* sufficient to guarantee fast convergence of the V-cycle, in the sense that increasing the number of smoothings t, essentially does not improve the effectiveness of the two-level algorithm. We will show that the underlying reason is the tightness of the spectral perturbation bounds for of Chapter 6 for $(\hat{A}, \hat{B}_+)$. This led us to seek stronger spectral inequalities between the subspaces of $(\hat{A}, \hat{B}_+)$. Such inequalities are implied by a *strong* uniform upper bound on $\kappa(\hat{A}^2, \hat{B}_+^2)$ -which as we saw in Chapter 6 is not implied by the $\kappa(\hat{A}, \hat{B}_+)$ bound. Indeed, we show that a uniform $\kappa(\hat{A}^2, \hat{B}_+^2)$ bound guarantees the convergence of the V-cycle with only one smoothing.

The presentation in this Section is inspired from the analysis in [McC84]. Analogues of both the uniform $\kappa(\hat{A}, \hat{B}_+)$ and $\kappa(\hat{A}^2, \hat{B}_+^2)$ bounds have been used in the multigrid literature, for example the M_1 and M_2 measures in [BCF$^+$00, CFH$^+$03]. A variant of M_2 was used in [McC84] to prove the optimal convergence of multigrid under the full elliptic regularity assumption. Our contribution is the adjustment of known convergence results in the context of our Steiner graph framework for multigrid.

7.3.1 Some Lemmas

Equation 7.4 expresses the iteration operator M_k as a sum of two terms, one related to the two-level method and one recursive term. This along with the fact that $\rho(M_k) \leq \rho(M_k^o)$, enables the use of inductive arguments for the analysis of $\rho(M_k^o)$. In the following, to simplify notation, we will denote M_k^o by M_k.

To make our presentation easier for readers that are accustomed to thinking in terms of the Euclidean norm we will deviate from the usual multigrid notation and analyze the spectral norm of the symmetric matrix $\hat{A}^{1/2} D^{1/2} M_k D^{-1/2} \hat{A}^{-1/2}$. We note that, using the symmetry of the involved matrices we have

$$\begin{aligned}
\rho^2(\hat{A}^{1/2} D^{1/2} M_k D^{-1/2} \hat{A}^{-1/2}) &= \rho(\hat{A}^{-1/2} D^{-1/2} M_k A M_k D^{-1/2} \hat{A}^{-1/2}) \\
&= \rho(M_k A M_k A^{-1}) && \text{[by Lemma 2.1.4]} \\
&= \rho(A^{-1/2} M_k A M_k A^{-1/2}) && \text{[by Lemma 2.1.4]} \\
&= \max_x |\tfrac{x^T M_k A M_k x}{x^T A x}| \\
&= \|M_k\|_A^2 .
\end{aligned}$$

Both the analysis with the weak and strong uniform assumptions require the analysis of the recursive term.

Lemma 7.3.1. *Let* $K = \hat{A}^{1/2} D^{1/2} R M_{k+1}^{\tau} A_{k+1}^{-1} R^T D^{1/2} \hat{A}^{1/2}$.
We have $\rho(K) \leq \rho(M_{k+1}^{\tau})$.

Let $Z = \hat{A}^{1/2}D^{1/2}R$, $\hat{A}_k = D_k^{-1/2}A_kD_k^{-1/2}$ where D_k is the diagonal of A_k and $\widetilde{M_k^\tau} = \hat{A_k}^{1/2}D^{1/2}M_k^\tau D^{-1/2}\hat{A_k}^{-1/2}$. We have

$$\begin{aligned}
\rho(K) &= \rho\left(ZM_{k+1}^\tau A_{k+1}^{-1}Z^T\right) \\
&= \rho\left(M_{k+1}^\tau A_{k+1}^{-1}Z^TZ\right) && \text{[by Lemma 2.1.4]} \\
&= \rho\left(M_{k+1}^\tau D_{k+1}^{-1/2}\hat{A}_{k+1}^{-1}D_{k+1}^{-1/2}Z^TZ\right) \\
&= \rho\left(\hat{A}_{k+1}^{1/2}D_{k+1}^{1/2}M_{k+1}^\tau D_{k+1}^{-1/2}\hat{A}_{k+1}^{-1/2}\hat{A}_{k+1}^{-1/2}D_{k+1}^{-1/2}Z^TZD_{k+1}^{-1/2}\hat{A}_{k+1}^{-1/2}\right) && \text{[similarity]} \\
&= \rho\left(\widetilde{M_{k+1}^\tau}\hat{A}_{k+1}^{-1/2}D_{k+1}^{-1/2}Z^TZD_{k+1}^{-1/2}\hat{A}_{k+1}^{-1/2}\right) \\
&\leq \rho\left(M_{k+1}^\tau\right)\rho\left(\hat{A}_{k+1}^{-1/2}D_{k+1}^{-1/2}Z^TZD_{k+1}^{-1/2}\hat{A}_{k+1}^{-1/2}\right) && \text{[by Lemma 2.1.12]} \\
&= \rho\left(M_{k+1}^\tau\right)\rho\left(ZD_{k+1}^{-1/2}\hat{A}_{k+1}^{-1}D_{k+1}^{-1/2}Z^T\right) && \text{[by Lemma 2.1.4]} \\
&= \rho(M_{k+1}^\tau)\rho\left(\hat{B}^{-1}\hat{A}\right) && \text{[by Lemma 2.1.4]} \\
&= \rho(M_{k+1}^\tau). && \text{[by Lemma 7.2.1]}
\end{aligned}$$

7.3.2 $\kappa(\hat{A}, \hat{B}_+)$-convergence

We let

$$\begin{aligned}
\tilde{M_k} &= \hat{A}^{1/2}D^{1/2}M_kD^{-1/2}\hat{A}^{-1/2} \\
\tilde{M_{k,k+1}} &= \hat{A}^{1/2}D^{1/2}M_{k,k+1}D^{-1/2}\hat{A}^{-1/2} \\
S &= I - \hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2}
\end{aligned}$$

and K be as defined in Lemma 7.3.1. We start with a recursive characterization of $\rho(M_k)$.

Theorem 7.3.2. $\rho(M_k) \leq \rho(M_{k,k+1}) + (1 - \rho(M_{k,k+1}))\rho(M_{k+1})^\tau$.

Proof. We wish to bound $\rho(M_k) = \rho(\tilde{M_k})$. Let x_t denote $(I - \hat{A})^t x$. We have

$$\rho(\tilde{M_k}) = \max_{\|x\|_2=1}\left(x_t^TSx_t + x_t^TKx_t\right)$$

Note that, by Lemma 7.2.2 we have $\mathcal{N}(K) = \mathcal{N}^\perp(S)$. Hence if $x_t^TSx_t = a$, $x_t^TKx_t \leq (1-a)\rho(K)$. By Lemma 7.3.1 we have $\rho(K) < \rho(M_{k+1}) < 1$, hence

$$\rho(\tilde{M_k}) \leq \rho(M_{k,k+1}) + (1 - \rho(M_{k,k+1}))\rho(M_{k+1})^\tau.$$

□

To characterize the two-level convergence we use the M_1-measure of [BCF$^+$00].

Lemma 7.3.3. *Let Π be any projection matrix onto $\mathcal{R}(D^{1/2}R)$. Assume that for all $x \neq 0$, we have*

$$M_1(\Pi, x) = \frac{x^T(I-\Pi)^2 x}{x^T \hat{A} x} \leq M.$$

Then,

$$\rho((I-\hat{A})S(I-\hat{A})) = \left\|(I-\hat{A})(I-\hat{B}^{-1}\hat{A})(I-\hat{A})\right\|_{\hat{A}} \leq 1 - 1/M$$

Proof. In [BCF$^+$00] it was shown that

$$\left\|(I-\hat{A})(I-\hat{B}^{-1}\hat{A})\right\|_{\hat{A}} \leq (1-1/M)^{1/2}$$

assuming that $\left\|\hat{A}\right\| = 1$ as it is the case in our setting. It can be shown [McC84] that

$$\left\|(I-\hat{A})(I-\hat{B}^{-1}\hat{A})\right\|_{\hat{A}} = \left\|(I-\hat{A})(I-\hat{B}^{-1}\hat{A})(I-\hat{A})\right\|_{\hat{A}}^{1/2}$$

□

Corollary 7.3.4. *We have $\rho(M_{k,k+1}) \leq 1 - 1/(2\lambda_{\max}(\hat{B}_+, \hat{A}))$.*

Proof. For all x we have

$$M_1(\Pi, x) = \frac{x^T(I-\Pi)^2 x}{x^T \hat{A} x} \leq \lambda_{\max}(\hat{B}_+, \hat{A}) \frac{x^T(I-\Pi)^2 x}{x^T \hat{B}_+ x}$$

Recall that $\hat{B}_+$ is the Schur complement of the Steiner graph with respect to the elimination of the internal vertices and it is of the form $\hat{B}_+ = (I - X)/2$ where $\mathcal{N}(X) = \mathcal{N}(R^T D^{1/2})$ and X is positive. Since $\hat{B}_+$ is positive we must have $\lambda_{\max}(X) \leq 1$. By combining these facts we get

$$x^T \hat{B}_+ x = \frac{1}{2}\left(x^T(I-\Pi)^2 x + x^T \Pi^2 x - x^T \Pi X \Pi x\right) \geq \frac{1}{2}(x^T(I-\Pi)^2 x).$$

Hence $M_1(\Pi, x) \leq 2\lambda_{\max}(\hat{B}_+, \hat{A})$ and the proof follows from Lemma 7.3.3. □

We are now ready to give the analogue of the convergence Theorem 4.1.1 for multigrid.

Theorem 7.3.5. *Let $\mathcal{H}(A)$ be a hierarchy of graphs with condition $\tau_{\mathcal{H}}$ and size reduction factor $\mu_{\mathcal{H}}$. If we take $t = 1$ and $\tau = \tau_{\mathcal{H}}$ in the statement of the multigrid algorithm, we have $\rho(M_k) \leq 1 - 1/\kappa$, where $\kappa = 2\max_i \lambda(B_i, A_i)$. If in addition the hierarchy satisfies*

$$\boxed{\tau_{\mathcal{H}}^2/\mu_H = (hierarchy\ condition)^2/(size\ reduction\ factor) < 1/2}$$

the complexity of $\mathbf{MG}(A, b, x, \mathcal{H}(A))$ is $O(\tau|A|)$.

Proof. We use induction on k. By Lemma 7.3.4 we know that $\rho(M_{k,k+1}) < 1-1/\kappa$ for some fixed κ. Assume inductively that $\rho(M_{k+1}) < 1 - 1/(2\kappa)$. Then, taking $\tau = 4\kappa \ln \kappa$ gives $(1 - \rho(M_{k,k+1}))\rho(M_{k+1})^\tau \leq 1/(2\kappa)$ and $\rho(M_k) < 1 - 1/(2\kappa)$. The complexity statement follows by an easy inductive argument similar to that of the proof of Theorem 4.1.1. □

Remark 1. Note that the definition of hierarchy condition is roughly similar to that in Chapter 4. Comparing then Theorems 7.3.5 and 4.1.1, highlights the difference of the two approaches in terms of the condition versus size reduction requirements.

Remark 2. Recall that that one run of $\mathbf{MG}$ reduces the A-norm of the error by a factor of $\rho(M_k)$. According to our discussion in Section 7.1, if $\rho(M_k) < 1 - 1/(2k)$, roughly $2k \ln \epsilon^{-1}$ repetitions of $\mathbf{MG}$ are required to make reduce the A-norm of the error by a factor of ϵ.

7.3.3 When and why $\kappa(\hat{A}, \hat{B}_+)$ is not sufficient

Let us for the sake of simplicity assume that $\kappa(\hat{A}, \hat{B}_+)$ is a constant. Consider the two level operator

$$\tilde{M}_k = (I - \hat{A})^t(I - \hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2})(I - \hat{A})^t.$$

We have $\rho(\tilde{M}_k) = \max_x x^T\tilde{M}_k x > x_2^T\tilde{M}_k x_2$ where x_2 is the first non-trivial eigenvector of A, normalized to have unit norm. Provided that λ_2 is small, the effect of $(I - \hat{A})^t$ is negligible for any reasonably small value of t, and thus we have

$$x_2^T\tilde{M}_k x_2 \simeq x_2^T(I - \hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2})x_2 = \max_{z \in \mathcal{N}(R^T D^{1/2}\hat{A}^{1/2})} (\frac{x_2^T z}{z^T z})^2$$

where the last equality follows from Lemma 7.2.2. Let $z = \hat{A}^{-1/2}w$ where w is an arbitrary unit norm in $\mathcal{N}(R^T D^{1/2})$ and thus an eigenvector of $\hat{B}_+$ with eigenvalue $1/2$. We have

$$z^T z = w^T\hat{A}^{-1}w = \frac{w^T\hat{A}^{-1}w}{w^T\hat{B}_+^{-1}w}(w^T\hat{B}_+^{-1}w) \leq 2\lambda_{\min}(\hat{A}^{-1}, \hat{B}_+^{-1}) = O(1).$$

On the other hand,

$$x_2^T z = \lambda_2^{-1/2}(x_2^T w)$$

and Theorem 6.1.3 guarantees only that $x_2^T w \leq O(\lambda_2^{1/2})$. If this bound is asymptotically matched by a lower bound, we get that $(x_2^T z)/(z^T z)$ is lower bounded by a constant for any reasonably small value of t.

Not surprisingly, to give a more specific example we will use the graphs (A, B) defined in Section 6.2.1 and used to show the optimality of Theorem 6.1.3. Recall that the Steiner graph S of the cycle graph A is constructed by grouping consecutive vertices of A into $n/2$ *disjoint* groups. It is easy to verify that the quotient graph Q is the Schur complement of A with respect to the elimination of the non-Steiner vertices and $Q = R^T A R$, where R is the restiction/indicator matrix for the partitioning of the vertices. It can be seen then that the matrices satisfy all the two-level requirements for the hierarchy. By using the analytic expressions for the eigenvectors and eigenvalues of $\hat{A}, \hat{B}_+$ we get

$$x_2^T \tilde{M}_k x_2 \geq \frac{1}{2}(1 - 2/n^2)^{2t}.$$

So, increasing the number of smoothings t, essentially does not affect the two-level convergence quality.

It seems that a better two-level convergence requires a spectral inequality stronger than that provided by Theorem 6.1.3 when $\kappa(\hat{A}, \hat{B}_+)$ is bounded. It is not hard to see that the same theorem provides stronger inequalities when $\kappa(\hat{A}^2, \hat{B}_+^2)$ is bounded. Indeed, it can be shown that if $\kappa(\hat{A}^2, \hat{B}_+^2)$ is bounded, increasing the number of smoothings to t, decreases the spectral radius roughly by a factor of t. This can be used to give a convergence bound on the basis of Theorem 7.3.2, by adjusting properly the size of t and making the two-level error sufficiently small. In the following Section we use ideas from [McC84] to show that a bound on $\kappa(\hat{A}^2, \hat{B}_+^2)$ actually implies a stronger two-level condition that leads to convergence with just $t = 1$ smoothing.

Remark. In Section 6.2.3 we mention that it is open whether the optimality bounds are optimal for the *full range* of the spectrum of the normalized Laplacian. If it is not, then the argument of this section might be an indication that the V-cycle has stronger error reduction properties for the very low frequency components of the error.

7.3.4 $\kappa(\hat{A}^2, \hat{B}_+^2)$-convergence

We start by showing that a multiplicative bound on the Rayleigh quotients for $(\hat{A}^2, \hat{B}_+^2)$ implies an additive bound on the Rayleigh quotients for $(\hat{A}^{-1}, \hat{B}^{-1})$. Concretely, we have

Lemma 7.3.6. $\rho(\hat{A}^{-1} - \hat{B}^{-1}) \leq 2\lambda_{\max}^{1/2}(\hat{B}_+^2, \hat{A}^2)(2 + \lambda_{\max}^{1/2}(\hat{A}^2, \hat{B}_+^2))$.

Proof. Let $M = \hat{A}^{-1} - \hat{B}^{-1}$. We note that $\rho(M) = \lambda_{max}^{1/2}(M^2)$. We concentrate on $\lambda(M^2)$. We have

$$\begin{aligned}
\rho(M^2) &= \rho\left((\hat{A}^{-1} - \hat{B}^{-1})^2\right) \\
&= \rho\left((I - \hat{B}^{-1}\hat{A})\hat{A}^{-2}(I - \hat{A}\hat{B}^{-1})\right) \\
&\leq \lambda_{\max}(\hat{A}^{-2}, \hat{B}_+^{-2})\rho\left((I - \hat{B}^{-1}\hat{A})\hat{B}_+^{-2}(I - \hat{A}\hat{B}^{-1})\right)
\end{aligned}$$

Recall that $\hat{B}_+^{-2} = 4(I + \hat{B}^{-1})^2 = 4(I + 2\hat{B}^{-1} + \hat{B}^{-2})$. By Lemma 7.2.1, the null space of the matrix $I - \hat{B}^{-1}A$ is $\mathcal{R}(D^{1/2}R)$. Notice in addition that for all vectors y, $\hat{B}^{-1}y \in \mathcal{R}(D^{1/2}R)$. Hence $(I - \hat{B}^{-1}A)\hat{B}^{-1} = 0$ and

$$\begin{aligned}
\rho(M^2) &\leq 4\lambda_{\max}(\hat{A}^{-2}, \hat{B}_+^{-2})\rho\left((I - \hat{B}^{-1}\hat{A})(I - \hat{A}\hat{B}^{-1})\right) \\
&= 4\lambda_{\max}(\hat{A}^{-2}, \hat{B}_+^{-2})\sigma_{\max}^2\left(I - \hat{B}^{-1}\hat{A}\right)
\end{aligned}$$

Using the fact that $I - \hat{B}^{-1}\hat{A} = I - \hat{B}_+^{-1}\hat{A} - \hat{A}$, we have

$$\sigma_{\max}(I - \hat{B}^{-1}\hat{A}) \leq 1 + \sigma_{\max}(\hat{A}) + \sigma_{\max}(\hat{B}_+^{-1}\hat{A}) \leq 2 + \lambda_{\max}^{1/2}(\hat{A}^2, \hat{B}_+^2).$$

Using Lemma 2.1.6 completes the proof. □

The rest of this section follows ideas from [McC84].

Corollary 7.3.7. *Let T be the projection matrix onto $\mathcal{N}(R^T D^{1/2}\hat{A}^{1/2})$ and $S = I - T$, be the projection matrix onto $\mathcal{R}(\hat{A}^{1/2}D^{1/2}R)$. Let x be an arbitrary vector and $\bar{x} = (I - \hat{A})x$. We have*

$$\|\bar{x}\|^2 \leq \alpha \|Tx\|^2 + \|Sx\|^2 \tag{7.5}$$

where $\alpha = 1 - 1/\left(2\lambda_{\max}^{1/2}(\hat{B}_+^2, \hat{A}^2)(2 + \lambda_{\max}^{1/2}(\hat{A}^2, \hat{B}_+^2))\right)$.

Proof. Using the fact that for all x we have $\|x\|^2 = \|Tx\|^2 + \|Sx\|^2$, equation 7.5 holds if and only if $\|x\|^2 - \|\bar{x}\|^2 \geq (1 - \alpha)\|Tx\|^2$. From this we get

$$a = 1 - \inf_x \frac{\|x\|^2 - \|\bar{x}\|^2}{\|Tx\|^2}.$$

By Lemma 7.2.2, we have $T^2 = T = I - \hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2}$. We also have

$$\|x\|^2 - \|\bar{x}\|^2 = x^T x - x^T(I-\hat{A})^2 x = x^T(I-(I-\hat{A})^2)x = x^T(2\hat{A}-\hat{A}^2)x \geq x^T\hat{A}x.$$

Combining the above and using Lemma 2.1.6, we have

$$\begin{aligned}\frac{\|x\|^2 - \|\bar{x}\|^2}{\|Tx\|^2} &\geq \frac{x^T\hat{A}x}{x^T(I-\hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2})x} \\ &\geq \lambda_{\min}(\hat{A}, I-\hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2}) = \lambda_{\min}(I, \hat{A}^{-1}-\hat{B}^{-1})\end{aligned}$$

The proof is completed by invoking Lemma 7.3.6. □

We conclude this section with a characterization of the convergence of the full V-cycle.

Theorem 7.3.8. *Let M_k be as defined in equation 7.4, with $t = \tau = 1$. We have $\rho^2(M_k) \leq \alpha$ where α is the constant in Corollary 7.3.7.*

Proof. Let $\tilde{M_k} = \hat{A}^{1/2}D^{1/2}M_k D^{-1/2}\hat{A}^{-1/2}$. We have $\rho(M_k) = \rho(\tilde{M_k})$, by the similarity transformation. Assume for induction that $\rho^2(M_{k-1}) \leq \alpha$. Let $T = I - \hat{A}^{1/2}\hat{B}^{-1}\hat{A}^{1/2}$ and $K = \hat{A}^{1/2}D^{1/2}RM_{k+1}A_{k+1}^{-1}R^T D^{1/2}\hat{A}^{1/2}$. Note that K is symmetric, $T^2 = T, TK = 0, ST = 0$ and $SK = KS = K$. Using these facts and applying Corollary 7.3.7, we have

$$\begin{aligned}\rho^2(\tilde{M_k}) &= \max_{\|x\|=1} \left\|\tilde{M_k}x\right\|^2 \\ &\leq \left\|(I-\hat{A})(T+K)x\right\|^2 \\ &\leq \alpha\|Tx\|^2 + \|Kx\|^2 \\ &= \alpha\|Tx\|^2 + \|K\|^2\|Sx\|^2 \\ &= \alpha\|Tx\|^2 + \rho^2(M_{k-1})\|Sx\|^2 \\ &\leq \alpha(\|Tx\|^2 + \|Sx\|^2) = \alpha.\end{aligned}$$

□

7.4 Multigrid based on edge separators

The discussion in this Section requires an understanding of the constructions in Section 5.3, and generalizes the construction we considered in the beginning of this Chapter. We

are concerned with multigrid schemes constructed from Steiner graphs based on edge separators, such as those constructed and discussed in Chapter 5. An example for the line graph is depicted in Figure 7.1.

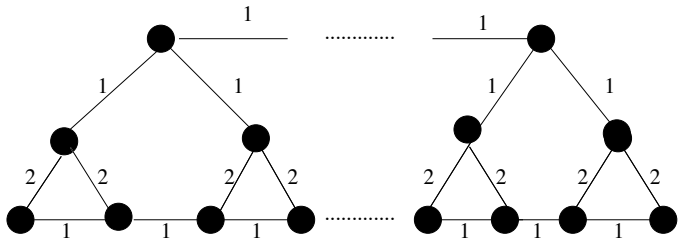

Figure 7.1: Multigrid based on edge separators.

Consider a laminar decomposition $\mathcal{H} = \{H_1, \ldots, H_d\}$ of a given graph A, where H_d contains the vertices of A as singletons. Recall that by definition, the sets of the i^{th} level of the decomposition, are grouped into sets in its $(i-1)^{th}$ level. We construct the Steiner graph S by taking the laminar Steiner tree described in Section 5.2, removing its vertices corresponding to sets contained in the decompositions above the l^{th} level, and connecting the roots of the remaining trees to form the quotient graph Q. Let R_i^{i-1} be the matrix with rows corresponding to sets in H_i and columns corresponding to the sets in H_{i-1}. For all j, k we define $R_i^{i-1}(j,k) = 0$, unless the j^{th} set of H_i is contained in the k^{th} set of H_{i-1}, in which case we let $R_i^{i-1}(j,k) = 1$. If $R = R_d^{d-1} R_{d-1}^{d-2} \ldots R_{l+1}^{l}$. By an inductive argument, It can be seen that $Q = R^T A R$. It can be also verified that the Schur complement B of S with respect to the elimination of the non-Steiner vertices, satisfies $B^{-1} = D^{-1} + RQ^{-1}R^T$. Thus the definition of the Steiner graph satisfies the requirements set in Section 7.2.1 for the two-level operators.

We give two concrete examples for model meshes. In the 1D-case, the line graph, the hierarchy of preconditioners described in Section 5.3 has condition $O((k \log k)^{1/2})$ and reduction factor k. These values do not satisfy the requirements of Theorem 7.3.5. Thus we cannot hope that multigrid based on edge separators can work in the general case. However, for the 2D-case, the square grid, the hierarchy has condition $O((k \log k)^{1/2})$ and reduction factor k^2. Hence for a large enough constant k it satisfies the requirements of Theorem 7.3.5. The relationship between the hierarchy condition and the reduction factor becomes more favorable for grids on higher dimensions.

From a practical point of view, for given 2D or 3D instances one should examine the possibility of using Steiner preconditioners in combination with multigrid, before resorting to the Chebyshev method which has more relaxed requirements but in general requires

more recursive calls and has higher space and time complexity that the two Richardson's iterations required (per visit) on each level of the hierarchy.

7.5 Multigrid based on vertex separators

In the algebraic multigrid terminology, disjoint clusterings of the variables are known as "aggregates" corresponding to "supernodes" in the second level graph. The fact that the V-cycle -with the simple $0-1$ restriction operators R considered in the previous section- does not converge has led to multigrid algorithms based on "smoothed aggregation". In those algorithms the restriction (or prolongator) R is "tentative" and the final restriction/prolongator operator is constructed by applying a smoothing operator S to R. If A is the given matrix, the second level matrix Q is constructed as $Q = (SR)^T A(SR)$, which in general may have more edges than A [VBM01].

The usual AMG approach consists of the following steps: (i) the choice of a subset of the variables that form the second level graph often called the "coarse" grid (ii) the assignment of each "fine" grid point to a small number of coarse grid points to which they depend strongly, (iii) the choice of interpolation/projection operators that transform vectors in the coarse space to vectors in the fine space, and vice-versa. In general, the algorithms for performing these steps are mostly based in heuristics whose computational costs "cannot be predicted precisely" [Bra86, BHM00].

Although the AMG heuristics are commonly viewed as a selection of coarse variables whose values are kept by the restriction/interpolation operators between the fine and the coarse grid, it can be also viewed as an partition of the fine grid vertices to *overlapping* clusters of vertices. This motivates us to consider multigrid derived from Steiner graphs based on vertex separators, or disjoint clusters of strongly dependent edges, that is expanders. The simple Steiner preconditioner for the line is shown in Figure 7.2.

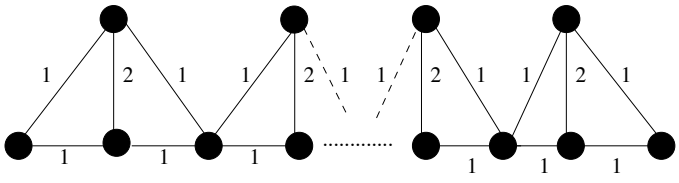

Figure 7.2: Multigrid based on vertex separators.

We now formally describe the two-level operators based on disjoint clusters of edges, such that the graph induced by each cluster is an expander. Let P be a $[\phi, \rho]$-decomposition

of a given graph $C = (V, E, w)$ into disjoint sets V_i, $i = 1, \ldots, m$, as defined in Section 5.5. We define the graph A, by replacing every edge e of C with two edges, each having weight $2w(e)$. We construct a Steiner preconditioner for A.

The separator P defines a vertex separator P' that disconnects the edge set of A into disjoint sets E_i. Let A_i be the graph induced by the edges in E_i. Also, let $\phi = \max_i \phi(A_i)$, where $\phi(A_i)$ denotes the conductance of A_i. Let S_i be the star graph with leaves corresponding to the vertices of A_i. We let the **Steiner graph** S be $S = \sum_{i=1}^{m} S_i$. Note that S is a bipartite graph, with edges joining only Steiner with non-Steiner vertices. If B_i is the Schur complement of S_i with respect to the elimination of the cener vertex, the **preconditioner** is given by $B = \sum_{i=1}^{m} B_i$. The **quotient** graph Q of S after the elimination of the non-Steiner vertices consists of m vertices, each corresponding to a set in the partition P, and $Q_{ij} = cap(V_i, V_j)$. This can be verified algebraically, or by using the electric analogy and the fact that between the roots of S_i and S_j we have a set of resistors connected in parallel. Note that every vertex of A belongs to either one or two subgraphs A_i. We define the $n \times m$ **restriction matrix** R as follows: (i) if vertex i belongs to A_i only, $R(i, j) = 1$, (ii) if vertex i belongs to A_j and A_k, $R(i, j) = 1/2$ and $R(i, k) = 1/2$, (iii) if vertex i does not touch A_j, $R(i, j) = 0$. If D is the diagonal matrix with the vertex volumes in A and D_S is the diagonal matrix with the volumes of the centers of the stars, we have

$$S = \begin{pmatrix} D & -DR \\ -R^T D & D_S \end{pmatrix}$$

The partial Cholesky factorization LDL^T of the non-Steiner vertices of S gives

$$S = \begin{pmatrix} I & 0 \\ -R^T & I \end{pmatrix} \begin{pmatrix} D & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} I & -R \\ 0 & I \end{pmatrix}.$$

Repeating the derivation of equation 7.1 shows that $B^{-1} = D^{-1} + RQ^{-1}R^T$. Viewing A as an electrical network, the j^{th} column of AR is the vector of the residual flows when the voltages are set to 1 on the interior vertices of A_j, to $1/2$ on its boundary vertices and to 0 on the remaining vertices. Using this it can be seen that we have

$$Q = R^T AR. \tag{7.6}$$

Thus the definition of the Steiner graph satisfies the requirements set in Section 7.2.1 for the two-level operators.

The preconditioners of this section are markedly different than those considered in Section 7.4. It can be seen that we have $\hat{A} = \sum_i \hat{A}_i = \sum_i D_i^{-1/2} A_i D_i^{-1/2}$ and $\hat{B}_+ = \sum_i \hat{B}_i = \sum_i D_i^{-1/2} B_i D_i^{-1/2}$, where A_i, B_i are expanders on the same set of vertices, with

(up to a factor of 2) equal vertex volumes, contained in D_i. This implies that for all i, $\kappa(\hat{A}_i^{\,2}, \hat{B}_i^{\,2}) = O(\phi^4)$. This can be shown by using the Cheeger inequality as in the proof of Theorem 5.3.1. Although this does not imply that $\kappa(\hat{A}^2, \hat{B}_+^2)$ is bounded, it may be a first step towards a better understanding of the success and the limitations of the V-cycle. In future research we intend to explore the theoretical and practical potential of multigrid based on vertex separators, constructed from $[\phi, \rho]$-decompositions.

Bibliography

[AHK04] Sanjeev Arora, Elad Hazan, and Satyen Kale. $O(\sqrt{(}\log n))$ approximation to SPARSEST CUT in $O(\tilde{n}^2)$ time. In *FOCS, 45th Symposium on Foundations of Computer Science (FOCS 2004)*, pages 238–247, 2004. 2.2.1

[AKPW95] Noga Alon, Richard Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995. 2.4.5

[ARV04] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC '04: Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 222–231, 2004. 2.2.1

[AS00] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. John Wiley and Sons, inc, 2000. 2.2.1

[Axe94] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, NY, 1994. 2.4.3, 2.4.4

[BCF+00] M. Brezina, A. J. Clerly, R.D. Falgout, V. E. Henson, J.E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal on Scientific Computing*, 22(5):1570–1592, 2000. 7.3, 7.3.2, 7.3.2

[BD90] Jesse Barlow and James Demmel. Computing accurate eigensystems of scaled diagonally dominant matrices. *SIAM Journal on Numerical Analysis*, 27(3):762–791, 1990. 6.1.1

[BGH+06] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 4:930–951, 2006. 2.4.5

[BH03] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, 2003. 2.1, 2.4.5, 5

[Bha97] Rajendra Bhatia. *Matrix Analysis*. Springer-Verlag, New York, 1997. 2.1, 2.1

[BHM00] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial: second edition*. Society for Industrial and Applied Mathematics, 2000. 2.4.2, 2.4.2, 7.5

[BHV04] Erik G. Boman, Bruce Hendrickson, and Stephen A. Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. *CoRR*, cs.NA/0407022, 2004. 1

[Big94] Norman Biggs. *Algebraic Graph Theory*. Cambdridge University Press, 1994. 2.2.3, 6.2.2

[BKR03] Marcin Bienkowski, Miroslaw Korzeniowski, and Harald Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms*, pages 24–33, 2003. 2.4.5, 5.2, 5.4, 5.4.1

[BMM99] Claudson F. Bornstein, Bruce M. Maggs, and Gary L. Miller. Tradeoffs between parallelism and fill in nested dissection. In *SPAA*, pages 191–200, 1999. 2.3.4

[BMMR97] Claudson F. Bornstein, Bruce M. Maggs, Gary L. Miller, and R. Ravi. Parallelizing elimination orders with linear fill. In *FOCS*, pages 274–283, 1997. 2.3.4

[BMR84] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*. Cambridge University Press, Cambridge, 1984. 2.4.2

[Bra77] A. Brandt. Multi–level adaptive techniques (MLAT) for partial differential equations: ideas and software. In J. R. Rice, editor, *Mathematical Software III*, pages 277–318. Academic Press, New York, 1977. 2.4.2

[Bra86] A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 19:23–56, 1986. 2.4.2, 7.5

[Bra93] James H. Bramble. *Multigrid Methods*. Chapman and Hall, 1993. 2.4.2

[CDS98] D.M Cvetkovic, M. Doob, and H. Sachs. *Spectra of Graphs*. Johann Ambrocious Barch, 1998. 2.2.3

[CFH+00] Andy Cleary, Rob Falgout, Van Emden Henson, Jim Jones, Tom Manteuffel, Steve McCormick, Jerry Miranda, and John Ruge. Robustness and scalability of algebraic multigrid. *SIAM Journal of Scientific Computing*, 21(5):1886–1908, 2000. 2.4.2

[CFH+03] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski. Spectral AMGe (ρAMGe). *SIAM J. Sci. Comput.*, 25(1):1–26, 2003. 7.3

[Che01] Doron Chen. Analysis, implementation, and evaluation of Vaidya's preconditioners. Master's thesis, School of Mathematical Sciences, Tel-Aviv University, 2001. 2.3.4, 2.4.5

[Chu97] F.R.K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. American Mathematical Society, 1997. 2.2.1, 2.2.3

[Chu07] Fan Chung. Random walks and local cuts in graphs. *Linear Algebra and its applications*, 423(1):22–32, 2007. 5.2

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal Symbolic Comp.*, 9(3):251–280, March 1990. 2.3.5

[Dem97] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997. 2.3.5

[DR83] Iain Duff and John Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Trans. Math. Softw.*, 9:302–325, 1983. 2.3.4

[DS00] Peter G. Doyle and J. Laurie Snell. Random walks and electric networks, 2000. 2.2.3

[Duf74] Iain Duff. On the number of nonzeros added when Gaussian elimination is performed on sparse random matrices. *Math. Comp.*, 28:219–230, 1974. 2.3.4

[EEST05] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 494–503, 2005. 2.2.1, 2.4.5, 4, 4.2

[EI95] Stanley C. Eisenstat and Ilse C. F. Ipsen. Relative perturbation techniques for singular value problems. *SIAM Journal on Numerical Analysis*, (6):1972–1988, 1995. 6.1.1

[EMT93] David Eppstein, Gary L. Miller, and Shang-Hua Teng. A deterministic linear time algorithm for geometric separators and its applications. In *Symposium on Computational Geometry*, pages 99–108, 1993. 2.3.4

[Fed64] R. P. Fedorenko. The speed of convergence of one iterative process. *Z. Vycisl. Mat. i. Mat. Fiz.*, 4:559–563, 1964. Also in U.S.S.R. Comput. Math. and Math. Phys., 4 (1964), pp. 227–235. 2.4.2

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973. 2.2.3

[FPS05] Francois Fouss, Alain Pirotte, and Marco Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *ACM International Conference on Web Intelligence*, pages 550–556, 2005. 1

[Fre87] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987. 2.2.2, 3, 3.6

[Geo73] Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973. 2.3.4

[GGKK94] Ananth Grama, Anshul Gupta, Vipin Kumar, and George Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings Publishing Company, Redwood City, CA, 1994. 2.3.4

[GHT84] John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984. 2.2.2, 2.3.4

[GL96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 3d edition, 1996. 2.3.2

[GM87] Hillel Gazit and Gary L. Miller. A parallel algorithm for finding a separator in planar graphs. In *28th Annual Symposium on Foundations of Computer Science*, pages 238–248, 1987. 2.2.2, 3, 3

[GM95] Stephen Guattery and Gary L. Miller. On the performance of spectral graph partitioning methods. In *SODA*, pages 233–242, 1995. 2.2.1

[GM98] Stephen Guattery and Gary L. Miller. On the quality of spectral separators. *SIAM J. of Matrix Analysis and Applications*, 19(3):701–719, July 1998. 2.2.1

[Goo95] Michael T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. Syst. Sci.*, 51(3):374–389, 1995. 2.2.2

[Gra06] Leo Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(11):1768–1783, 2006. 1

[Gre96] Keith 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. 2.4.5, 4, 4.2.3, 5, 5.3, 7

[GT87] John R. Gilbert and Robert E. Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50(4):377–404, 1987. 2.3.4

[Hac78] W. Hackbusch. On the multigrid method applied to difference equations. *Computing*, 20:291–306, 1978. 2.4.2

[HJ85] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985. 2.1

[HJ91] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991. 2.1

[HL95] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations, 1995. 2.2.1

[Jos97] Anil Joshi. *Topics in Optimization and Sparse Linear Systems*. PhD thesis, University of Illinois at Urbana Champaing, 1997. 2.4.5

[Kel04] Jonathan A. Kelner. Spectral partitioning, eigenvalue bounds, and circle packings for graphs of bounded genus. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 455–464, New York, NY, USA, 2004. ACM Press. 2.2.2

[KK98] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998. 2.2.1

[Kle93] Philip N. Klein. On Gazit and Miller's parallel algorithm for planar separators: Achieving greater efficiency through random sampling. In *SPAA*, pages 43–49, 1993. 4.2.2

[KRV06] Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 385–390, New York, NY, USA, 2006. ACM Press. 2.2.1

[KST01] Marcos A. Kiwi, Daniel A. Spielman, and Shang-Hua Teng. Min-max-boundary domain decomposition. *Theor. Comput. Sci.*, 261(2):253–266, 2001. 2.2.2, 3

[KVV04] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004. 5.5

[Li98] Ren-Cang Li. Relative perturbation theory: I. eigenvalue and singular value variations. *SIAM Journal on Matrix Analysis and Applications*, 19(4):956–982, 1998. 6.1.1

[Li99] Ren-Cang Li. Relative perturbation theory: II. eigenvalue and singular value variations. *SIAM Journal on Matrix Analysis and Applications*, 20(2):471–492, 1999. 6.1.1

[Lov93] László Lovász. Random walks on graphs: A survey. *Combinatorics.*, Paul Erdős is Eighty (2):1–46, 1993. 2.2.3

[LR99] Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. 2.2.1

[LRT79] R.J. Lipton, D. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis*, 16:346–358, 1979. 2.3.4, 4.2.3

[LT79] R. J. Lipton and R. E. Tarjan. A planar separator theorem. *SIAM Journal of Applied Mathematics*, 36(2):177–189, April 1979. 2.2.2, 2.3.4, 3.1

[Lub86] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986. 3.3

[McC84] S. F. McCormick. Multigrid methods for variational problems: further results. *SIAM J. Numer. Anal.*, 21:255–263, 1984. 7.3, 7.3.2, 7.3.3, 7.3.4

[Mil86a] Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986. 2.2.2

[Mil86b] Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, June 1986. invited publication. 3.1, 3.5

[MMP+05] Bruce M. Maggs, Gary L. Miller, Ojas Parekh, R. Ravi, and Shan Leung Maverick Woo. Finding effective support-tree preconditioners. In *Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms*, pages 176–185, 2005. 2.4.5, 4.2.3, 5, 5.2, 5.3, 5.4, 5.4.2

[Moh99] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, 12(1):6–26, 1999. 2.2.2

[MR04] Gary L. Miller and Peter C. Richter. Lower bounds for graph embeddings and combinatorial preconditioners. In *Proceedings of the sixteenth Annual ACM Symposium on Parallel Algorithms*, pages 112–119, 2004. 2.4.5, 4.2.3

[MV98] Roy Mathias and Krešimir Veselić. A relative perturbation bound for positive definite matrices. *Linear Algebra and its applications*, 270:315–321, 1998. 6, 6.1.1, 6.1.2, 6.1.2

[Nic78] R. A. Nicolaides. On the observed rate of convergence of an iterative method applied to a model elliptic difference equation. *Math. Comp.*, 32:127–133, 1978. 2.4.2

[NJW01] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm, 2001. 1

[NSS98] Assaf Natanzon, Ron Shamir, and Roded Sharan. A polynomial approximation algorithm for the minimum fill-in problem. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 41–47, New York, NY, USA, 1998. ACM Press. 2.3.4

[Pap94] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994. 3

[PR93] Victor Y. Pan and John H. Reif. Fast and efficient parallel solution of sparse linear systems. *SIAM J. Comput.*, 22(6):1227–1250, 1993. 4.2.2

[PSL90] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990. 2.2.1

[RÖ02] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 43–52. IEEE, 2002. 2.4.5

[Rei98] John Reif. Efficient approximate solution of sparse linear systems. *Computers and Mathematics, with Applications*, 36(9):38–52, 1998. 2.4.5

[RG97] Gordon Royle and Chris Godsil. *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer Verlag, 1997. 2.2.3

[RMMM93] Margaret Reid-Miller, Gary L. Miller, and Francesmary Modugno. List ranking and parallel tree contraction. In John Reif, editor, *Synthesis of Parallel Algorithms*, pages 115–194. Morgan Kaufmann, 1993. 3.6, 5.5

[Sha03] Y. Shapira. *Matrix-Based Multigrid : Theory and Applications*. Numerical Methods and Algorithms. Springer, 2003. 2.4.2

[SS90] G.W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, Boston, 1990. 2.1, 2.3.2, 6

[ST96] Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *FOCS*, pages 96–105, 1996. 2.2.2

[ST03] Daniel A. Spielman and Shang-Hua Teng. Solving Sparse, Symmetric, Diagonally-Dominant Linear Systems in Time 0($m^{1.31}$). In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 416. IEEE Computer Society, 2003. 2.2.1, 2.4.5

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 81–90, June 2004. 2.2.1, 2.3.4, 2.4.5, 4.1.2, 4.2, 5.5, 7

[ST06] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems, 2006. 2.4.4, 2.4.5

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematic.*, 13:354–356, 1969. 2.3.5

[TM06] David Tolliver and Gary L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 1053–1060, 2006. 1

[TSO00] Ulrich Trottenberg, Anton Schuller, and Cornelis Oosterlee. *Multigrid*. Academic Press, 1st edition, 2000. 2.4.2, 7.1

[Vai91] Preadeep M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. A talk based on this manuscript, October 1991. 2.3.4, 2.4.5

[VBM01] Petr Vanek, Marian Brezina, and Jan Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001. 7.5

[Wes04] Pieter Wesseling. *An Introduction to Multigrid Methods*. R.T. Edwards, Inc., 2nd edition, 2004. 2.4.2

[Yan81] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal of Algebraic and Discrete Mathematics*, 2(1):77–79, 1981. 2.3.4