Mathematical Programming 66 (1994) 211–239

A computational study of graph partitioning

Julie Falkner[a], Franz Rendl[b], Henry Wolkowicz[c,*]

[a]*Massey University, Department of Mathematics, Private Bag 11222, Palmerston North, New Zealand*
[b]*Technische Universität Graz, Institut für Mathematik, Kopernikusgasse 24, A-8010 Graz, Austria*
[c]*University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario, N2L 3G1, Canada*

Received 28 August 1992; revised manuscript received 2 August 1993

Abstract

Let $G=(N, E)$ be an edge-weighted undirected graph. The graph partitioning problem is the problem of partitioning the node set N into k disjoint subsets of specified sizes so as to minimize the total weight of the edges connecting nodes in distinct subsets of the partition. We present a numerical study on the use of eigenvalue-based techniques to find upper and lower bounds for this problem. Results for bisecting graphs with up to several thousand nodes are given, and for small graphs some trisection results are presented. We show that the techniques are very robust and consistently produce upper and lower bounds having a relative gap of typically a few percentage points.

Keywords: Graph bisection; Graph partitioning; Eigenvalue bounds; Quadratic 0, 1 programming; Computational tests

1. Introduction

Let $G=(N, E)$ be an edge-weighted undirected graph with node set $N=\{1, \ldots, n\}$, edge set E and weights w_{ij}, $ij \in E$. We consider the problem of partitioning the node set N into k disjoint subsets $S_1, \ldots, S_k$ of specified sizes $m_1 \geqslant m_2 \geqslant \cdots \geqslant m_k$, $\sum_{j=1}^{k} m_j = n$, so as to minimize the total weight of the edges connecting nodes in distinct subsets of the partition. This problem is well known to be NP-hard and therefore finding an optimal solution is likely to be a difficult task. Yet partitioning problems are important in the context of layout problems and VLSI design. See e.g. [18] for a recent survey of Integrated Circuit Layout. Several researchers have developed methods for finding 'good' partitions, and one of the most successful heuristics was proposed by Kernighan and Lin [17] in 1970. A recent survey by

* Corresponding author.

Johnson et al. [15] compares several heuristics for the graph bisection problem, which is the problem of partitioning the nodes into just two sets of equal size. The authors provide substantial numerical tests on sparse random (unweighted) graphs with up to 1000 nodes.

Less attention seems to have been given to estimating the quality of a partition in terms of upper and lower bounds on the optimal solution values. In the early 70s Donath and Hoffman [12] provided an eigenvalue-based upper bound on the weight of the edges not cut by any partition. They also proposed a parametric improvement strategy for their bound and provided numerical results on sparse random graphs with up to 100 nodes. Their results indicate that the gap between lower and upper bounds is fairly large as the number of nodes increases. Recently Boppana [6] has proposed a bounding technique for the special case of graph bisection. He does not give any numerical results but shows that on a certain class of random graphs, his bound is asymptotically tight. Finally, attempts have been made to use polyhedral combinatorics to solve the bisection problem. In [20] the facial structure of the equipartition problem is analyzed, but no computational results are provided.

The present paper is a sequel to [21] where several new eigenvalue-based bounds for the graph partitioning problem are presented. Whereas in [21] the focus was on the theoretical framework of the bounds, the purpose of the present paper is to study thoroughly the performance of these bounds on various classes of graphs. We present computational results on a variety of randomly generated graphs having various characteristics and also consider problems coming from real-world applications. Our results show that the eigenvalue approach is a robust and powerful tool for producing reasonably tight intervals for the optimal value of graph partitioning problems.

The paper is organized as follows. Section 2 contains basic notation. In Section 3 we discuss the theoretical background, showing how eigenvalue information can be used to obtain upper and lower bounds. Section 4 addresses computational considerations, and in Section 5 the numerical results are presented. The paper concludes with a discussion and summary in Section 6.

2. Basic notation and problem statement

Let $G=(N, E)$ denote an undirected graph with edge weights w. We denote by $A=(a_{ij})$ the weighted adjacency matrix of G, i.e.

$$a_{ij}=\begin{cases} w_{ij} & ij \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Since G is undirected, A is symmetric. The $j-$ largest eigenvalue of a symmetric matrix M will be denoted by $\lambda_j(M)$. The operator *diag*$(\cdot)$ is used in two ways. If v is a vector, diag(v) is the diagonal matrix formed from v. If M is a (square) matrix, diag(M) is the vector containing the main diagonal of M. The trace of M is denoted by tr(M). The column vector consisting of all ones is denoted by u (or u_l to indicate its size). The (column) vector of row sums of a matrix M is denoted by $r(M)$, thus $r(M)=Mu$. Similarly, $s(M):=u^tMu$

denotes the sum of all elements of M. We will also make extensive use of the following $l\times(l-1)$-matrix V_l, representing $u^{\perp}$:

$$V_l=\begin{bmatrix} y & \cdots & y \\ 1+x & & x \\ \vdots & \ddots & \vdots \\ x & \cdots & 1+x \end{bmatrix}, \tag{2.1}$$

where $x=-1/(l+\sqrt{l})$, $y=-1/\sqrt{l}$ and $l\geqslant 2$ It can be easily verified that $V_l^{\mathrm{t}}u_l=0$ and $V_l^{\mathrm{t}}V_l=I_{l-1}$.

An instance of a graph partitioning problem is described by a symmetric matrix A of size n and an integer vector $m=(m_1, \ldots, m_k)$ such that $m^{\mathrm{t}}u=n$, defining the specified sizes for the subsets of the partition. We assume without loss of generality that

$$m_1\geqslant\cdots\geqslant m_k\geqslant 1 \quad \text{and that} \quad k<n.$$

Finally we denote by $w(E)$ the sum of all edge weights of G, i.e. $w(E)=\frac{1}{2}s(A)$, and by $w(E_{\text{cut}})$ the total weight of the edges cut by an *optimal* partition. Moreover let $w(E_{\text{uncut}}):=w(E)-w(E_{\text{cut}})$.

The following nonlinear optimization problem solves the graph partitioning problem, see e.g. [21].

$$\text{(GP)} \qquad w(E_{\text{uncut}})=\max\ \tfrac{1}{2}\operatorname{tr}\ X^{\mathrm{t}}AX$$

such that

$$X^{\mathrm{t}}X=\operatorname{diag}(m), \tag{2.2}$$

$$Xu_k=u_n; \qquad X^{\mathrm{t}}u_n=m, \tag{2.3}$$

$$X\geqslant 0. \tag{2.4}$$

The constraints guarantee that all entries of the $n\times k$ matrix X are either 0 or 1 with precisely one nonzero entry in each row. The nonzero entries of column j of X represent the nodes contained in S_j.

3. Eigenvalue-based bounds

3.1. Upper bounds

We note that there exist very efficient heuristics to bound $w(E_{\text{uncut}})$ from below. Starting with the work of Kernighan and Lin in 1970 [17], various strategies have been proposed to generate and improve good partitions based on e.g. simulated annealing [15] or more recently genetic algorithms, see e.g. [8, 7]. On the other hand the mathematical analysis of upper bounds for graph partitioning seems to have attracted much less attention.

Our main goal in this paper is to demonstrate an approach that produces both lower and upper bounds of good quality. We begin by describing how the formulation of (GP) can be used to obtain tractable relaxations of the graph partitioning problem, which lead to tight upper bounds.

Dropping the constraints (2.3) and (2.4) leads to one of the first relaxations for graph partitioning. It was proposed by Donath and Hoffman in the 1970s [12].

$$w(E_{\text{uncut}}) \leqslant \max\{\tfrac{1}{2}\text{tr}X^{\text{t}}AX\text{: } X \text{ satisfies } (2.2)\} = \frac{1}{2}\sum_{j=1}^{k} m_j\lambda_j(A). \tag{3.1}$$

Any X containing pairwise orthogonal eigenvectors x_j corresponding to $\lambda_j(A)$ and having the correct length $\|x_j\|^2 = m_j$ constitutes a maximand in (3.1).

The Donath–Hoffman bound can be further strengthened by dropping only the nonnegativity conditions from (GP), see [21]. In the case where the m_j are all equal (to n/k), the linear term in the bound becomes trivial, i.e. a constant. From now on we will focus on this special case.

Assumption. $m_1 = \cdots = m_k = n/k.$

The following bound is derived in [21].

$$\begin{aligned} w(E_{\text{uncut}}) &\leqslant \max\{\tfrac{1}{2}\text{tr}X^{\text{t}}AX\text{: } X \text{ satisfies } (2.2), (2.3)\} \\ &= \frac{n}{2k}\sum_{j=1}^{k-1} \lambda_j(V_n^{\text{t}}AV_n) + \frac{1}{2k}s(A). \end{aligned} \tag{3.2}$$

This upper bound is attained for

$$X = \frac{1}{k}u_n u_k^{\text{t}} + \sqrt{\frac{n}{k}}\, V_n Z V_k^{\text{t}}, \tag{3.3}$$

where Z contains a set of $k-1$ orthonormal eigenvectors corresponding to the largest $\lambda_j(V_n^{\text{t}}AV_n)$. By construction, this X satisfies the orthogonality constraint (2.2) and the row and column sum constraint (2.3), but it need not be integer, since nonnegativity is dropped.

A further improvement can be achieved along the following lines [12, 21]. Let $d \in \mathbb{R}^n$ and X be an arbitrary feasible partition, i.e. X satisfies (2.2), (2.3), (2.4). Then it can readily be seen that

$$\text{tr}\, X^{\text{t}}\left(\text{diag}(d) - \frac{d^{\text{t}}u}{n} I\right)X = 0.$$

Therefore, see [21], we conclude that

$$\begin{aligned} w(E_{\text{uncut}}) &= \max\left\{ \tfrac{1}{2}\operatorname{tr} X^{\text{t}}\left(A+\operatorname{diag}(d)-\frac{d^{\text{t}}u}{n}\,I\right)X \colon X\ldots\text{feasible partition}\right\} \\ &\leqslant \max\left\{ \tfrac{1}{2}\operatorname{tr} X^{\text{t}}\left(A+\operatorname{diag}(d)-\frac{d^{\text{t}}u}{n}\,I\right)X \colon X \text{ satisfies (2.2), (2.3)}\right\} \\ &= \frac{1}{2k}\,s(A)+\frac{n}{2k}\sum_{j=1}^{k-1}\lambda_j\left(V_n^{\text{t}}\left(A+\operatorname{diag}(d)-\frac{d^{\text{t}}u}{n}\,I\right)V_n\right) \\ &:= \frac{1}{2k}\,s(A)+\frac{n}{2k}f(d) \end{aligned}$$

thereby defining $f(d)$. Since adding a multiple of the identity to a matrix shifts the eigenvalues we may also write

$$f(d)=\sum_{j=1}^{k-1}\lambda_j(V_n^{\text{t}}(A+\operatorname{diag}(d))V_n)-\frac{k-1}{n}\,d^{\text{t}}u. \tag{3.4}$$

This shows in particular that $f(d)=f(d')$, if $d'=d+\alpha u$. Therefore the restriction of f to $\{d\in\mathbb{R}^n : d^{\text{t}}u=0\}$ does not change the range of f. This observation will be used later on. For computational purposes it is preferable to have a smaller dimensional set over which to minimize. From a theoretical viewpoint it seems easier to work with an unconstrained problem.

The function f has several well-known and often used properties, see [9, 21], which we summarize below.

Proposition 3.1. *The function f is*
(a) *convex,*
(b) *bounded from below,*
(c) *continuous,*
(d) *differentiable if and only if $\lambda_k(\cdot)<\lambda_{k-1}(\cdot)$. In this case*

$$\frac{\partial}{\partial d_j}(f)=e_j^{\text{t}}V_nZZ^{\text{t}}V_n^{\text{t}}e_j-\frac{k-1}{n},$$

where Z contains an orthonormal set of eigenvectors associated with the λ_j's.

In [21] it was observed that $\inf\{f(d) : d\in\mathbb{R}^n\}$ is attained and therefore the best choice for d to produce an upper bound on $w(E_{\text{uncut}})$ is to find

$$\min\{f(d) : d\in\mathbb{R}^n\}.$$

This leads to the following bound for partitioning the nodes into subsets of equal size, see [21]:

$$w(E_{\text{uncut}})\leqslant\frac{s(A)}{2k}+\min\left\{\frac{n}{2k}f(d) : d\in\mathbb{R}^n\right\}. \tag{3.5}$$

If f is differentiable at a minimizer d, then the resulting maximand X can be shown to possess additional properties.

Theorem 3.1. *Let d^* be a minimizer of $f(d)$ and suppose f is differentiable at d^*. Let Z denote the matrix whose columns span the eigenspace associated with $f(d^*)$. Then the matrix X formed according to* (3.3) *using Z satisfies*

$$\operatorname{diag}(XX^t) = u_n.$$

Proof. Since d^* is a minimizer of f and f is differentiable at d^*, the first order optimality conditions imply

$$0 = \frac{\partial}{\partial d_j} f(d^*) = e_j^t V_n ZZ^t V_n^t e_j - \frac{k-1}{n}, \quad \forall j.$$

Now note that from (3.3)

$$V_n Z = \sqrt{\frac{k}{n}}\, XV_k, \qquad V_k V_k^t = I_k - \frac{1}{k}\, u_k u_k^t \quad \text{and} \quad Xu_k = u_n.$$

Substituting for $V_n Z$ we get

$$\frac{k-1}{n} = \frac{k}{n}\, e_j^t XX^t e_j - \frac{1}{n}.$$

Therefore $e_j^t XX^t e_j = 1$, $\forall j$, or equivalently $\operatorname{diag}(XX^t) = u$, completing the proof. $\square$

Corollary 3.1. *Under the conditions of the Theorem suppose further that $k = 2$. Then the resulting matrix X is an optimal solution to the graph bisection problem.*

Proof. In this case we have for each i: $x_{i1}^2 + x_{i2}^2 = 1$, $x_{i1} + x_{i2} = 1$, showing that the entries of X are 0 or 1. $\square$

It is not hard to see that for $k > 2$ the additional condition $\operatorname{diag}(XX^t) = u$ does not necessarily force integer entries in X. The corollary indicates that if the optimal d^* is reached and the resulting λ_1 is simple, then the underlying bisection problem is solved. Our numerous computational experiments on randomly generated graphs do not suggest that this phenomenon will occur often. In fact, on all the bisection problems that we considered we noticed that during the process of minimizing $f(d)$, the distance between the two largest eigenvalues tended to 0, therefore the corollary was never applicable. Thus the lesson to be learned from this result seems to be that one has to be prepared to face nondifferentiability when minimizing $f(d)$.

We note however that graphs having a 'dominating' optimal bisection often possess the differentiability property of Theorem 3.1, see [6]. Members of this class can be constructed as follows: We select a vertex bisection (S_1, S_2) and choose edges joining S_1 and S_2 with probability p and all other edges with probability $q>p$. It is an immediate consequence of [6] that if q is sufficiently larger than p, then the conditions of Theorem 3.1 are almost surely satisfied.

Finding a suitable descent direction in the presence of multiple eigenvalues can be extremely difficult. We describe later on in more detail how we deal with nondifferentiability from a computational point of view. Here we point out a certificate that permits us to check whether a given direction d is indeed a descent direction. For simplicity we consider only the case of bisection, $k=2$, thus

$$f(d)=\lambda_1(V_n^t(A+\operatorname{diag}(d))V_n):=\lambda_1(\hat{A}(d)). \tag{3.6}$$

Here we introduce $\hat{A}(d)$ as a shorthand for the 'projected matrix'.

Lemma 3.1. *Suppose that $\bar{d}\in\mathbb{R}^n$ is given and that the largest eigenvalue at $\bar{d}$, $f(\bar{d})$, has multiplicity t with corresponding eigenspace spanned by Q, where Q is $(n-1)\times t$ and $Q^tQ=I_t$. Then d is a descent direction for f at $\bar{d}$ if and only if the matrix*

$$Q^tV_n^t\operatorname{diag}(d)V_nQ<0, \tag{3.7}$$

i.e. is negative definite.

Proof. Suppose that d is a descent direction at $\bar{d}$. Thus there exists $\bar{\alpha}$ such that

$$f(\bar{d}+\alpha d)<f(\bar{d}), \quad \forall 0<\alpha\leqslant\bar{\alpha}. \tag{3.8}$$

Now if (3.7) fails, then there exists y with $\|y\|=1$ such that

$$y^tQ^tV_n^t\operatorname{diag}(d)V_nQy\geqslant 0.$$

Let $z=Qy$. Then $\|z\|=1$ and

$$f(\bar{d}+\alpha d)\geqslant z^t\hat{A}(\bar{d}+\alpha d)z\geqslant f(\bar{d}), \quad \forall 0\leqslant\alpha,$$

a contradiction. This proves necessity.

To prove sufficiency, let

$$B=\hat{A}(\bar{d})-f(\bar{d})I, \qquad C=V_n^t\operatorname{diag}(d)V_n.$$

Then B is negative semidefinite while C is negative definite on the null space of B, by (3.7). Therefore, (see e.g. p. 408 in [19]) $B+\alpha C$ is negative definite for sufficiently small positive α, therefore d is a descent direction. □

We have just pointed out that under certain circumstances minimizing $f(d)$ may actually lead to an optimal solution of the bisection problem. It would be perhaps more interesting to have some information on how bad the resulting bounds can be as compared to $w(E_{\text{uncut}})$. We are not aware of a general answer to this question. If we are restricting ourselves to subsets of equal size in graphs with nonnegative edge weights, we can at least show that the following choice (3.9) for d yields a bound that is not worse than the trivial bound $w(E)$, the sum of all weights.

Before proving this claim we point out the following extremal property of d from (3.9). For a symmetric matrix A, it is known that (see [28])

$$\sum_{i=1}^{k} \frac{\lambda_i(A)}{k} \leqslant m + \sqrt{\frac{n-k}{k}}\, s,$$

where $m = \operatorname{tr}(A)/n$ and $s^2 = \operatorname{tr}(A^2)/n - m^2$. In order to find a good initial diagonal shift d, a good heuristic would be to minimize the sum of the $k-1$ largest eigenvalues of the projected matrix $\hat{A}(d)$. Note that $\operatorname{tr}(V_n^{\mathrm{t}} A V_n) = \operatorname{tr} PAP$, where $P = I - uu^{\mathrm{t}}/n$. From this we see that the mean of the eigenvalues, $m(d)$, is constant with respect to d, since $d^{\mathrm{t}}u = 0$. Therefore, we need minimize only the variance $s^2(d)$, which now depends only on the first term $\operatorname{tr}(\hat{A}^2(d))/n$. Therefore we need minimize only the Frobenius norm with respect to d. The solution to this is given by the shift (3.9) below. We leave it to the interested reader to work out the details.

Theorem 3.2. *Let A be the weighted adjacency matrix of a graph with nonnegative edge weights. Define*

$$d := \frac{s(A)}{n} u - r(A). \tag{3.9}$$

Then the graph partitioning problem, with k subsets of equal size, has the following bound

$$w(E_{\text{uncut}}) \leqslant \frac{s(A)}{2k} + \frac{n}{2k} \sum_{i=1}^{k-1} \lambda_i (V_n^{\mathrm{t}}(A + \operatorname{diag}(d)) V_n) \leqslant w(E).$$

Moreover, the second inequality holds with equality if and only if the graph has (at least) k components.

Proof. First note that $d^{\mathrm{t}}u = 0$, so the first inequality follows from (3.5). Now since $\operatorname{diag}(A) = 0$ and A is nonnegative elementwise, we see that the matrix $A - \operatorname{diag}(r(A))$ is negative semidefinite. Therefore, for all y with $\|y\| = 1$,

$$y^{\mathrm{t}}(A + \operatorname{diag}(d))y = y^{\mathrm{t}}(A - \operatorname{diag}(r(A)))y + \frac{s(A)}{n} \leqslant \frac{s(A)}{n}.$$

Choosing $y_i = V_n x_i$, for the appropriate eigenvector x_i, yields

$$\lambda_i(V_n^{\mathrm{t}}(A + \operatorname{diag}(d)) V_n) \leqslant \frac{s(A)}{n}$$

and the second inequality follows. To discuss when equality holds, we add a multiple of the identity to make the resulting matrix nonnegative elementwise. Consider

$$A' := A - \operatorname{diag}(r(A)) + \alpha I,$$

where α is chosen large enough to ensure $A' \geqslant 0$, elementwise. The largest eigenvalue of A' is α. By the Perron–Frobenius Theorem for nonnegative matrices, α is simple if the directed graph associated with the nonzeros of A' is strongly connected (see e.g. [14] p. 508). Since A' is symmetric this is equivalent to the original graph being connected. Thus α is simple if G is connected, and so (u being an eigenvector for α),

$$\begin{aligned}\alpha &> \max\{x^{\mathrm{t}}A'x : x^{\mathrm{t}}x = 1, x \perp u\}\\ &= \lambda_1(V_n^{\mathrm{t}}A'V_n) = \lambda_1(V_n^{\mathrm{t}}(A+\operatorname{diag}(d))V_n) - \frac{s(A)}{n} + \alpha,\end{aligned}$$

therefore strict inequality holds. More generally, $\alpha > \lambda_{k-1}(V_n^{\mathrm{t}}A'V_n)$ if the graph has fewer than k components and so strict inequality holds again. Conversely, if the graph has at least k components, then we can choose the eigenvector consisting of the vector of ones for a fixed component and zeros elsewhere, showing that the dimension of the eigenspace corresponding to α is at least k. To conclude we observe that the eigenvalues of $V_n^{\mathrm{t}}A'V_n$ interlace those of A', therefore the $k-1$ largest eigenvalues are all equal to α. □

The bound (3.5) seems to be the best bound currently available for partitioning into sets of equal size. In our numerical tests presented later we will focus mainly on this case with $k=2$, the bisection problem. We provide substantial numerical results using (3.5) on various classes of graphs. Before doing so we discuss how the partitioning bounds relate to the Laplacian eigenvalues.

3.2. Partitioning bounds and Laplacian eigenvalues

It is interesting to note that the eigenvalues appearing in the previous theorem are closely related to the eigenvalues of the Laplacian matrix of the graph in question. Since the Laplacian spectrum of graphs is a well-studied area of spectral graph theory, see e.g. [10], we establish in this short subsection connections between our graph partitioning bounds and the Laplacian spectrum of graphs. The *Laplacian* L_A of a graph having adjacency matrix A is defined as

$$L_A := \operatorname{diag}(r(A)) - A. \tag{3.10}$$

The eigenvalues of L_A are all nonnegative and are denoted by

$$\lambda_1(L_A) \geqslant \cdots \geqslant \lambda_n(L_A) = 0,$$

following our previous notation. (We point out that the numbering of the Laplacian eigenvalues is often done in the reverse order.)

Lemma 3.2. *Let d be given by* (3.9). *For $j = 1, \ldots, n-1$:*

$$\lambda_j(V_n^{\mathrm{t}}(A+\operatorname{diag}(d))V_n) = \frac{s(A)}{n} - \lambda_{n-j}(L_A).$$

Proof. We first observe that

$$\lambda_j(V_n^t(A+\text{diag}(d))V_n)=\frac{s(A)}{n}+\lambda_j(V_n^t(-L_A)V_n).$$

Now let x be some eigenvector of $\lambda_j(L_A)$ for $j<n$. Then $x\perp u$ because u is eigenvector to $\lambda_n(L_A)=0$. Therefore $x=V_ny$ for some $y\neq 0$. This shows that

$$V_n^t(L_A)V_ny=\lambda_j(L_A)y$$

and therefore

$$\lambda_j(V_n^tL_AV_n)=\lambda_j(L_A)$$

for $j=1,\ldots,n-1$. □

A general graph partitioning problem is described by the adjacency matrix A and the vector m of specified sizes. Here the m_i need not be pairwise equal. Following [21] let

$$\bar{m}:=(\sqrt{m_1},\ldots,\sqrt{m_k})^t.$$

Let the matrix W be characterized by

$$W^tW=I_{k-1},\qquad W^t\bar{m}=0.$$

We can now express the graph partitioning bound from [21], Theorem 5.1, in terms of the smallest Laplacian eigenvalues of L_A. Let $M:=\text{diag}(m)$.

Theorem 3.3. *Let A and m describe a graph partitioning problem. Then*

$$w(E_{\text{cut}})\geqslant\frac{1}{2}\sum_{j=1}^{k-1}\lambda_{n-j}(L_A)\lambda_j(W^tMW).$$

Proof. In Theorem 5.1 of [21] it is shown that

$$\begin{aligned}w(E_{\text{cut}})&:=w(E)-w(E_{\text{uncut}})\\&\geqslant\frac{s(A)}{2}-\frac{s(A)s(M^2)}{2n^2}-\frac{1}{2}\sum_{j=1}^{k-1}\lambda_j(V_n^t(A+\text{diag}(d))V_n)\lambda_j(W^tMW).\end{aligned}$$

Now observe that

$$\text{tr}W^tMW=n-\frac{s(M^2)}{n}.$$

After substituting the expression for the eigenvalues from the previous lemma the terms containing $s(A)$ cancel, leaving the eigenvalue terms. □

In general it is rather cumbersome to express the eigenvalues of W^tMW in closed form.

In the special case $k=2$, the bound resulting from Theorem 3.3 turns out to be a well-established theorem in spectral graph theory, see [1, 16].

Corollary 3.2. *Suppose $k=2$ in Theorem 3.3. Then*

$$w(E_{\text{cut}}) \geqslant \lambda_{n-1}(L_A)\,\frac{m_1 m_2}{n}.$$

Proof. An independent proof is contained in e.g. [1] or in [16]. Here we simply note that, since $k=2$, we have

$$\lambda_1(W^{\mathrm{t}}MW) = \operatorname{tr}(W^{\mathrm{t}}MW) = \frac{2m_1 m_2}{n}. \qquad \square$$

In the case $k=3$, it is still possible to obtain closed-form representations of the eigenvalues of $W^{\mathrm{t}}MW$.

Corollary 3.3. *Suppose $k=3$ in Theorem 3.3. Let*

$$\mu_{1,2} := \frac{1}{n}\left(m_1m_2 + m_1m_3 + m_2m_3 \pm \sqrt{m_1^2m_2^2 + m_1^2m_3^2 + m_2^2m_3^2 - nm_1m_2m_3}\right).$$

Then

$$w(E_{\text{cut}}) \geqslant \tfrac{1}{2}\lambda_{n-1}(L_A)\,\mu_1 + \tfrac{1}{2}\lambda_{n-2}(L_A)\,\mu_2.$$

Proof. Let $\mu_i = \lambda_i(W^{\mathrm{t}}MW)$. We observe that μ_1 and μ_2, $(\mu_1 \geqslant \mu_2)$, are characterized by

$$\mu_1 + \mu_2 = \operatorname{tr}(W^{\mathrm{t}}MW) = n - \frac{s(M^2)}{n},$$

$$\mu_1^2 + \mu_2^2 = \operatorname{tr}(W^{\mathrm{t}}MWW^{\mathrm{t}}MW) = s(M^2) - \frac{2}{n}\,s(M^3) + \frac{s(M^2)^2}{n^2}.$$

After a somewhat tedious calculation, it can be verified that μ_1 and μ_2 as defined above satisfy these two equations. $\square$

Further simplifications occur if some of the μ_i are equal, but we do not pursue this any further.

3.3. Lower bounds

The upper bounding techniques find approximate solution matrices X which in general are not feasible because they are not integer. However, in [21] it was observed that X can also be used to obtain lower bounds on $w(E_{\text{uncut}})$, see also [4, 5].

$$\max\{\operatorname{tr} X^{t}Y : Y \text{ satisfies } (2.3),\ (2.4)\} \qquad (3.11)$$

produces a partition Y that is closest to X in Frobenius norm. Alternatively, the problem

$$\max\{\operatorname{tr} X^{t}AY : Y \text{ satisfies } (2.3)(2.4)\} \qquad (3.12)$$

amounts to a linearization of the graph partitioning cost function at X. The optimal Y from this problem is a best partition corresponding to this linearized model. Both these problems can be solved efficiently as transportation problems. Note that for linear objective functions the constraint (2.2) can be dropped because the set characterized by (2.3), (2.4) has only integer extreme points.

The approximate solution matrices X from (3.3) are in general not unique because the eigenvectors chosen in (3.3) are not unique in the presence of multiple eigenvalues.

We will take a closer look at the transportation problems in the case of the bisection problem. We use

$$V_2^{t} = \frac{1}{\sqrt{2}}(-1\ 1), \qquad Y = (y_1, u_n - y_1),$$

where y_1 is a zero-one column having $\frac{1}{2}n$ entries equal to one. Thus

$$\operatorname{tr} X^{t}AY = \tfrac{1}{2}u_n^{t}Au_n + \tfrac{1}{2}\sqrt{n}(AV_nZ)^{t}u_n - \sqrt{n}(AV_nZ)^{t}y_1 .$$

Note that the first two summands on the right hand side are constant and therefore maximizing $\operatorname{tr} X^{t}AY$ with respect to Y is equivalent to minimizing

$$(AV_nZ)^{t}y_1, \quad \text{where} \quad y_1 \in \{0, 1\}^{n}, \quad y_1^{t}u_n = \tfrac{1}{2}n.$$

It is easy to see that the optimal solution to this problem is obtained simply by setting the elements of y_1 corresponding to the $\frac{1}{2}n$ largest elements of AV_nZ to zero and setting the remaining terms to one.

Replacing A by the identity matrix I, we immediately see that the optimal solution to (3.11) is given in a similar way by setting y_1 using V_nZ. Therefore both problems simply require finding the median of a real vector of n components to get a bisection. We will see that the partitions obtained in this way compare very well with solutions obtained from other much more involved heuristics. If $k>2$ then one has to solve a transportation problem.

4. Computational considerations

In this section we discuss implementation details for the bound (3.5) for the bisection problem. Since we need to minimize a convex, but possibly nonsmooth function, it seems natural to apply an iterative procedure. Several software packages designed for this type of problem exist, and we have chosen to use the Bundle Trust (BT) method. In each iteration we have to calculate

$$\lambda_1(V_n^{t}(A + \operatorname{diag}(d))V_n).$$

We use a block Lanczos Algorithm for this purpose. Finally, once the upper bound is computed we have eigenvector information which can be used to generate good partitions.

4.1. Use of Lanczos algorithm

To find the largest eigenvalue of $V_n^t(A+\mathrm{diag}(d))V_n$ using a Lanczos routine from [24], we must frequently perform the matrix-vector multiplication

$$(V_n^t(A+\mathrm{diag}(d))V_n)p, \quad p\in\mathbb{R}^n.$$

Therefore this step must be implemented carefully. We will not multiply out the triple matrix product in this term, since this amounts to $O(n^3)$ double precision operations, which are too costly for problems with several hundred nodes. Also the possible sparsity of A will most likely be lost after the multiplication. Using the special form of V_n we are able to carry out the multiplication in $O(|E|+n)$ double precision operations, which is asymptotically optimal. ($|E|$ denotes the number of edges of the graph.)

Lemma 4.1. *$(V_n^t(A+\mathrm{diag}(d))V_n)p$ can be calculated in $O(|E|+n)$ arithmetic operations.*

Proof. We multiply from right to left and use the representation of V_n from (2.1). So we first form

$$w:=V_np \quad \text{by} \quad s:=\sum_{i=1}^{n-1} p_i \quad \text{and} \quad w_1=ys, \quad w_{i+1}=xs+p_i, \quad i=1,\dots,n-1$$

in $2n-2$ additions and 2 multiplications. Then we form

$$v:=(A+\mathrm{diag}(d))(V_np)=(A+\mathrm{diag}(d))w$$

in $O(|E|+n)$ operations using sparse matrix multiplication. We get the final result $z:=V_n^t(A+\mathrm{diag}(d))V_np=V_n^tv$ by

$$t:=\sum_{i=2}^{n} v_i, \qquad z_i=yv_1+xt+v_{i+1}, \quad i=1,\dots,n-1$$

in another $2n-1$ additions and 2 multiplications, completing the proof. (For the definition of the numbers x and y see (2.1).) □

The block Lanczos algorithm does not work with individual vectors but instead with orthonormal blocks of vectors. The block size should be at least as large as the multiplicity of the largest eigenvalue. We have already observed that in the course of the minimization the eigenvalues tend to cluster. This has to be taken into account when setting up the parameters for the Lanczos code. In our calculations we have set the maximum allowed blocksize to seven (somewhat arbitrarily). The number of decimal digits of accuracy required in the eigenvalues was set to five. Finally we limited the maximum number of matrix multiplications to compute the eigenvalues for a given d to $\frac{1}{2}n$. Whenever the routine failed to find the largest eigenvalue with the prescribed accuracy, we increased the blocksize by one and called the routine again. If the maximum allowed blocksize (of 7) was reached, we stopped the block Lanczos procedure, and stopped trying to find further improvements of the upper bound.

4.2. Use of BT method

As in [21], the BT-method proposed in [22] and [23] is used to carry out the minimization. This method is iterative and uses information from the current and possibly previous iterations to select a new trial point d.

We have to provide an initial choice for d to start the BT-iterations. Our experiments showed that for graphs which are not too sparse, the simple choice $d = 0$ provides an acceptable starting point. For extremely sparse graphs we found that the choice $d = 0$ often gives an upper bound much higher than the sum of all the edge weights. In this case initializing d according to Theorem 3.2 seems to make more sense. We observed however that this choice of d can lead to numerical problems if the graph consists of several components, because in this case the largest eigenvalue has multiplicity larger than one immediately. To overcome these difficulties we use a scaling factor between 0 and 1 and the value of d given by Theorem 3.2 is multiplied by this scaling factor. The following is a parameter setup for the BT-Code which led to a satisfactory performance of the eigenvalue optimization:

- RESET = 10 (maximum bundle size)
- EPS = 0.1 (stopping condition for norm of ϵ subgradients)
- MAXIT = 30 (maximum number of iterations allowed)
- MAXCOM = MAXIT + 10 (maximum number of function evaluations allowed)
- F (initial guess on best upper bound). Here we use a convex combination of the value of the initial upper and lower bound. A good choice turned out to be the mean of the two values, or a combination where the upper bound was slightly favoured.

4.3. Finding a feasible solution

At the end of the sequence of BT-iterations, the algebraic multiplicity of the largest eigenvalue is normally greater than one. Thus we have an infinite variety of eigenvectors available for use in the generation of lower bounds, as described in Section 3.

The eigenvectors found by the Lanczos algorithm form an orthonormal basis of the eigenspace. We approximate the search of the entire eigenspace by considering pairs of eigenvectors from the basis. For each pair z_1 and w_1 say, we first calculate $z = AV_nz_1$ and $w = AV_nw_1$, see also the section on lower bounds. We then consider a unit vector spanned by z and w:

$$v(\gamma) = (\cos\gamma)z + (\sin\gamma)w. \tag{4.1}$$

The values of γ which are of interest are those where the resulting partition (obtained by sorting) changes. Heuristically, we can assume that about half of the components are positive and about half are negative, and then consider the values of γ where the trajectory crosses the x-axis, i.e.

$$v_i(\gamma_i) = 0 \;\Rightarrow\; \gamma_i = \arctan\left(-\frac{z_i}{w_i}\right) \quad \text{for } i = 1, \ldots, n. \tag{4.2}$$

For each γ_i, we obtain a feasible solution by sorting the corresponding linear combination. If the cost of this new partition is better than that of the best partition previously found, the new solution is stored. As n gets larger, it may become too time consuming to look at all n partitions, generated by rounding $v(\gamma_i)$ for all i. We have chosen to move in steps of 5 for larger problems ($n \geqslant 1000$), i.e. we consider only the partitions generated by $v(\gamma_{5i})$, $i = 1, \ldots, \frac{1}{5}n$.

When all values of γ_i have been considered, we apply the Kernighan–Lin [17] heuristic to the best solution found. Given any feasible bisection, the Kernighan–Lin heuristic attempts to improve it by performing a series of interchanges between the two sets of the partition. The algorithm begins by choosing the two nodes which, if interchanged, will give the maximum gain g_1 say. These two nodes are then removed from consideration and the maximum additional gain achieved by interchanging two further nodes is found. This process is continued until there are no more nodes to consider. The algorithm then chooses k to maximize the partial sum $G = \sum_{i=1}^{k} g_i$. If $G > 0$ then an improvement in cost of value G can be achieved by interchanging the corresponding $2k$ nodes. The resulting partition is then treated as an initial partition, and the whole procedure is repeated until $G = 0$.

In our experiments, the initial partitions are already of reasonably good quality, so intuitively it is unlikely that a swap of a very large number of nodes will lead to an improvement. Therefore we have modified the Kernighan–Lin heuristic to explore only swap sets of size at most 30. This has led to a considerable speedup of the heuristic, with virtually no loss in the quality of the partitions found.

In summary we perform the following steps to find a feasible solution.

Lower bound routine:
Let two unit vectors z_1 and w_1 be given, orthogonal to each other.

Step 1: Go around the circle formed from z_1 and w_1 as described above and consider the bisections given by (4.1) and (4.2).

Step 2: Apply the Kernighan–Lin heuristic modified to a limited swap depth to the best bisection found in Step 1 and output this bisection.

5. Numerical results

5.1. Introduction

In this section we present computational experience, concentrating in particular on the relative gap between the upper and lower bounds. This gives an estimate of the distance of the feasible solution from optimality. We define the gap as

$$\text{gap} := \frac{\text{upper bound} - \text{lower bound}}{\text{lower bound}}.$$

The numerical results described below are structured as follows. First we describe our approach in full detail on the graph from Fig. 5.1. Then we present a series of results on randomly generated graphs $G \in \mathcal{G}_{n,p}$. This is the standard graph model where graphs on n nodes are generated, having edge probability p. We consider both weighted and unweighted problems. We also use test graphs from this class, studied previously for partitioning heuristics, see [15].

Next we investigate graphs having some underlying geometric structure. These graphs come closer to real-world applications of graph partitioning in circuit design. We also show that our approach works well on graphs consisting of several components. To explore the potential of our method we also look at larger graphs having several thousand nodes. We

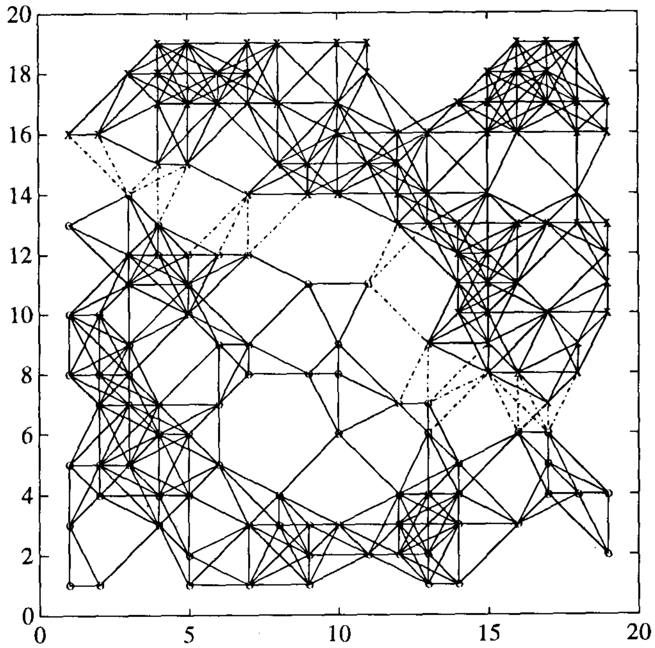

Fig. 5.1. Geometric graph with bisection.

also present some results on trisecting random graphs. Finally we present results on data from real-world problems. An informal description of our method is as follows.

Phase 1: (Preprocessing).
Use an initial choice for d to produce a starting upper bound and a lower bound on the optimal bisection. The upper bound is given by (3.6). To find a 'good' starting bisection we calculate initially the two largest eigenvalues and use the corresponding eigenvectors for the lower bound routine described in Section 4.3.

Phase 2: (Optimizing the upper bound).
Use the BT-Code to improve the upper bound.

Phase 3: (Optimizing the lower bound).
After exiting from the BT-code, we calculate the 3 largest eigenvalues and corresponding eigenvectors. These eigenvalues differ in general only marginally. We use each of the three combinations of pairs of eigenvectors to call the lower bounding routine, and output the best partition found.

5.2. A detailed example

The geometric graph from Fig. 5.1 will be used to feature in full detail various aspects of our approach to obtaining lower and upper bounds on the bisection problem. The nodes of the graph correspond to grid points on a 20×20 grid. There are $n = 150$ nodes altogether,

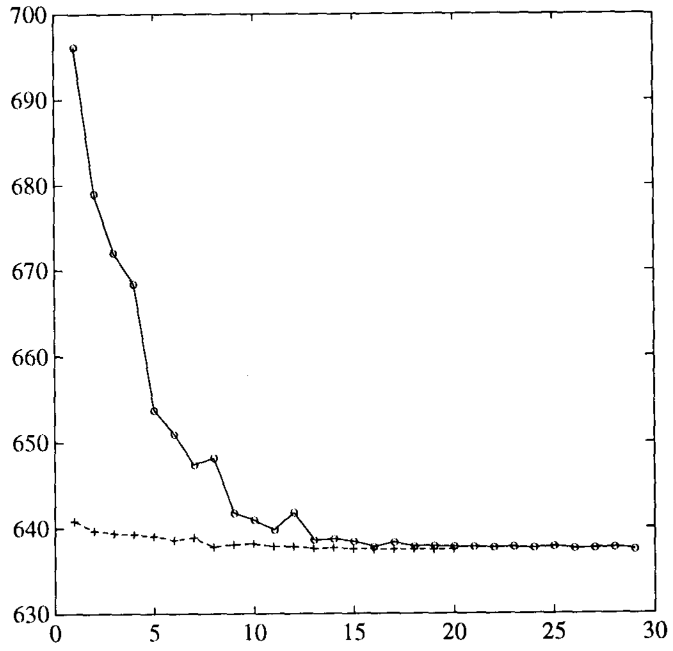

Fig. 5.2. Improvement of upper bound.

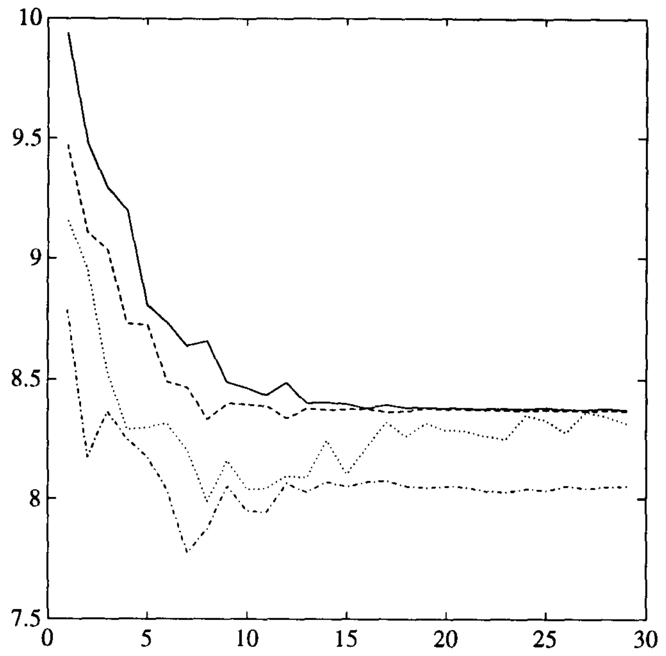

Fig. 5.3. Largest eigenvalues during iterations.

and two nodes i and j are joined by an edge of weight $w_{ij} = 1$ whenever their distance is at most $\sqrt{8}$. This gives a total of $w(E) = 647$ edges. Fig. 5.1 also shows a bisection where only 27 edges are cut, leaving 620 edges uncut, i.e.

$$w(E_{\text{uncut}}) \geqslant 620.$$

The nodes of the partitions S_1 and S_2 are indicated by 'o' and 'x'. In addition, the edges cut by this particular bisection are drawn as dotted lines.

In order to estimate the quality of this bisection we calculate the upper bound on $w(E_{\text{uncut}})$, given in Section 3.1. This bound is determined iteratively (Phase 2 above). In Fig. 5.2 we show how this upper bound improves during the iterations. We use two initial choices for the diagonal perturbation d. In variant 1 we set $d = 0$ initially, whereas in variant 2 we use the choice of d from Theorem 3.2, guaranteeing that the initial upper bound does not exceed $w(E)$. We note that in both cases the bound decreases quickly to a value of

$$w(E_{\text{uncut}}) \approx 637.4,$$

which can be considered a good approximation to the true minimum, since in both variants we stop with a subgradient of norm less than 0.1. It is also interesting to see that the bad starting choice of d in variant 1 is improved in only a few iterations to a value below the starting bound of variant 2. Variant 2 however uses only 20 iterations to reach the convergence criterion of the BT code, (norm of subgradient $\leqslant 0.1$) while variant 1 takes 30 iterations. Fig. 5.2 also indicates that most of the improvement is obtained in the first few

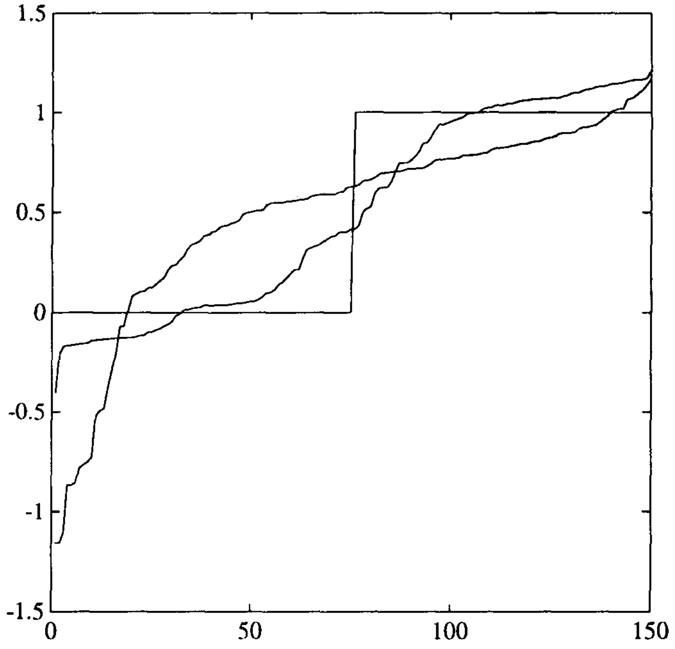

Fig. 5.4. Sorted eigenvector components.

iterations. In Fig. 5.3 we exhibit the largest eigenvalues during the iterations for variant 1. We note that after about 15 iterations the two largest eigenvalues stay very close to each other, while it takes another 15 iterations to have the third largest eigenvalue come close enough to satisfy the stopping criterion. Fig. 5.3 demonstrates in more detail that after an initial improvement phase, most of the iterations are needed to reach the preset convergence condition, while the upper bound improves only marginally. For practical purposes this means that when the upper bound stops changing significantly, there is little point in continuing to iterate. We noticed this phenomenon on virtually all the test problems we considered. We conclude this subsection by discussing how we found the bisection of Fig. 5.1 (phase 3 above). We recall Subsection 3.3 where we describe how we use the available eigenvector information to generate bisections. The main point to remember is that, given some appropriate eigenvector, a simple rounding argument leads to a bisection. Of course this rounding process is motivated by the hope that our relaxation is not too far from a 0–1 solution. To put it differently, the rounding process should lead to reasonably good bisections, provided most of the components of the vector to be rounded are either close to 0 or close to 1. In Fig. 5.4 we display the *sorted* eigenvector components from the beginning and the end of the iterations for variant 1. Ideally half of the components should be 0 and the other half be 1. It is interesting to see that, at the end of the optimization process, the number of components either close to 0 or close to 1 has gone up dramatically, as compared to the initial distribution of the eigenvector components. Rounding the final eigenvector has led (without local improvement) to the bisection indicated in Fig. 5.1. Even though we are unable to prove that this bisection is optimal, we know that

$$620 \leqslant w(E_{\text{uncut}}) \leqslant 637.$$

Therefore the error is not more than about 2.7%, or 17 edges.

All the computations described above were done on a 486-PC and took a total of no more than a few minutes of computation time.

5.3. Random graphs with uniform edge probabilities

Firstly 27 random graphs with edge weights in the range 1 to 10 were generated. Each edge was generated independent of other edges according to the edge probability p controlling the density of the graph. The graphs had between 50 and 500 nodes and densities ranging from 10% to 100%. Table 5.1 contains the results. The mean percentage difference between lower and upper bounds was 3.8, with standard deviation 1.45.

Secondly, 22 further random graphs were generated; these were as above except that the edge weights were now in the range 1 to 100. Similar results were obtained for these problems: the mean percentage difference was 3.6, with standard deviation 1.42. Thirdly, 19 unweighted random graphs were generated. These also had between 50 and 500 nodes, and the densities ranged from 10% to 75%. The mean percentage difference was 3.8, with standard deviation 1.69.

These results are encouraging. The techniques outlined in this paper can obtain, for random graphs, feasible solutions which are within a few percentage points of optimality.

It can be seen however that generally the sparser the graph, the greater the percentage gap. This is further illustrated by considering the performance of the algorithm on very sparse graphs. We solved fourteen problems with a density of 2% and between 200 and 1500 nodes. Here the mean percentage gap was 7.6 with standard deviation 0.80. A group of fourteen problems with a density of 0.5% and between 200 and 1500 nodes was also solved. This time the mean percentage gap was 5.8 with standard deviation 2.18. There was a noticeable increase in the gap with problem size, which explains the larger standard deviation observed in this case.

Experiments were also performed with random graphs provided by David Johnson; Johnson et al. [15] use these graphs to test their simulated annealing algorithm, which provides good partitions but not upper bounds. The results of our experiments are presented in Table 5.2, together with the cost of the best partitions provided in [15].

For these graphs the mean percentage gap was 6.0 with standard deviation 1.40. These results are consistent with the experience reported above: the larger gaps occur because these graphs are much more sparse than those in Table 5.1.

It can be seen that, with one exception, the Johnson partitions are at least as good as the best partitions obtained by our method: on average they are 0.6% better. It is however important to note that Johnson et al. report the best solutions ever found after performing, for each graph, 20 runs of simulated annealing, 2000 runs of Kernighan–Lin, and 2000 runs

Table 5.1
Weighted random graphs (weights in range 0 to 10)

n	$\|E\|$	Density (%)	Upper bd	Lower bd	% gap
50	136	10	542	516	5.0
50	301	25	1057	1000	5.7
50	630	50	1902	1857	2.4
50	925	75	2486	2418	2.8
50	1225	100	3265	3219	1.4
100	511	10	1881	1792	5.0
100	1249	25	3741	3582	4.4
100	2465	50	6882	6659	3.4
100	3691	75	9716	9481	2.5
100	4950	100	12503	12260	2.0
150	1105	10	3703	3517	5.3
150	2775	25	8215	7930	3.6
150	5603	50	15130	14754	2.6
150	8426	75	21698	21238	2.2
150	11175	100	27784	27389	1.4
200	2015	10	6626	6229	6.4
200	4975	25	14327	13692	4.6
200	9942	50	26455	25698	3.0
300	4489	10	14049	13257	6.0
300	11126	25	31807	30274	5.1
300	22672	50	58876	57306	2.7
400	7880	10	24124	22839	5.6
400	20111	25	54859	52975	3.6
400	39916	50	103661	100522	3.1
500	12465	10	37214	35383	5.2
500	31107	25	83884	81127	3.4
500	62338	50	160126	155422	3.0

of a local optimization algorithm. In contrast, we performed just one run of our algorithm on each graph, and each run required only a few calls to the Kernighan–Lin algorithm.

5.4. Geometric graphs

Thirty geometric graphs were considered. All of these graphs were either connected, or had at most one isolated node. These graphs were generated as follows, see also Fig. 5.1. We use a square grid of given size and select each gridpoint with a predefined probability to represent a vertex of the graph. Then we introduce edges (of weight 1) between selected gridpoints whenever their (Euclidean) distance is below a predefined threshold value.

The results are presented in Table 5.3. We present the results for only 20 of the 30 graphs generated, the results for the missing 10 graphs being similar in quality to the results presented. The mean percentage difference was only 1.3, with standard deviation 0.44. It is

Table 5.2
Johnson et al. random graphs

n	$\|E\|$	Density (%)	Upper bd	Lower bd	% gap	Johnson
124	149	2	141	136	3.7	136
124	318	4	271	254	6.7	255
124	620	8	467	442	5.6	442
124	1271	17	853	822	3.8	822
250	331	1	316	301	5.0	302
250	612	2	531	495	7.3	498
250	1283	4	981	925	6.1	926
250	2421	8	1675	1588	5.5	1593
500	625	0.5	600	573	4.7	573
500	1223	1	1071	1001	7.0	1004
500	2355	2	1844	1713	7.6	1727
500	5120	4	3564	3358	6.1	3376
1000	1272	0.25	1228	1172	4.8	1170
1000	2496	0.5	2193	2030	8.0	2045
1000	5064	1	3958	3676	7.7	3697
1000	10107	2	7109	6700	6.1	6718

Table 5.3
Geometric graphs

n	$\|E\|$	Upper bd	Lower bd	% gap
182	493	489	483	1.2
228	825	816	805	1.4
354	1394	1383	1360	1.7
408	1109	1105	1094	1.0
502	1519	1515	1501	0.9
518	5171	5094	4957	2.8
694	3340	3327	3275	1.6
760	3631	3617	3561	1.6
788	3066	3060	3028	1.1
798	2388	2385	2357	1.2
948	6278	6249	6154	1.5
960	3611	3605	3576	0.8
1088	5688	5673	5600	1.3
1110	5655	5643	5577	1.2
1362	5598	5595	5541	1.0
1378	9482	9455	9335	1.3
1426	10134	10103	9958	1.5
1466	14039	13982	13741	1.8
1878	9868	9859	9789	0.7
1936	10600	10587	10480	1.0

very encouraging to see that such good feasible solutions can be obtained for graphs which resemble real-world graphs.

We also considered the eight geometric graphs from Johnson [15]. They again performed 20 runs of annealing and thousands of runs of Kernighan–Lin and local optimization, and also found it necessary to develop a special hybrid algorithm to handle graphs with several components. Our results on these problems are presented in Table 5.4. Due to the sparsity of these graphs, we used a scaling factor for d of 0.98 in these experiments. It can be seen that the percentage gaps are small and that, with one exception, the feasible solutions obtained are as good as or better than the Johnson partitions. We also observe that the bisections which we found by the eigenvalue approach were obtained by [8]. They found their best solutions after 2000 runs of their genetic algorithm, while we look at no more than about n different cuts, and use at most 4 calls to the Kernighan–Lin improvement routine.

The graphs in Table 5.3 were all connected or almost-connected. We also experimented with a few disconnected geometric graphs. These graphs had up to 38 components. The results are presented in Table 5.5. It is not surprising that few if any edges are cut in these bisections. It is encouraging to see that our algorithm can find both good upper bounds and good feasible solutions for these problems. A scaling factor of 0.99 for d was used here and in one case (680 nodes) this led to a final upper bound which was greater than the number

Table 5.4
Johnson et al. geometric graphs

n	$\|E\|$	Upper bd	Lower bd	% gap	Johnson
500	1282	1281	1280	0.1	1278
500	2355	2347	2329	0.8	2329
500	4549	4493	4370	2.8	4371
500	8793	8629	8381	3.0	8381
1000	2394	2394	2393	0.0	2391
1000	4696	4690	4657	0.7	4657
1000	9339	9296	9117	2.0	9117
1000	18015	17775	17278	2.9	17278

Table 5.5
Disconnected geometric graphs

n	$\|E\|$	Upper bound	Lower bound	% gap
220	460	459	458	0.2
244	389	389	387	0.5
328	482	482	482	0.0
500	949	949	946	0.3
680	1247	1249	1245	0.3
724	2580	2575	2555	0.8

of edges in the graph. A scaling factor of 1.0 would have been necessary to avoid this trivial result.

5.5. *Solving larger problems*

The performance of the algorithm on some larger random graphs was also studied. In order to solve these problems in a reasonable time, a few changes were made. The initial call to the Kernighan–Lin heuristic in Phase 1 was removed. Also, whereas previously the Kernighan–Lin routine was called in Phase 3 once for each pair of eigenvectors considered, now it was called only if the feasible solution for that pair was better than any previous feasible solution. For the three largest problems, the maximum number of iterations (MAXIT) was reduced from 30 to 20.

After these changes were made, the results presented in Table 5.6 were obtained. The mean percentage difference was 4.6 with standard deviation 0.50, so the speed-up measures did not have a serious effect on the quality of the lower bounds obtained.

5.6. *Data from applications*

Up to now we have only presented artificial problems, which we (or others) generated randomly. In this subsection we apply our general procedure to problems coming from engineering applications. These problems often have an underlying geometric structure which makes them similar to the geometric graphs considered above. Our primary interest consists in deriving upper bounds on the partitions for these problems. To the best of our knowledge, no one has seriously tried to get nontrivial upper bounds for these problems. As a byproduct of our approach, we also generate bisections.

The first set of data was provided to us by H. Simon [25, 26, 3] and comes from a larger set of structural design problems. A short description provided in [25] for some of the data in Table 5.7 follows:

venkat: 2d unstructured grids for airfoils.

Table 5.6
Larger random graphs

n	$\|E\|$	Density (%)	Upper bound	Lower bound	% gap
600	17780	10	51864	49066	5.7
800	32205	10	91586	87557	4.6
1000	49819	10	139034	132846	4.7
1200	71888	10	198504	190246	4.3
1400	97806	10	267312	256875	4.1
1600	127867	10	347738	332833	4.5
1800	161891	10	437957	417864	4.8
2000	200364	10	537617	516139	4.2

Table 5.7
Real-world data

Name	n	$\|E\|$	Upper bound	Lower bound	% gap
venkat2	460	1303	1297	1268	2.3
cornell1	496	1200	1196	1180	1.4
venkat2d	843	1226	1224	1210	1.2
spiral	1200	3191	3190	3182	0.3
parc	1240	3355	3352	3334	0.5
pow9	1723	2394	2392	2385	0.3
ac	2851	15093	15078	14889	1.3
kall0	3000	15950	15894	15670	1.4
kall1	4363	26570	26480	26275	0.8
barth4	6019	17473	17466	17359	0.6
kall2	6880	46750	46503	45809	1.5
barth4dual	11451	16880	16878	16821	0.3
shuttle.eddy	10429	46585	46580	46448	0.3
kall3	10556	76109	75853	74954	1.2
barth5	15606	45878	45872	45731	0.3
chip1–2	300	1070	358.16	350.04	2.3
prim1–2	832	4707	1107.8	1092.21	1.4
prim2–2	3014	21010	3684.2	3631.99	1.4
bio–2	6417	20868	6466.71	6452.31	0.2
ind2–2	12142	73914	14891.9	14782.06	0.7

parc, *spiral*: problem used by J. Gilbert, Xerox PARC, for mesh generation, partition.

ac: connectivity matrix for HSCT (high speed civil transport), from Olaf Storaasli, NASA, Langley.

barth: grids for airfoils, dual is the dual of the Delauney triangulation, comm is the communication graph used in [27].

shuttle: connectivity for some large finite element problems, collected by Dawson Deuermeyer, CRAY Res., in [11].

The second group of data was provided to us by S. Areibi and A. Vannelli [2, 13]. This data represents hypergraph partitioning problems, where hyperedges are represented by cliques, which underestimate cuts in the original hypergraph. These problems come from netlist partitioning problems which arise from chip layout.

The results are not significantly different from our previous experiments on random geometric problems. We note that some of these graphs are extremely sparse and we manage to find bisections cutting very few edges using the eigenvector information. The relative gap between lower and upper bounds seems to be much smaller than on completely unstructured random problems.

5.7. Graph trisection

We next considered trisecting random graphs; these graphs had up to 300 nodes and weights in the range 0 to 10. Some changes to the implementation were required. It was no

Table 5.8
Trisecting random graphs

n	$\|E\|$	Density (%)	Upper bound	Lower bound	% gap
90	415	10	1325	1178	12.5
90	982	25	2592	2345	10.5
90	1985	50	4377	4010	9.2
120	734	10	2330	2061	13.1
120	1760	25	4448	4039	10.1
120	3535	50	7532	6945	8.5
150	1105	10	3262	2898	12.6
150	2775	25	6716	6053	11.0
150	5603	50	11619	10776	7.6
180	1578	10	4558	4068	12.1
180	4086	25	9709	8833	9.9
180	8121	50	16588	15465	7.3
210	2136	10	6067	5365	13.1
210	5460	25	12624	11444	10.3
210	10980	50	22087	20598	7.2
240	2897	10	8046	7202	11.7
240	7159	25	16319	14865	9.8
240	14489	50	28715	26781	7.2
270	3527	10	9466	8388	12.9
270	9004	25	20179	18327	10.1
270	18196	50	35805	33494	6.9
300	4489	10	11834	10573	11.9
300	11126	25	24589	22630	8.7
300	22672	50	43712	41034	6.5

longer possible to solve the transportation problem with a simple sort, and so instead a rounding heuristic was applied to get a feasible solution. Also, the Kernighan–Lin heuristic in its original form applies only to bisection. Thus it was repeatedly applied to pairs of subsets from the feasible partition. Table 5.8 presents some results. The mean percentage gap was 10.0%, with standard deviation 2.17%.

Even though the performance of our approach for graph trisection does not match the results for bisection we consider these results as quite encouraging. We are not aware of any comparable computational results published in the literature where both upper and lower bounds are investigated. Also we obtained our results by more or less straightforward modifications of our bisection code. It is likely that further improvement could be obtained by using more sophisticated heuristics to generate good trisections from the eigenvector information.

6. Discussion and summary

We have applied an eigenvalue approach for finding partitions to various classes of randomly generated graphs. In view of the computational experience presented in the previous section, we offer the following observations.

• The upper bound can be computed efficiently.

The bounding technique relies essentially on the ability to calculate the largest eigenvalue of the symmetric matrix $V_n^t A' V_n$. We use a representation of V_n that takes full advantage of the possible sparsity of the adjacency matrix A, without ever multiplying out the triple matrix product. In practice the computational effort to calculate the upper bound grows like

$$O(n|E|),$$

because we allow $\frac{1}{2}n$ matrix multiplications, each of $O(|E|+n)$ operations, for one eigenvalue calculation, and the number of function evaluations in the BT-Code was normally limited to (at most) 40, independent of the size of the problem. The computational overhead of the BT-Code is negligible, compared to the function evaluations. Most of the computations were done on a 486 PC (66 Mhz). On the smaller problems, the eigenvalue computations dominated the total computational effort, while on larger problems (n at least 2000), the routines to generate feasible solutions took most of the time. To give an idea of actual times, the calculation of the largest eigenvalue for a problem with $n = 10\,000$ and $|E| = 50\,000$ takes about 20 minutes (in each iteration). Graphs with $n = 1000$ and $|E| = 10\,000$ have a total running time of no more than about 10 minutes (on a PC). Since we exploit the sparsity of a problem instance, we are able to limit the space requirements of the algorithm to be linear in the number of nodes and edges of the graph.

• The upper bound leads to good partitions.

An additional advantage of our approach is that we are able to generate, at low computational cost, partitions that are often only a few percentage points from optimality. In particular it turns out that 'eigenvector rounding' often leads to bisections of high quality. Therefore we were able to apply the Kernighan–Lin heuristic with a limited swap depth without losing the potential of the general heuristic.

• The approach is robust and efficient.

We tested our bounding procedure on a variety of graphs and consistently obtained lower and upper bounds having a relative gap of less than 10%. It is particularly interesting to see the good performance on real-world problems, on graphs consisting of several connected components, and on graphs having some underlying geometric structure.

Acknowledgements

We thank Helga Schramm and Jochem Zowe for permission to use their BT-Code. We thank Horst Simon from NASA Ames Research Center and also Shawki Areibi and Tony Vannelli from University of Waterloo for giving us their data from engineering applications. The work of the first author was supported by an International Research Fellowship from NSERC of Canada. She also thanks the Christian Doppler Labor 'Diskrete Optimierung' for financial support of a visit to the University of Technology, Graz, where some of this research was carried out. The second and third authors thank the Austrian Ministry of

Science and NSERC of Canada for the continuing financial support, which has allowed them to pursue an ongoing cooperation in the study of matrix methods in combinatorial optimization.

References

[1] N. Alon and V.D. Milman, "λ_1, isoperimetric inequalities for graphs and superconcentrators," *JCT* 38 (1985) 73–88.

[2] S. Areibi and A. Vannelli, Personal communication, technical report, University of Waterloo, Waterloo, Ontario, Canada, 1993.

[3] S.T. Barnard and H.D. Simon, "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems," technical report RNR-092-033, NASA Ames Research Center, Moffett Field, CA 94035, November 1992 (to appear in *Concurrency: Practice and Experience*).

[4] E.R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM Journal on Algebraic and Discrete Mathematics* 3 (1982) 541–550.

[5] E.R. Barnes, A. Vannelli and J.Q. Walker, "A new heuristic for partitioning the nodes of a graph," *SIAM Journal on Discrete Mathematics*, 1 (1988) 299–305.

[6] R.B. Boppana, "Eigenvalues and graph bisection: An average case analysis," in: *Proceedings of the 28th Annual Symposium on Computer Science* (IEEE, London, 1987) pp. 280–285.

[7] T.N. Bui and C. Jones, "A heuristic for reducing fill-in in sparse matrix factorization," in: *Proceedings of the 6th SIAM Conference on Parallel Processing*, 1993.

[8] T.N. Bui and B.R. Moon, "Hyperplane synthesis for genetic algorithms," in: *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993.

[9] J. Cullum, W.E. Donath and P. Wolfe, "The minimization of certain nondifferentiable sums of eigenvalues of symmetric matrices," *Mathematical Programming Study* 3 (1975) 35–55.

[10] D.M. Cvetkovic, M. Doob and H. Sachs, *Spectra of Graphs – Theory and Applications* (Academic Press, New York, NY, 1979).

[11] D. Deuermeyer, "Large-scale solutions in structural analysis," *CRAY Channels* 12(1) (1990) 15–17.

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

[13] S.W. Hadley, B.L. Mark and A. Vannelli, "An efficient eigenvector approach for finding netlist partitions," *IEEE Transactions on Computer-Aided Design* 11 (1992) 885–892.

[14] R. Horn and C. Johnson, *Matrix Analysis* (Cambridge University Press, New York, 1985).

[15] D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part 1, graph partitioning," *Operations Research* 37 (1989) 865–892.

[16] M. Juvan and B. Mohar, "Optimal linear labelings and eigenvalues of graphs," *Discrete Applied Mathematics* 36 (1992) 153–168.

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

[18] T. Lengauer, *Combinatorial algorithms for integrated circuit layout* (John Wiley and Sons, Chicester, 1990).

[19] D.G. Luenberger, *Linear and Nonlinear Programming*, second edition (Addison-Wesley, Reading, Massachusetts, 1984).

[20] M.R. Rao M. Conforti and A. Sassano, "The equipartition polytope i and ii," *Mathematical Programming* 49 (1990) 49–90.

[21] F. Rendl and H. Wolkowicz, A projection technique for partitioning the nodes of a graph. Technical Report CORR 90-20, University of Waterloo, Waterloo, Canada, 1990.

[22] H. Schramm and J. Zowe, "A combination of the bundle approach and the trust region concept," in: J. Guddat et al., editor, *Advances in Mathematical Optimization* (Akademie Verlag Berlin, 1988).

[23] H. Schramm and J. Zowe, "A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results," *SIAM Journal on Optimization* 2 (1992) 121–152.

[24] D.S. Scott, Block lanczos software for symmetric eigenvalue problems, technical report 84-08, Oak Ridge National laboratory, 1979.

[25] H. Simon, Personal communication, technical report, NASA Ames Research Center, Moffett Field, CA, 1993.

[26] H.D. Simon, "Partitioning of unstructured problems for parallel processing," *Computing Systems in Engineering* 2(2/3) (1991) 135–148.

[27] V. Venkatakrishnan, H. Simon and T. Barth, "A mimd implementation of a parallel Euler solver for unstructured grids," *The Journal of Supercomputing* 6(2) (1992) 117–127.

[28] H. Wolkowicz and G.P.H. Styan, "More bounds for eigenvalues using traces," *Linear Algebra and its Applications* 31 (1980) 1–17.