Spectral Methods for Bisecting Graphs

Area Exam by Robert D. Blumofe

January 28, 1993

1 Introduction

Partitioning the vertices of a graph into two equal size pieces so as to minimize the number of edges crossing the partition, a problem known as the *graph bisection* problem, finds use in applications ranging from approximation algorithms to VLSI layout. For example, Lipton and Tarjan [27] have shown that by partitioning a graph into two (nearly) equal size pieces such that a (fairly) small number of edges connect the two pieces, many NP-complete graph problems admit good approximation algorithms. As another application, Denning [12] points out that the performance of paged virtual memory systems can be improved by partitioning code into pages in a way that minimizes the number of references between different pages. In numerical methods, graph partitioning can be used in nested dissection to compute a good ordering for the parallel factorization of sparse, symmetric matrices [29]. In VLSI layout, Bhatt and Leighton [5] give a layout algorithm that uses a graph bisection algorithm as a subroutine, and they give a quantitative relationship showing that if the bisection algorithm finds bisections that are near optimal, then the layout algorithm produces layouts that are near optimal. Of course graph bisection finds many other applications, but one more deserves special mention: partitioning finite-element graphs for execution on parallel supercomputers [22, 30]. Many scientific and engineering computations involve large, irregular finite-element graphs. To execute these computations on a parallel machine with P processors, the graph must partitioned into P pieces in a manner that minimizes the communication between different pieces. The spectral methods for graph bisection that are the focus of this paper have found widespread use in this application [24, 30].

Given the utility of graph bisection, it is disappointing (though not surprising) that this problem is difficult from a complexity point-of-view. For general graphs, Garey, Johnson, and Stockmeyer [21] (see also [10]) have shown that the decision problem, "Does graph G have a bisection with fewer than k edges?" is NP-complete. For planar graphs the complexity is unknown, but Papadimitriou and Sideri [28] conjecture that the problem is still NP-complete. Polynomial time algorithms for finding optimal bisections are known only for trees, bounded-width planar graphs, and solid grid graphs [28]. Furthermore, no approximation algorithms are known even for planar graphs. The current state-of-the-art consists of heuristics that often perform well in practice [2, 4, 11, 25] and algorithms that have been shown to have good average case behavior over certain probability distributions on graphs [7, 9, 11, 14].

The next section briefly surveys some of these algorithms. The rest of this paper focuses on the spectral methods and is organized as follows. Section 3 covers some preliminary facts concerning the eigenvalues and eigenvectors of real, symmetric matrices. This section concludes with two theorems that play a central role in the two spectral algorithms that we will study. One of these two theorems

is a slight generalization of a theorem due to Fan [15] that forms the basis for Section 4's simplified exposition of an algorithm due to Barnes [2]. The other theorem fills in a major hole in Boppana's extended abstract [7]. Boppana's algorithm is the topic of Section 5. We point out a problem with the original algorithm, discuss a remedy proposed by Boppana [8], and then demonstrate the problem with the remedy. Fortunately, these problems can be worked around by showing that for some classes of random graphs, they don't occur very often. This probabilistic analysis is the subject of Section 6. This paper concludes in Section 7 with a comparison and critique of the two algorithms discussed.

Before moving on, we pause briefly for some definitions. Given a graph $G = (V, E)$ with $n = |V|$ vertices, a *bisection* (V_1, V_2) is a partitioning of the vertices into two pieces ($V_1, V_2 \subset V$ with $V_2 = V - V_1$) such that $||V_1| - |V_2|| \leq 1$. The *size* $\mathrm{b}(G, (V_1, V_2))$ of the bisection is the number of edges $e \in E$ such that e has one endpoint in V_1 and the other endpoint in V_2. A minimum (or optimal) bisection is a bisection with the smallest size, and the *bisection width* $\mathrm{b}(G)$ is the size of a minimum bisection.

2 A survey of algorithms

Graph bisection algorithms can be classified as either local or global. Local algorithms start with some initial bisection from which iterative improvements are made. Global algorithms start with just the adjacency matrix and use global properties such as minimum cuts or spectral decompositions in order to identify a small bisection. In practice, by starting from several randomly chosen initial bisections or from a bisection found by a global algorithm, local algorithms can often identify optimal bisections [9, 11].

Local algorithms

Given a graph $G = (V, E)$ and an initial bisection (V_1, V_2), a local algorithm at each iteration identifies a pair (v_1, v_2) of vertices with $v_1 \in V_1$ and $v_2 \in V_2$ and swaps them to create a new bisection (V_1', V_2') with $V_1' = (V_1 - \{v_1\}) \cup \{v_2\}$ and $V_2' = (V_2 - \{v_2\}) \cup \{v_1\}$. Of course, the greedy approach is to always pick the pair (v_1, v_2) that produces the largest decrease in bisection size. The algorithm ends when no swap can decrease the bisection size, but most implementations will accept a swap that produces no net change in the bisection size and therefore must incorporate some method to ensure that the algorithm terminates. (For further discussion of the greedy algorithm, see [11].)

By iteratively decreasing the bisection size, the greedy algorithm is guaranteed to find a locally optimal bisection, that is, a bisection for which no single swap can decrease its size. Pretending the space of bisections is one-dimensional, the greedy algorithm at a locally optimal bisection might appear as in Figure 1. No neighboring bisection has smaller size, but in this case, a smaller bisection does exist. By refusing to make a swap that increases the bisection size, the greedy algorithm cannot get over the hump to find the smaller bisection.

With a willingness to accept some swaps that increase the bisection size, other local algorithms can scale some of these humps and thereby often find better bisections than the greedy algorithm. For example, the Kernighan-Lin algorithm [25] at each iteration constructs a list $(v_1^{(1)}, v_2^{(1)}), (v_1^{(2)}, v_2^{(2)}), \ldots, (v_1^{(l)}, v_2^{(l)})$ of $l = \lfloor n/2 \rfloor$ pairs defined as follows: $(v_1^{(1)}, v_2^{(1)})$ is the pair whose swap produces the largest decrease in bisection size. In general, $(v_1^{(i)}, v_2^{(i)})$ is the pair whose

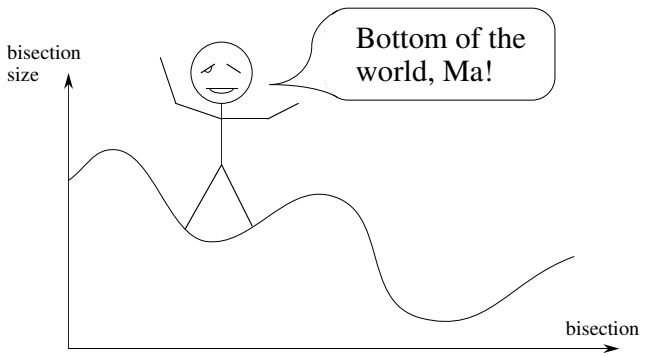

Figure 1: Cagney in *White Heat*.

swap produces the largest decrease in bisection size starting from the bisection produced by first swapping all the pairs $(v_1^{(j)}, v_2^{(j)})$ for $j < i$. The algorithm then chooses the best possible prefix of this list and performs those swaps; that is, it chooses the value k that maximizes the decrease in bisection size that results from swapping each pair $(v_1^{(j)}, v_2^{(j)})$ for $j \leq k$, and then it performs those swaps. (With a clever use of data structures, each iteration of Kernighan-Lin can be made to run in $O(n)$ time [16]). The algorithm terminates when no prefix can improve the bisection. Notice that this algorithm may perform some swaps that increase the bisection size so long as performing such swaps enables it to find later swaps that ultimately result in a net improvement.

The simulated annealing approach to graph bisection as described in [11] uses a randomized method for accepting bad swaps. At each iteration the algorithm chooses a random pair. If swapping this pair improves the bisection, then the swap is performed. If the swap would increase the bisection size by an amount Δ, then the swap is performed with probability $e^{-\Delta/T}$ where T denotes the *temperature* of the algorithm. The algorithm starts with the temperature high so that the algorithm accepts some bad swaps. Over the course of running the algorithm, the temperature decreases so that eventually only good swaps are accepted. Thus, the simulated annealing algorithm, like the other local algorithms, always finds a locally optimal bisection.

When run on randomly generated d-regular graphs with small bisection width, Bui, Chaudhuri, Leighton, and Sipser [9] observed that if started from several randomly chosen initial bisections, both Kernighan-Lin and simulated annealing almost always outperform the greedy algorithm. In fact, for $d \geq 4$, these two algorithms almost always output the minimum size bisection. For $d = 3$, all three of these local algorithms perform poorly, often outputting a $\Theta(n)$ size bisection on graphs with bisection width 0. The authors found, however, that this problem with low-degree graphs can be overcome by first contracting the graph to generate a high-degree graph and then using a bisection found in this new graph as a starting bisection for Kernighan-Lin or simulated annealing to use on the original graph [11]. Not surprisingly, some mountains are too high for these local algorithms to cross, and therefore, the quality of the initial bisection is critical.

Before moving on to the global algorithms, one more local algorithm merits mention. As we will illustrate later, finding a minimum bisection is the same as minimizing a particular quadratic function of n variables subject to some conditions. Barnes, Vannelli, and Walker [4] show that given an initial bisection, this quadratic function can be approximated by a linear function for bisections that are close to the initial bisection. This linear function is very easy to minimize. The solution to the linear problem yields a new bisection, and the algorithm iterates.

Global algorithms

Unlike the local algorithms, global algorithms do not rely on an initial bisection, and rather than settle for a locally optimal bisection, global algorithms attempt to produce a minimum size bisection.

Furthermore, whereas local algorithms have defied analysis (a notable exception being Bui's lower bound [11] demonstrating the ineffectiveness of the greedy algorithm for 3-regular graphs), several global algorithms admit average case analysis over particular classes of random graphs. A natural class to consider is $\mathcal{G}(n,m)$ — a uniform distribution on graphs with n vertices, $V = \{1,2,\ldots,n\}$, and m edges. But through a series of tight probabilistic bounds, Bui [10] has shown that almost every graph in this class has bisection width $\Theta(m)$. In fact, as $m/n \to \infty$ this bound becomes $m/2 - o(m)$, so the best bisection differs from the average bisection in only a second order term. Therefore, the class $\mathcal{G}(n,m)$ cannot distinguish (beyond a constant factor) a good algorithm from a really bad algorithm (for example, an algorithm that tries to maximize the bisection size). To remedy this situation, several authors [7, 9, 14] argue that the class $\mathcal{G}(n,m,b)$ — a uniform distribution on graphs with n vertices, m edges, and bisection width $b < m/2$ — facilitates a more meaningful analysis of their graph bisection algorithms. We will have more to say later about the adequacy of average case analysis based on such probability distributions; for now we simply mention two of these results.

Dyer and Frieze [14] give a polynomial expected time algorithm for dense graphs with fairly small bisection. Their algorithm always finds a minimum size bisection, and for graphs drawn from $\mathcal{G}(n,m,b)$ with $m = \Omega(n^2)$ and $b = (\frac{1}{2} - \epsilon)m$ (where $\epsilon > 0$ is a fixed constant), their algorithm runs in polynomial expected time. This algorithm actually employs a series of 3 algorithms. The first algorithm runs in polynomial time, and for almost every graph in $\mathcal{G}(n,m,b)$, it produces a minimum size bisection and a proof of optimality. For those graphs that fail the first algorithm, a second, slightly-sub-exponential time algorithm is called upon. The fraction of graphs that fail the first algorithm is small enough and the fraction of graphs that also fail the second algorithm is further small enough that by finishing with a brute-force, exponential time algorithm, the expected total running time remains polynomial.

Bui, Chaudhuri, Leighton, and Sipser [9] consider a slightly different class of random graphs and employ a max-flow/min-cut based technique. The class they consider is $\mathcal{G}(n,d,b)$ — a uniform distribution on d-regular ($d \geq 3$) n-vertex graphs with bisection width $b = o(n^{1-1/\lfloor (d+1)/2 \rfloor})$. The authors show that over this class of random graphs, with probability at least $1-o(1)$, their algorithm outputs the unique minimum size bisection along with a proof of optimality, all in polynomial time. At each iteration, the algorithm picks two vertices u and v, collapses a neighborhood of vertices around u to create an infinite capacity source, collapses a neighborhood of vertices around v to create an infinite capacity sink, and then runs a flow algorithm to identify a minimum cut in the graph. Of course, such a cut need not be a bisection. But it turns out that for almost every graph in $\mathcal{G}(n,d,b)$ (with b small as stated above), the minimum size bisection is unique and a large number of the source/sink pairs find a cut that contains a large number of edges from this unique optimal bisection. Thus, the minimum size bisection can be identified by keeping track of how often each edge appears in a minimum cut. In practice, the authors found that the algorithm can be made to run quickly; it can get by with a fairly small random sample of source/sink pairs rather than trying every pair (though such an implementation can no longer output a guarantee of optimality). Furthermore, the algorithm almost always outputs the optimal bisection or a bisection that can be made optimal with just one or two local fixes.

It is interesting to note that both of the global algorithms discussed here make heavy use of the fact that for b small enough relative to m, almost every graph in the classes under consideration has a unique minimum bisection.

Other global algorithms employ spectral methods, and these are the main topic of this paper. We will cover two algorithms: one due to Barnes [2] and the other due to Boppana [7]. Other spectral methods related to graph bisection — the lower bounding technique of Barnes and Hoffman [3] and the work of Fiedler [17, 18, 19] relating spectral properties to graph connectivity being notable examples — can be found in the literature, but due to space restrictions, they will not be considered here.

3 Linear algebra preliminaries

Before moving on, we cover some basic facts concerning the eigenvalues and eigenvectors of real, symmetric matrices. This section concludes with two theorems that form the basis for the two algorithms being considered. For all but these last two theorems, we supply no proof; proofs can be found in most any undergraduate linear algebra text such as [31].

Let B be an $n \times n$ matrix and let x be a vector. If $Bx = \lambda x$ where λ is a scalar, then λ is an *eigenvalue* and x is an *eigenvector*. We say that x is an eigenvector associated with the eigenvalue λ. B has n eigenvalues that we denote $\lambda_1, \lambda_2, \ldots, \lambda_n$. Two useful identities are: $\lambda_1 \lambda_2 \cdots \lambda_n = \det(B)$ and $\lambda_1 + \lambda_2 + \cdots + \lambda_n = \text{trace}(B)$.

When the matrix B is real and symmetric, then all n eigenvalues are real. We order them so $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Furthermore, all the eigenvectors are real. In fact, it is possible to find n mutually orthogonal eigenvectors $u_1, u_2, \ldots, u_n$; the indices are given so that u_i is an eigenvector corresponding to the eigenvalue λ_i. We generally take these eigenvectors to be scaled so they are orthonormal.

If we take these n orthonormal eigenvectors and write them into the columns of an $n \times n$ matrix Q, we get the *spectral decomposition*: $B = Q \Lambda Q^T$ where Λ is the diagonal matrix with diagonal entries equal to the eigenvalues, that is, $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. As we will see, this decomposition for symmetric matrices is very useful. Notice that since the matrix Q is orthogonal, $Q^{-1} = Q^T$.

A most remarkable consequence of spectral decomposition is that the maximum value of any quadratic form over a sphere occurs at the eigenvector associated with the largest eigenvalue of the corresponding symmetric matrix. This basic fact is the crux of all the spectral methods considered in this paper. Therefore, we formalize and generalize this fact in the next 3 theorems.

Theorem 1 ([31, Section 6.4]) *Let B be a real, symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Then*

$$\lambda_1 = \max_{x \in \mathbb{R}^n} \frac{x^T B x}{x^T x}$$

with the maximum occurring only when x is an eigenvector of λ_1. And in general for any j in the range $1 \leq j \leq n$,

$$\lambda_j = \min_{S_{n-j+1}} \left\{ \max_{x \in S_{n-j+1}} \frac{x^T B x}{x^T x} \right\}$$

where S_{n-j+1} is any $(n-j+1)$-dimensional subspace of $\mathbb{R}^n$. This minimax occurs only when S_{n-j+1} is the orthogonal complement of the subspace spanned by the $j-1$ eigenvectors associated with the $j-1$ largest eigenvalues, and x is an eigenvector associated with the jth largest eigenvalue. ∎

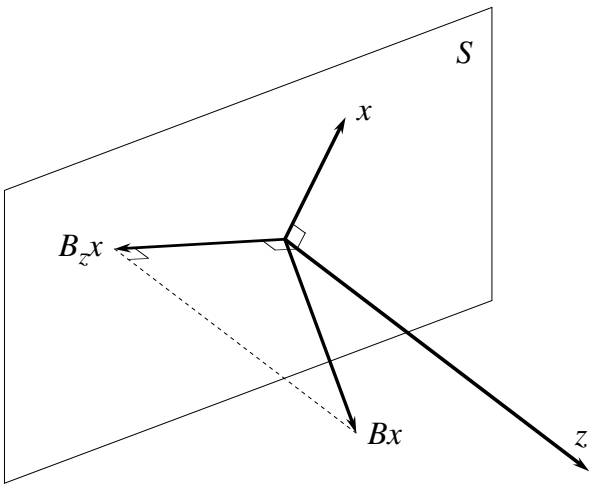

Figure 2: The mapping performed by B_z for the case $n = 3$.

As a simple consequence of this theorem, we have the following useful bounds.

Corollary 2

- $\lambda_1 \geq b_{ii} \geq \lambda_n$ *where* b_{ii} *is a diagonal entry of* B.
- $\lambda_1 \geq \text{sum}(B)/n \geq \lambda_n$ *where* $\text{sum}(B)$ *denotes the sum of all the entries of* B.
- $\lambda_i(cB) = c\lambda_i(B)$ *for* $i = 1, 2, \ldots, n$ *where* c *is a scalar constant.*[1]
- $\lambda_i(A + B) \leq \lambda_i(A) + \lambda_i(B)$ *for* $i = 1, 2, \ldots, n$ *where* A *is also a real, symmetric matrix.*

■

When we examine Boppana's algorithm, we will consider the restriction of B to the orthogonal complement of a vector $z \in \mathbb{R}^n$. Let S denote the orthogonal complement of z. By restricting B to S, we create a new linear operator, which we denote by B_z, from S to S that maps x to the projection of Bx onto S (the point closest to Bx in S). This mapping for the case $n = 3$ is illustrated in Figure 2. The following theorem about the largest eigenvalue of B_z is the key to Boppana's algorithm.

Theorem 3 *Let* B *be a real, symmetric* $n \times n$ *matrix, let* z *be any vector in* $\mathbb{R}^n$, *and let* $\lambda_1(B_z)$ *denote the largest eigenvalue of* B *restricted to the orthogonal complement of* z. *Then*

$$\lambda_1(B_z) = \max_{\substack{x \\ x^T z = 0}} \frac{x^T B x}{x^T x}.$$

Furthermore, this maximum occurs only when x *is an eigenvector of the largest eigenvalue of* B_z.

[1] We use the notation $\lambda(B)$ when we need to be explicit about what matrix the eigenvalue comes from.

Proof: Let S denote the orthogonal complement of z. S has dimension $n-1$ and B_z is a linear operator from S to S. Let $e_1, e_2, \ldots e_{n-1}$ be $n-1$ orthonormal vectors that span S. Using these vectors as a basis, we can write B_z as an $(n-1)\times(n-1)$ matrix: $B_z = P^T B P$ where P is the $n \times (n-1)$ matrix whose columns are the vectors $e_1, e_2, \ldots, e_{n-1}$. Notice that $P^T P = I$ so

$$\begin{aligned} \frac{x^T B_z x}{x^T x} &= \frac{x^T (P^T B P) x}{x^T (P^T P) x} \\ &= \frac{(Px)^T B (Px)}{(Px)^T (Px)} \\ &= \frac{y^T B y}{y^T y} \end{aligned}$$

where $y = Px$. For any $y \in S$, we can write y as a linear combination of the vectors $e_1, e_2, \ldots, e_{n-1}$, and therefore, $y = Px$ for some x. Finally since B_z is symmetric we can use Theorem 1:

$$\begin{aligned} \lambda_1(B_z) &= \max_{x \in S} \frac{x^T B_z x}{x^T x} \\ &= \max_{y \in S} \frac{y^T B y}{y^T y}. \end{aligned}$$

That this maximum occurs only when x is an eigenvector associated with the largest eigenvalue of B_z follows also from Theorem 1. ∎

When generalized, Theorem 3 shows that the eigenvalues of B_z are interleaved with those of B. Specifically,

$$\lambda_1(B) \geq \lambda_1(B_z) \geq \lambda_2(B) \geq \lambda_2(B_z) \geq \cdots \geq \lambda_{n-1}(B_z) \geq \lambda_n(B).$$

Notice that $\lambda_1(B) = \lambda_1(B_z)$ when z is orthogonal to an eigenvector associated with the largest eigenvalue of B. Notice also that $\lambda_1(B_z) = \lambda_2(B)$ when z is orthogonal to an eigenvector associated with the second largest eigenvalue of B. Of course when the largest eigenvalue is repeated, both of these equalities occur simultaneously.

The final theorem of this section is a slight generalization of a theorem due to Fan [15] and is central to the algorithm discussed in the next section.

Theorem 4 ([15]) *Let B be a real, symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_n$. For any positive integer $q \leq n$,*

$$\max_{x_1, \ldots, x_q} \sum_{j=1}^{q} \frac{x_j^T B x_j}{x_j^T x_j} = \sum_{j=1}^{q} \lambda_j$$

with the maximum taken over q mutually orthogonal vectors $x_1, x_2, \ldots, x_q$. Furthermore, this maximum occurs only when the vectors $x_1, x_2, \ldots, x_q$ lie in the q-dimensional subspace determined by the q mutually orthogonal eigenvectors associated with the q largest eigenvalues of B.

Proof: Recall the spectral decomposition: $B = Q \Lambda Q^T$ where Q is the $n \times n$ matrix whose columns contain the n orthonormal eigenvectors $u_1, u_2, \ldots, u_n$ and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. Then

$$\frac{x_j^T B x_j}{x_j^T x_j} = \frac{x_j^T (Q \Lambda Q^T) x_j}{x_j^T x_j}$$

$$
\begin{aligned}
&= \frac{(Q^T x_j)^T \Lambda (Q^T x_j)}{x_j^T x_j} \\
&= \sum_{i=1}^{n} \lambda_i \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 \\
&= \lambda_q \sum_{i=1}^{n} \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 + \sum_{i=1}^{q} (\lambda_i - \lambda_q) \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 + \sum_{i=q+1}^{n} (\lambda_i - \lambda_q) \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 \\
&= \lambda_q + \sum_{i=1}^{q} (\lambda_i - \lambda_q) \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 + \sum_{i=q+1}^{n} (\lambda_i - \lambda_q) \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 \qquad (1)
\end{aligned}
$$

with the last line a consequence of the fact that the u_i being orthonormal implies that

$$\sum_{i=1}^{n} \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 = 1.$$

Looking at the last term of Equation (1), we have $\lambda_i - \lambda_q \leq 0$ for $i > q$. Therefore,

$$\frac{x_j^T B x_j}{x_j^T x_j} \leq \lambda_q + \sum_{i=1}^{q} (\lambda_i - \lambda_q) \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 \qquad (2)$$

with equality only when x_j is orthogonal to every eigenvector that corresponds to an eigenvalue smaller than λ_q.

Summing Inequality (2) over $j = 1, 2, \ldots, q$ and then switching the order of summation, we have

$$\sum_{j=1}^{q} \lambda_j - \sum_{j=1}^{q} \frac{x_j^T B x_j}{x_j^T x_j} \geq \sum_{i=1}^{q} (\lambda_i - \lambda_q) \left[1 - \sum_{j=1}^{q} \left(\frac{u_i^T x_j}{\|x_j\|} \right)^2 \right]. \qquad (3)$$

Since $\lambda_i - \lambda_q \geq 0$ for $i \leq q$ and $\sum_{j=1}^{q} (u_i^T x_j / \|x_j\|)^2 \leq \sum_{j=1}^{n} (u_i^T x_j / \|x_j\|)^2 = 1$, the right-hand size of Inequality (3) is at least 0. The left-hand size of Inequality (3) equals 0 when $x_j = u_j$ for $j = 1, 2, \ldots, q$, and this establishes the maximum property.

The right-hand size of Inequality (3) equals 0 only when $\sum_{j=1}^{q} (u_i^T x_j / \|x_j\|)^2 = 1$ for every $i < q$ with $\lambda_i > \lambda_q$. This fact combined with the observation about when Inequality (2) achieves equality serves to establish that the maximum occurs only when $x_1, x_2, \ldots, x_q$ lie in the q-dimensional subspace determined by q mutually orthogonal eigenvectors associated with the q largest eigenvalues of B. ■

The special case of this theorem with $q = 2$ and $x_j^T x_j = n/2$ plays a lead role in the next section.

4 Bisection in n dimensions

In this section we consider a graph bisection algorithm due to Barnes [2]. We first show that a bisection can be represented as two orthogonal vectors with a special form in n-dimensional space and that finding a minimum size bisection is the same as minimizing a quadratic function of these two vectors. This quadratic function takes on its minimum value at the eigenvectors associated

with the two largest eigenvalues of the adjacency matrix. The algorithm proposed by Barnes is then nothing more than finding a bisection as close to these eigenvectors as possible.

We start with some notation. Consider a graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$, and let $A = (a_{ij})$ denote the $n \times n$ adjacency matrix of G. For simplicity, assume n is even. We are dealing with undirected graphs so A is symmetric. Now let D denote any diagonal $n \times n$ matrix, and let $d_1, d_2, \ldots, d_n$ denote its diagonal entries — we will discuss the choice of D later. Finally, let the $n \times n$ symmetric matrix $B = (b_{ij})$ be given by $B = A + D$ (so $b_{ii} = d_i$ and $b_{ij} = a_{ij}$ for $i \neq j$).

We can represent a bisection (V_1, V_2) of G with two n-dimensional 0-1 vectors x and $\overline{x}$ as follows: for $i = 1, 2, \ldots, n$, let x_i equal 1 if vertex i is in V_1 and 0 otherwise; likewise, let $\overline{x}_i$ equal 1 if vertex i is in V_2 and 0 otherwise (since (V_1, V_2) is a partition, $\overline{x}_i = 1 - x_i$). Notice that the vectors x and $\overline{x}$ are orthogonal, that is, $x^T \overline{x} = 0$. Notice also that x and $\overline{x}$ both have Euclidean norm $\sqrt{n/2}$, that is, $\|x\|^2 = x^T x = n/2$ and $\|\overline{x}\|^2 = \overline{x}^T \overline{x} = n/2$. Let $\mathrm{b}(G, (x, \overline{x}))$ denote the size of the bisection.

We now show that the bisection size is a quadratic function of x and $\overline{x}$. Observe that for an edge (i, j), the product $x_i x_j$ equals 1 if (i, j) is an edge internal to V_1 and otherwise it is 0; likewise for $\overline{x}_i \overline{x}_j$. Therefore, $\sum_{ij} a_{ij}(x_i x_j + \overline{x}_i \overline{x}_j)$ equals twice the number of edges *not* cut by the bisection.[2] Notice also that the sum of all the entries in the matrix A (a quantity we denote by $\mathrm{sum}(A)$) equals twice the number of edges in the graph, that is, $\mathrm{sum}(A) = \sum_{ij} a_{ij} = 2\,|E|$. Now the following equality gives an explicit formula for the bisection size $\mathrm{b}(G, (x, \overline{x}))$.

$$\begin{aligned} x^T B x + \overline{x}^T B \overline{x} &= \sum_{ij} b_{ij}(x_i x_j + \overline{x}_i \overline{x}_j) \\ &= \sum_{i} d_i(x_i^2 + \overline{x}_i^2) + \sum_{i \neq j} a_{ij}(x_i x_j + \overline{x}_i \overline{x}_j) \\ &= \sum_{i} d_i + 2\,(|E| - \mathrm{b}(G, (x, \overline{x}))) \\ &= \mathrm{sum}(B) - 2\,\mathrm{b}(G, (x, \overline{x})). \end{aligned}$$

Thus, we have our promised quadratic function:

$$\mathrm{b}(G, (x, \overline{x})) = \frac{1}{2}\,\mathrm{sum}(B) - \frac{1}{2}\,(x^T B x + \overline{x}^T B \overline{x})\,. \tag{4}$$

Finding a minimum bisection is therefore a matter of minimizing the above equality — or equivalently maximizing $x^T B x + \overline{x}^T B \overline{x}$ — subject to the condition that the vectors x and $\overline{x}$ actually represent a bisection, that is, that x and $\overline{x}$ are 0-1 vectors with exactly $n/2$ coordinates being 1 and $\overline{x} = \vec{1} - x$.

Of course all we have done so far is restate the problem; we have not made life any easier yet. However if we ease the restrictions on x and $\overline{x}$, then dealing with the quadratic becomes considerably easier. In particular, we know from Theorem 4 of Section 3 the following bound.

$$\max_{\substack{x^T \overline{x} = 0 \\ \|x\|^2 = \|\overline{x}\|^2 = n/2}} (x^T B x + \overline{x}^T B \overline{x}) \;=\; \frac{n}{2}\,(\lambda_1(B) + \lambda_2(B))\,. \tag{5}$$

Furthermore, we know that the maximum occurs only when x and $\overline{x}$ lie in the plane defined by two eigenvectors associated with the two largest eigenvalues of B.

As a consequence of this equality, we have the following lower bound on bisection width.

[2] When explicit summation limits are left out, each index variable is being summed over the entire range from 1 to n.

Theorem 5 *Let G be an n-vertex graph with adjacency matrix A. Let D be any diagonal matrix, and let $B = A + D$. The bisection width of G is then lower bounded by*

$$\mathrm{b}(G) \geq \frac{1}{2}\,\mathrm{sum}(B) - \frac{n}{4}\left(\lambda_1(B) + \lambda_2(B)\right).$$

Furthermore, this bound is a concave function of D.

Proof: Let the vectors x and $\overline{x}$ denote a minimum size bisection of G. As already pointed out, $x^T\overline{x} = 0$ and $\|x\|^2 = \|\overline{x}\|^2 = n/2$. Therefore, Equation (5) together with Equation (4) establishes the bound.

To see that this bound is a concave function of D, we show that each eigenvalue $\lambda_i(B)$ for $i = 1, 2, \ldots, n$ is a convex function of D. From the facts (see Corollary 2) that $\lambda_i(A + D) \leq \lambda_i(A) + \lambda_i(D)$ and for any scalar $c \neq 0$, $\lambda_i(cA) = c\lambda_i(A)$, we have

$$\begin{aligned}
\lambda_i\left(A + \frac{1}{2}(D_1 + D_2)\right) &= \lambda_i\left(\frac{1}{2}(A + D_1) + \frac{1}{2}(A + D_2)\right) \\
&\leq \frac{1}{2}\left(\lambda_i(A + D_1) + \lambda_i(A + D_2)\right).
\end{aligned}$$

■

When D is chosen so that $\mathrm{sum}(B) = 0$, the bound of Theorem 5 is the same as the bound given by Donath and Hoffman [13, Theorem 1].

As we have already observed, the bound of Equation (5) achieves equality at the pair of eigenvectors (u_1, u_2) of the two largest eigenvalues of B. Thus, if u_1 and u_2 turn out to have the special form of a bisection, then they represent an optimal bisection. Of course, u_1 and u_2 may not have this form; all we know is that they are orthogonal and we can scale them to have Euclidean norm $\sqrt{n/2}$.

In order to produce a bisection, Barnes [2] proposes picking a bisection as close to the eigenvectors as possible. That is, he proposes picking a bisection $(x, \overline{x})$ that minimizes $\|x - u_1\|^2 + \|\overline{x} - u_2\|^2$. Let $u_1 = [u_{1,1}, u_{2,1}, \ldots, u_{n,1}]^T$ and let $u_2 = [u_{1,2}, u_{2,2}, \ldots, u_{n,2}]^T$. Since $\overline{x} = \vec{1} - x$, we have

$$\begin{aligned}
\|x - u_1\|^2 + \|\overline{x} - u_2\|^2 &= \|x\|^2 - 2x^T u_1 + \|u_1\|^2 + \|\overline{x}\|^2 - 2\overline{x}^T u_2 + \|u_2\|^2 \\
&= 2n - 2\sum_i (x_i u_{i,1} + (1 - x_i) u_{i,2}) \\
&= 2n - 2\sum_i u_{i,2} - 2\sum_i w_i x_i
\end{aligned}$$

where the vector w is given by $w = u_1 - u_2$. Thus, we minimize $\|x - u_1\|^2 + \|\overline{x} - u_2\|^2$ subject to $(x, \overline{x})$ denoting a bisection by maximizing $\sum_i w_i x_i$ subject to x having exactly $n/2$ coordinates with value 1 and the other $n/2$ coordinates having the value 0. Clearly this maximization is accomplished by setting $x_i = 1$ in the $n/2$ largest coordinates of w and setting $x_i = 0$ in the $n/2$ smallest coordinates of w (breaking ties arbitrarily).

To summarize, the Barnes algorithm is the following:

1. Let A denote the adjacency matrix of the n-vertex graph G. Let D be any diagonal matrix, and let $B = A + D$.

2. Compute the two eigenvectors, u_1 and u_2, corresponding to the two largest eigenvalues of B, and let $w = u_1 - u_2$.

3. Place the $n/2$ vertices corresponding to the $n/2$ largest coordinates of w on one side of the bisection, and place the other $n/2$ vertices on the other side.

Two issues remain unresolved: how to choose D and which eigenvectors to use.

First consider choosing which eigenvectors to use. When the two largest eigenvalues are unique, that is when $\lambda_1(B) \neq \lambda_2(B) \neq \lambda_3(B)$, then the two eigenvectors are also unique (up to a scalar constant factor). We assume that both eigenvectors, u_1 and u_2, are scaled to have Euclidean norm $\sqrt{n/2}$, so the only choice is the sign of each vector. This actually gives two different choices for w: either $w = u_1 - u_2$ or $w = u_1 + u_2$. (Notice that $w = -u_1 + u_2$ gives the same bisection as $w = u_1 - u_2$, and likewise $w = -u_1 - u_2$ gives the same bisection as $w = u_1 + u_2$.) In this case, Barnes suggests trying both bisections and picking the better one. The situation is much worse when the largest eigenvalues are repeated. In this case, Barnes offers no help — just pick any two of the eigenvectors corresponding to the two largest eigenvalues and check both bisections as just described. Trying a small random sample of eigenvector pairs might improve this situation.

Choosing a matrix D presents less difficulty. Barnes suggests using nothing more than $D = 0$. If we look ahead to the next section, however, we find another choice in the technique used by Boppana [7]. Since the lower bound of Theorem 5 is a concave function of D, the ellipsoid method can be used to find a D that maximizes this lower bound [23]. Let $\widehat{D}$ denote such a maximizing matrix, and let $\widehat{B} = A + \widehat{D}$. If $\widehat{B}$ actually produces equality in the lower bound of Theorem 5, that is, if

$$\frac{1}{2}\operatorname{sum}(\widehat{B}) - \frac{n}{4}\left(\lambda_1(\widehat{B}) + \lambda_2(\widehat{B})\right) = \mathrm{b}(G), \tag{6}$$

then since Equation (5) achieves equality only in the plane of two eigenvectors associated with the two largest eigenvalues of $\widehat{B}$, it must be the case that any pair of vectors $(x, \overline{x})$ defining a minimum size bisection lie in such a plane.

This situation is particularly favorable when the two largest eigenvalues of $\widehat{B}$ are unique, since in this case the plane is unique. Somewhere in this plane lies an optimal bisection; of course finding this bisection may not be so easy. In any case, these facts indicate that $\|x - u_1\|^2 + \|\overline{x} - u_2\|^2$ may not be the best choice of metric to minimize. We want a bisection that is closest to the plane defined by u_1 and u_2, not necessarily the bisection closest to these actual eigenvectors. Unfortunately, metrics that capture this desire are quadratic and therefore difficult to minimize over the constraint that $(x, \overline{x})$ is a bisection.

5 Bisection in $n-1$ dimensions

Rather than use two orthogonal vectors to represent a bisection, Boppana [7] uses a single vector. In particular, the bisection (V_1, V_2) is represented by the vector y where for $i = 1, 2, \ldots, n$, coordinate y_i is set to $+1$ if vertex i is in V_1 and to -1 if in V_2. Again for simplicity, assume n is even. Notice that $\sum_i y_i = 0$ and the set of vectors with this property is the $(n-1)$-dimensional subspace orthogonal to the 1-vector. Notice also that $\|y\| = \sqrt{n}$.

We now exhibit Boppana's algorithm by paralleling the development of the previous section. Again let A denote the adjacency matrix, let $D = \operatorname{diag}(d_1, d_2, \ldots, d_n)$ be any diagonal matrix, and let $B = A + D$.

As before, we start by writing the bisection size $\mathrm{b}(G, y)$ as a quadratic function of y. For an edge (i, j), the product $y_i y_j$ equals 1 if (i, j) is *not* cut by the bisection and -1 if it is. Thus,

$$\begin{aligned} y^T B y &= \sum_{ij} b_{ij} y_i y_j \\ &= \sum_i d_i y_i^2 + \sum_{i \neq j} a_{ij} y_i y_j \\ &= \sum_i d_i + 2\left(|E| - \mathrm{b}(G, y)\right) - 2\,\mathrm{b}(G, y) \\ &= \mathrm{sum}(B) - 4\,\mathrm{b}(G, y). \end{aligned}$$

And we have our quadratic function:

$$\mathrm{b}(G, y) = \frac{1}{4}\mathrm{sum}(B) - \frac{1}{4} y^T B y. \tag{7}$$

Again, we have just restated the problem: minimize this quadratic function subject to y actually representing a bisection. Easing this restriction on y, we have the following bound from Theorem 3 of Section 3.

$$\max_{\substack{y \\ y^T \vec{1} = 0 \\ \|y\|^2 = n}} y^T B y \;=\; n \lambda_1(B_{\vec{1}}) \tag{8}$$

where $B_{\vec{1}}$ is the restriction of B to the orthogonal complement of the 1-vector. Furthermore, this maximum occurs only when $y = v_1$, an eigenvector associated with the largest eigenvalue of $B_{\vec{1}}$.

Once more, we obtain a lower bound on the bisection width.

Theorem 6 ([7, Theorem 3.2]) *Let G be an n-vertex graph with adjacency matrix A. Let D be any diagonal matrix, and let $B = A + D$. The bisection width of G is then lower bounded by*

$$\mathrm{b}(G) \geq \frac{1}{4}\mathrm{sum}(B) - \frac{n}{4}\lambda_1(B_{\vec{1}}).$$

Furthermore, this bound is a concave function of D.

Proof: Let y denote a minimum bisection of G. As already noted, $y^T \vec{1} = 0$ and $\|y\|^2 = n$. Therefore, Equation (8) together with Equation (7) establishes the bound. The concavity follows as in the proof of Theorem 5. ∎

Rather than simply approximate the eigenvector which may not be a bisection, Boppana hopes to use D in order to produce an eigenvector that is a bisection. As already pointed out, the concavity of the lower bound enables the ellipsoid method to find a D that maximizes the lower bound [23]. Let $\widehat{D}$ denote such a maximizing value, let $\widehat{B} = A + \widehat{D}$, and let $\widehat{B}_{\vec{1}}$ denote the restriction of $\widehat{B}$ to the orthogonal complement of the 1-vector. Boppana hopes that this maximization makes the bound tight, that is

$$\frac{1}{4}\mathrm{sum}(\widehat{B}) - \frac{n}{4}\lambda_1(\widehat{B}_{\vec{1}}) = \mathrm{b}(G). \tag{9}$$

In fact, we will see in the next section that for certain classes of random graphs, such a $\widehat{D}$ almost always exists.

Boppana claims [7] that whenever Equation (9) is satisfied, a minimum size bisection can be output as follows: let v_1 denote an eigenvector corresponding to the largest eigenvalue of $\widehat{B}_{\vec{1}}$, then place the $n/2$ vertices corresponding to the $n/2$ largest components of v_1 on one side of the bisection and the other $n/2$ on the other side. This is not quite true. We know that the maximum of Equation (8) only occurs when y is an eigenvector corresponding to the largest eigenvalue. Thus, when Equation (9) is satisfied, any minimum size bisection must be an eigenvector corresponding to the largest eigenvalue of $\widehat{B}_{\vec{1}}$. When this largest eigenvalue is unique, then so is the eigenvector (up to a scalar constant factor). Therefore in this case, v_1 must be the unique minimum size bisection. The problem occurs when the largest eigenvalue is repeated. In this case, if the largest eigenvalue is repeated k times then the set of corresponding eigenvectors forms a k-dimensional subspace. Somewhere in this subspace lies a minimum size bisection, and in fact, any bisection in this subspace must be of minimum size. When $k > 1$, finding a bisection in this eigenspace may not be so easy.

To remedy this situation, Boppana proposes finding a minimum bisection in the structure of the set of matrices D that make the lower bound of Theorem 6 tight [8]. This set can be found by the ellipsoid method as it performs the maximization. We will show, however, that this solution is plagued by a problem similar to the one just described.

To illustrate the structure of the matrices D that make the lower bound tight, we first show that shifting D by a constant multiple of the identity matrix does not change the value of the lower bound.

Lemma 7 ([8]) *Let D be any $n \times n$ diagonal matrix and let c be any scalar constant. Set $D' = D + cI$ and $B = A + D$ and $B' = A + D'$. Then*

$$\frac{1}{4}\,\mathrm{sum}(B') - \frac{n}{4}\lambda_1(B'_{\vec{1}}) = \frac{1}{4}\,\mathrm{sum}(B) - \frac{n}{4}\lambda_1(B_{\vec{1}})$$

where $B_{\vec{1}}$ is the restriction of B and $B'_{\vec{1}}$ is the restriction of B' to the subspace orthogonal to the 1-vector.

Proof: We show that both $\mathrm{sum}(B)$ and $n\lambda_1(B_{\vec{1}})$ increase by nc when we add cI to B. The first of these facts is obvious: $\mathrm{sum}(B') = \mathrm{sum}(B) + nc$. The other fact follows because applying cI to any vector simply stretches it by a factor of c; therefore, $\lambda_1(B'_{\vec{1}}) = \lambda_1(B_{\vec{1}}) + c$. ∎

Given that we can add any scalar multiple of the identity matrix to D without changing the value of the lower bound, we consider only those matrices D for which $\lambda_1(B_{\vec{1}}) = 0$. Such a matrix is referred to as *shifted*.

We want to look at those matrices D that make the lower bound tight, that is, those that satisfy Equation (9) (with $\widehat{D} = D$). Such a matrix is referred to as *maximal*. It is convenient to view the diagonal matrix D as a vector $d = [d_1, d_2, \ldots, d_n]^T$. We now characterize the set of shifted maximal vectors. Refer to Figure 3.

Lemma 8 ([8]) *The set of shifted maximal vectors forms a line segment.*

Proof: Let d be a shifted maximal vector, let $D = \mathrm{diag}(d)$ be the corresponding diagonal matrix, and let $B = A + D$. Since d is shifted, we have $\lambda_1(B_{\vec{1}}) = 0$. Also, since d is maximal, for any minimum size bisection y, it must be the case that y is an eigenvector corresponding the largest

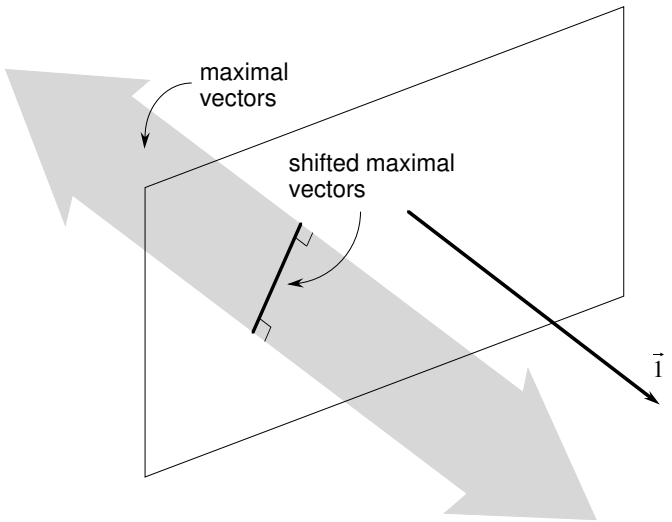

Figure 3: The set of shifted maximal vectors forms a bounded line segment.

eigenvalue of $B_{\vec{1}}$, which as just mentioned is 0. Thus, $B_{\vec{1}}y = 0$ or equivalently $By = c\vec{1}$ for some scalar constant c. Since $(A + D)y = c\vec{1}$, we have $Dy = c\vec{1} - Ay$. If we look at the ith coordinate of this vector identity, we have $d_i y_i = c - (Ay)_i$. Multiplying both sides by y_i we get

$$d_i = cy_i - (Ay)_i y_i. \tag{10}$$

Since for every shifted maximal vector d there must be a scalar c such that the above identity is satisfied, the set of shifted maximal vectors lie on a line with a slope that identifies the minimum size bisection.

To see that this line is actually a line segment, consider changing c to $c' = c + \epsilon$. Let $\mathrm{diag}(y)$ denote the diagonal matrix with diagonal entries given by the coordinates of y. Changing c to c' changes d to $d' = d + \epsilon y$, and likewise, D to $D' = D + \epsilon\,\mathrm{diag}(y)$ and B to $B' = B + \epsilon\,\mathrm{diag}(y)$. As long as $\lambda_1(B'_{\vec{1}})$ is still 0, then d' is still a shifted maximal vector (notice that $\mathrm{sum}(B') = \mathrm{sum}(B)$). Since $B'y = c'$, the vector y is still an eigenvector and 0 is still an eigenvalue. The eigenvectors of $B'_{\vec{1}}$ are continuous functions of ϵ so we expect that as we vary ϵ, the other eigenvalues will move around (one eigenvalue at 0 stays put) with one of them eventually becoming greater than 0. In fact, this does happen; we now show that for ϵ positive enough or negative enough, we have $\lambda_1(B'_{\vec{1}}) > 0$.

$$\begin{aligned}
\lambda_1(B'_{\vec{1}}) &= \max_{\substack{z \\ z^T\vec{1}=0 \\ \|z\|=1}} z^T B' z \\
&= \max_{\substack{z \\ z^T\vec{1}=0 \\ \|z\|=1}} \left(z^T B z + \epsilon z^T\,\mathrm{diag}(y) z\right).
\end{aligned}$$

Since $\lambda_1(B_{\vec{1}}) = 0$, we know that for any vector z with $z^T\vec{1} = 0$, we have $z^T B z \le 0$. Thus, for any vector z with $z^T\,\mathrm{diag}(y)z > 0$, by setting ϵ large enough we get $\lambda_1(B'_{\vec{1}}) > 0$. Also, for any vector z with $z^T\,\mathrm{diag}(y)z < 0$, by setting ϵ negative enough we again get $\lambda_1(B'_{\vec{1}}) > 0$. Hence, the set of shifted maximal vectors forms a bounded line segment, the slope of which gives the minimum size bisection. ∎

Boppana hoped to conclude from these two lemmas that if a maximal vector exists, then a minimum size bisection can be output as the slope of the set of shifted maximal vectors. Of course the problem with this is probably obvious by now: if the largest eigenvalue of $B_{\vec{1}}$ is repeated, then it is possible that the set of shifted maximal vectors degenerates to a single point. Notice that the scheme originally proposed by Boppana never works when the largest eigenvalue is repeated whereas this new scheme can work in some of these cases; the new scheme only fails when the largest eigenvalue is repeated *and* the set of shifted maximal vectors degenerates to a single point. The only way out of this dilemma is to resort to the probabilistic analysis of the next section in which we show that for certain classes of random graphs a shifted maximal vector does exist (with high probability) and further that the largest eigenvalue of $B_{\vec{1}}$ is unique; this uniqueness guarantees that the set of shifted maximal vectors is not degenerate.

A simple consequence of the above analysis is that the strategy proposed by Boppana can never work for graphs with more than one minimum size bisection. The classes of random graphs under consideration better not contain many graphs with multiple optimal bisections.

Before moving on to the probabilistic analysis, we summarize Boppana's algorithm as it stands now.

1. Maximize $\text{sum}(A + D) - n\lambda_1((A + D)_{\vec{1}})$ as a function of D to compute the set of shifted vectors that maximize this quantity.

2. If this set of vectors is a line segment and not just a point, then let y denote the vector giving the slope of this line segment. Place the $n/2$ vertices corresponding to the largest $n/2$ components of y on one side of the bisection and the other $n/2$ vertices on the other side.

3. Otherwise the algorithm has failed. Either output nothing or resort to the techniques of the previous section and try to find a bisection that comes close to some eigenvector.

6 Average case analysis

To extricate ourselves from the jam we found ourselves in at the close of the last section, we now show that Boppana's algorithm does work for certain classes of random graphs. Recall the definition of $\mathcal{G}(n, m, b)$: a uniform distribution on n-vertex graphs with m edges and bisection width b. For b small enough relative to m, these graphs almost always have a unique minimum size bisection. Furthermore, we will show that these graphs almost always have a non-degenerate set of shifted maximal vectors. This fact will establish the efficacy of Boppana's algorithm (for this class of random graphs) since as we saw in Lemma 8, when the set of shifted maximal vectors is not degenerate, it is a line segment whose slope determines the unique minimum size bisection.

Before stating and proving this final result, we state without proof two lemmas that will soon be of value. The first is a Chernoff bound.

Lemma 9 ([1, Appendix A]) *Let the random variable X be a sum of random variables, $X = X_1 + X_2 + \cdots + X_n$, with $\Pr\{X_i = 1 - p_i\} = p_i$ and $\Pr\{X_i = -p_i\} = 1 - p_i$ for $i = 1, 2, \ldots, n$. Then for any $\alpha > 0$,*

$$\Pr\{X > \alpha\} < e^{-2\alpha^2/n}.$$

■

The second lemma bounds the eigenvalues of random symmetric matrices and is due to Füredi and Komlós.

Lemma 10 ([20, Theorem 2]) *Let $A = (a_{ij})$ be a random symmetric $n \times n$ matrix with entries a_{ij} independent (up to symmetry), not necessarily identically distributed, with a common expectation $\mu = 0$, and variance bounded by σ^2. Then with probability $1 - o(1)$,*

$$\max_{1 \le i \le n} |\lambda_i(A)| = 2\sigma\sqrt{n} + O(n^{1/3} \log n).$$

■

We now state and sketch the proof of a slightly weaker version of Boppana's main result.[3]

Theorem 11 ([7, Theorem 4.1]) *Let G be a random graph from $\mathcal{G}(n, m, b)$ with*

$$b \le \frac{1}{2}m - \frac{1}{2}\sqrt{n^3 \ln n}.$$

Then with probability at least $1 - o(1)$, Boppana's algorithm outputs the unique minimum size bisection of G.

Proof sketch: Recalling the discussion of the last section, it suffices to show that with high probability G has a shifted maximal vector such that $\lambda_1(B_{\vec{1}}) = 0$ is the unique largest eigenvalue. Specifically, we need to show that there exists a diagonal matrix D such that

$$\frac{1}{4}\operatorname{sum}(B) - \frac{n}{4}\lambda_1(B_{\vec{1}}) = \mathrm{b}(G)$$

and $\lambda_2(B_{\vec{1}}) < \lambda_1(B_{\vec{1}}) = 0$.

Rather than work directly with the distribution $\mathcal{G}(n, m, b)$ we consider the closely related distribution $\mathcal{G}(n, p, q)$ with

$$p = \frac{m - b}{\frac{n}{2}\left(\frac{n}{2} - 1\right)} \qquad \text{and} \qquad q = \frac{b}{\left(\frac{n}{2}\right)^2}. \tag{11}$$

This is the distribution on random graphs with vertex set $V = \{1, 2, \ldots, n\}$ and edges determined by the following process:

1. Pick a random partition (V_1, V_2) with half of the vertices in V_1 and the other half in V_2.
2. For each pair of vertices (i, j) with $i \in V_1$ and $j \in V_2$, create an edge (i, j) with probability q.
3. For each pair of vertices (i, j) with $i, j \in V_1$ or $i, j \in V_2$, create an edge (i, j) with probability p.

That certain results about this distribution — in particular, the result we need — can be translated to $\mathcal{G}(n, m, b)$ follows from results of Bollobás [6, Section II.1].

For ease of exposition, we modify the random process one more time. Rather than picking the partition (V_1, V_2) at random, let $V_1 = \{1, 2, \ldots, n/2\}$ and $V_2 = \{n/2 + 1, \ldots, n\}$. A random graph from $\mathcal{G}(n, p, q)$ can be obtained from this new distribution by randomly relabeling the vertices.

[3] I believe the result as stated by Boppana is in error due to a small mistake in employing a Chernoff bound.

This relabeling does not change the eigenvalues. Thus, if for almost every graph generated by this modified process, we can produce a shifted maximal vector such that the largest eigenvalue is unique, then the desired result follows.

Consider a random n-vertex graph G with adjacency matrix A created by the modified process just described, and consider the bisection (V_1, V_2) which is represented by the vector $y = [y_1, y_2, \ldots, y_n]^T$ where $y_i = +1$ for $1 \leq i \leq n/2$ and $y_i = -1$ for $n/2 + 1 \leq i \leq n$. Now consider the vector $d = [d_1, d_2, \ldots, d_n]^T$ where d_i for $i = 1, 2, \ldots, n/2$ equals the number of vertices in V_2 adjacent to vertex i minus the number of vertices in V_1 adjacent to vertex i, and d_i for $i = n/2+1, \ldots, n$ equals the negation of this expression. That is, $d_i = -(Ay)_i y_i$; recall Equation (10). We will show that with high probability d is a shifted maximal vector and that $\lambda_1(B_{\vec{1}}) = 0$ is the unique largest eigenvalue of $B_{\vec{1}}$, the restriction of $B = A + \mathrm{diag}(d)$ to the orthogonal complement of the 1-vector.

Notice that $\mathrm{sum}(B) = 4\,\mathrm{b}(G, y)$ and that 0 is an eigenvalue of $B_{\vec{1}}$ with eigenvector y. Thus, we just need to show that with high probability 0 is the unique largest eigenvalue of $B_{\vec{1}}$.

We now examine the expected value $\mathrm{E}[B]$ of the random matrix $B = A + \mathrm{diag}(d)$. Recall that the random matrix $A = (a_{ij})$ is given by $\Pr\{a_{ij} = 1\} = p$ for either $1 \leq i, j \leq n/2$ or $n/2+1 \leq i, j \leq n$ and $\Pr\{a_{ij} = 1\} = q$ for every other entry. Clearly, the expected value $\mathrm{E}[A]$ of A is the matrix with p in every entry of the upper-left and lower-right quarters and q in every other entry. When $n = 6$, for example,

$$\mathrm{E}[A] = \left[\begin{array}{ccc|ccc} p & p & p & q & q & q \\ p & p & p & q & q & q \\ p & p & p & q & q & q \\ \hline q & q & q & p & p & p \\ q & q & q & p & p & p \\ q & q & q & p & p & p \end{array}\right].$$

Also, since d is given by $d_i = -(Ay)_i y_i$, we have $\mathrm{E}[d_i] = (q - p)n/2$. Therefore,

$$\mathrm{E}[B] = \mathrm{E}[A] - \frac{n}{2}(p - q)I. \tag{12}$$

To determine whether or not $B_{\vec{1}}$ has a second eigenvalue that is greater than or equal to 0, we only have to observe the behavior of $B_{\vec{1}}$ on vectors orthogonal to y. Since y is an eigenvector associated with the eigenvalue 0 of $B_{\vec{1}}$, if $B_{\vec{1}}$ has another eigenvalue that is greater than or equal to 0, then it has an associated eigenvector that is orthogonal to y. Now notice that for any vector x orthogonal to y, we have $(\mathrm{E}[A])x = 0$. Therefore, rather than observe $B_{\vec{1}}$, we can observe $(B - \mathrm{E}[A])_{\vec{1}}$ on vectors orthogonal to y. From Equation (12), $B - \mathrm{E}[A] = B - \mathrm{E}[B] - \frac{n}{2}(p - q)I$. Then since $\lambda_1(B - \mathrm{E}[B]) \geq \lambda_1((B - \mathrm{E}[B])_{\vec{1}})$ (look at Theorem 3), to show that $B_{\vec{1}}$ has only one non-negative eigenvalue, we just need to show that $\lambda_1(B - \mathrm{E}[B]) < \frac{n}{2}(p - q)$.

To bound $\lambda_1(B - \mathrm{E}[B])$ we use

$$\lambda_1(B - \mathrm{E}[B]) \leq \lambda_1(A - \mathrm{E}[A]) + \lambda_1(D - \mathrm{E}[D]) \tag{13}$$

where $D = \mathrm{diag}(d)$.

The eigenvalues of $D - \mathrm{E}[D]$ are the diagonal entries $d_i - \mathrm{E}[d_i]$ for $i = 1, 2, \ldots, n$. From the Chernoff bound of Lemma 9, we have $d_i < \sqrt{n \ln n}$ with probability at least $1 - 1/n^2$. Therefore,

$$\lambda_1(D - \mathrm{E}[D]) < \sqrt{n \ln n} \tag{14}$$

with probability at least $1 - 1/n$.

To bound $\lambda_1(A - \mathrm{E}[A])$, we use Lemma 10. Since for the random variable $a_{ij} - \mathrm{E}[a_{ij}]$ the mean is 0 and the variance is bounded by $\sigma^2 \leq \max\{p(1-p), q(1-q)\} \leq 1$, we have

$$\lambda_1(A - \mathrm{E}[A]) \leq 2\sqrt{n} + O(n^{1/3} \log n) < \sqrt{n \ln n} \tag{15}$$

with probability at least $1 - o(1)$ and for large enough n.

Substituting Inequalities (14) and (15) into Inequality (13), we have

$$\lambda_1(B - \mathrm{E}[B]) < 2\sqrt{n \ln n}$$

with probability at least $1 - o(1)$. For $m - 2b \geq \sqrt{n^3 \ln n}$ and the values of p and q given in Equation (11), we have $2\sqrt{n \ln n} \leq \frac{n}{2}(p - q)$ and this gives the desired result. ∎

7 Conclusions

We have looked at two very similar algorithms. Both express bisection size as a quadratic function, minimize this function over a sphere by locating the eigenvectors associated with the largest eigenvalue of the graph's adjacency matrix, and then use these eigenvectors to guide the search for a bisection. Along with a bisection, both algorithms can output a lower bound that can be used to check whether the output bisection is optimal. On the down side, we have seen that Boppana's algorithm cannot find a minimum bisection in a graph with more than one optimal bisection, and a similar statement is true for Barnes' algorithm. Since two different bisections cannot lie in the same plane, even if Barnes' algorithm can make the lower bound tight, the two largest eigenvalues must be non-unique, and as we have seen, this situation presents some difficulties. Of course with either algorithm, if we resort to simply trying to find a bisection that is close to an actual eigenvector, then we might get lucky.

Contrasting the two algorithms, Barnes expresses a bisection with two orthogonal vectors and performs the minimization in n-dimensional space whereas Boppana uses a single vector and performs the minimization in $(n-1)$-dimensional space. Boppana's approach seems to give more control over the eigenvectors as Boppana is able to show that his algorithm finds the optimal bisection for almost every graph in certain limited classes; no such result appears forthcoming from the approach used by Barnes. On the other hand, Barnes' approach readily generalizes to the graph partitioning problem in which we may want to split the graph into unequal size pieces or more than two pieces.

The fact that Boppana's algorithm provably works well for certain classes of random graphs leaves room for optimism, but this average case analysis does not tell the whole story. To begin with, the classes considered are extremely limited; they require fairly dense graphs with very small bisections. More importantly, even though the algorithm may work well for almost every graph in a certain class, it may be the case that the graphs arising in practice are the ones on which the algorithm fails. This scenario has been observed in other realms; for example, even though greedy routing algorithms are known to work well for almost every routing problem, the routing problems that often arise in practice are exactly those problems on which greedy routing performs poorly [26, Sections 3.4.2 and 3.4.4]. We can hope that the situation with Boppana's algorithm is more benevolent, but the fact that it always fails on graphs with more than one optimal bisection may indicate otherwise.

References

[1] Noga Alon and Joel H. Spencer. *The Probabilistic Method.* John Wiley & Sons, 1992.

[2] Earl R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic and Discrete Methods*, 3(4):541–550, December 1982.

[3] Earl R. Barnes and Alan J. Hoffman. Partitioning, spectra and linear programming. *Progress in Combinatorial Optimization*, pages 13–25, 1984.

[4] Earl R. Barnes, Anthony Vannelli, and James Q. Walker. A new heuristic for partitioning the nodes of a graph. *SIAM Journal on Discrete Mathematics*, 1(3):299–305, August 1988.

[5] Sandeep N. Bhatt and Frank Thomson Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, April 1984.

[6] Béla Bollobás. *Random Graphs.* Academic Press, 1985.

[7] Ravi B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 280–285, Los Angeles, California, October 1987.

[8] Ravi B. Boppana. Private communication, January 1993.

[9] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[10] Thang Nguyen Bui. On bisecting random graphs. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, January 1983. Also: MIT Laboratory for Computer Science Technical Report MIT/LCS/TR-287.

[11] Thang Nguyen Bui. *Graph Bisection Algorithms.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, January 1986.

[12] Peter J. Denning. Virtual memory. *ACM Computing Surveys*, 2(3):153–189, September 1970.

[13] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, September 1973.

[14] M. E. Dyer and A. M. Frieze. Fast solutions of some random NP-hard problems. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 331–336, October 1986.

[15] Ky Fan. On a theorem of Weyl concerning eigenvalues of linear transformations, I. In *Proceedings of the National Academy of Sciences, USA*, volume 35, pages 652–655, 1949.

[16] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181, January 1982.

[17] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[18] Miroslav Fiedler. Eigenvectors of acyclic matrices. *Czechoslovak Mathematical Journal*, 25(100):607–618, 1975.

[19] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975.

[20] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[21] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, February 1976.

[22] Ananth Y. Grama and Vipin Kumar. Scalability analysis of partitioning strategies for finite element graphs: A summary of results. In *Supercomputing '92 Proceedings*, pages 83–92, Minneapolis, Minnesota, November 1992.

[23] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. Corrigendum in *Combinatorica*, 4(4):291–295, 1984.

[24] S. Lennart Johnsson. Scalable scientific software libraries: Algorithms and software technologies. MIT Supercomputing Technologies Seminar, January 20, 1993.

[25] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, February 1970.

[26] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*. Morgan Kaufmann Publishers, San Mateo, California, 1992.

[27] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, August 1980.

[28] Christos H. Papadimitriou and Martha Sideri. The bisection width of grid graphs. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 405–410, San Francisco, California, January 1990.

[29] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, July 1990.

[30] H. D. Simon. Partitioning of unstructured problems for parallel processing. In *Symposium on Parallel Methods on Large-Scale Structural Analysis and Physics Applications*, pages 135–148, NASA Langley Research Center, Hampton, Virginia, February 1991. Published as a special issue of *Computing Systems in Engineering*, 2(2/3).

[31] Gilbert Strang. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, Publishers, San Diego, 1988.