Multi-level Spectral Hypergraph Partitioning with Arbitrary Vertex Sizes

Jason Y. Zien, Martine Schlag, Pak K. Chan

UCSC-CRL-96-15
October 17, 1996

Baskin Center for
Computer Engineering & Computer Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

Abstract

This paper presents a new spectral partitioning formulation which directly incorporates vertex size information. The formulation results in a generalized eigenvalue problem, and this problem is reduced to the standard eigenvalue problem. Experimental results show that incorporating vertex sizes into the eigenvalue calculation produces results that are 50% better than the standard formulation in terms of scaled ratio-cut cost, even when a Kernighan-Lin style iterative improvement algorithm taking into vertex sizes is applied as a post-processing step. The standard spectral partitioning formulation is impractical for use in multi-level partitioning schemes since it requires a contraction method that produces nearly-equal-size clusters. To evaluate the new method for use in multi-level partitioning, we combine the partitioner with a multi-level bottom-up clustering algorithm and iterative refinement. Experimental results show that our new spectral algorithm is more effective than the standard spectral formulation and other partitioners in the multi-level partitioning of hypergraphs.

1 Introduction

Previous spectral algorithms for partitioning graphs and hypergraphs have been limited by the fact that they implicitly assume that all vertices in a graph are the same size. In some problems, such as the partitioning of logic blocks for field-programmable gate arrays, this assumption may be valid, however, in problems such as macro cell partitioning or partitioning with hierarchical clustering, vertices are unlikely to all have the same size.

In this paper, we present a new spectral partitioning formulation which directly incorporates vertex sizes. This formulation yields a generalized eigenvalue problem that can be reduced to the standard eigenvalue problem. Thus, existing standard eigenvalue computation code can be used with no modifications. We apply our new method to standard benchmarks to quantitatively show the effectiveness of the new method.

Multiple levels of bottom-up clustering reduce the problem size and tend to produce superior results. Multi-level algorithms have been developed using both spectral [4], and iterative [15, 18, 17, 16], approaches. However after the application of hierarchical clustering algorithms, the resulting graph may contain vertices of different sizes. An effective spectral partitioner must take those sizes into account to produce near-optimal solutions. Our spectral partitioner, MKP (Multi-level K-way Partitioner), does exactly that.

1.1 Background

Spectral algorithms were first proposed for placement and partitioning by Hall[12]. Fast bipartitioning methods were developed based on a linear ordering of the vertices using the eigenvector associated with the second smallest eigenvalue of the Laplacian of a graph in [20, 11]. A k-way spectral partitioning algorithm and new k-way ratio-cut cost function were presented in [8]. Subsequent methods for spectral k-way ratio-cut partitioning are presented in [1, 3, 2]. Spectral quadrisection

and octasection formulations are given in [13] and [14]. Additional approaches to spectral partitioning are presented in [6, 5].

Multi-level algorithms for partitioning have become popular in recent years. A multi-level partitioning algorithm using recursive applications of the ratio-cut paradigm to form clusters is presented in [23]. The first spectral multi-level algorithm was developed by [4], however, it did not include iterative refinement at successive levels. Hendrickson and Leland implemented a multi-level algorithm with a k-way Kernighan-Lin style refinement algorithm [15]. An in-depth study of various multi-level contraction, initial partitioning, and refinement strategies on meshes concludes that all methods tested perform nearly equally as well [17].

2 Spectral partitioning with vertex sizes

In this section we describe our new spectral method. An overview of the method is as follows:

1. Establish a necessary partitioning constraint on the graph matrix representing a partition R by replacing the previous constraint $R^T R = I$ with our new constraint $R^T M R = I$ where M represents the vertex size information.
2. Formulate the relaxed version of the problem as a generalized eigenvalue problem:

$$\text{minimize } tr(X^T Q X) \text{ subject to } X^T M X = I \tag{2.1}$$

   where Q is the Laplacian of the graph.
3. Reformulate the problem, Equation (2.1) as an equivalent standard eigenvalue problem:

$$\text{minimize } tr(\hat{X}^T \hat{Q} \hat{X}) \text{ subject to } \hat{X}^T \hat{X} = I. \tag{2.2}$$

By finding the eigenvalues of the equivalent problem and scaling them appropriately, we can solve the relaxed partitioning problem with vertex sizes.

We begin by revisiting the definitions and making the modifications necessary to incorporate vertex sizes.

2.1 Definitions

Given a graph with a set of n vertices, V, we wish to find k partitions of this graph. Each vertex is associated with its size attribute. A partitioning of the graph is a division of the n vertices into k disjoint, non-empty subsets $P_1, P_2, \cdots, P_k$ such that $V = P_1 \cup P_2 \cup \cdots \cup P_k$.

- The $n \times n$ adjacency matrix, A, is composed of entries a_{ij} which represent the sum of the weights of the edges between vertices i and j.
- The $n \times n$ diagonal degree matrix, D, has entries d_{ii} equal to the sum of the weights of all edges on vertex i.
- The Laplacian matrix is defined as $Q = D - A$.
- E_h is the sum of the weights of the edges which have exactly one vertex in partition h.

- $||P_h||$ is the sum of the sizes of all vertices in partition h.
- R is the $n \times k$ ratioed assignment matrix. It represents a solution to the partitioning problem. The entry r_{ih} has value $\frac{1}{\sqrt{||P_h||}}$ when vertex i is in partition h and 0 otherwise. This definition differs slightly from [7].
- M is the $n \times n$ diagonal matrix whose m_{ii} entry is the size of vertex i.

2.2 Problem Formulation

There are many variations of the k-way partitioning problem. We will focus on optimizing the k-way ratio-cut cost function [7], that is, finding a solution R such that

$$\sum_{h=1}^{k} \frac{E_h}{||P_h||}$$

is minimized. Although it may appear to be the same cost function as presented in [7], there is a subtle difference: the definitions of R and $||P_h||$ include the actual vertex sizes rather than the number of vertices in a partition.

As in the proof of Lemma 1 found in [8]: we can show that the h^{th} diagonal entry of $R^T Q R$ satisfies:

$$(R^T Q R)_{hh} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(r_{ih} - r_{jh})^2 = \frac{E_h}{||P_h||} \tag{2.3}$$

Hence the trace of $R^T Q R$ is $\sum_{h=1}^{k} \frac{E_h}{||P_h||}$.

Vertex sizes are implicitly incorporated into our problem by our new definitions of R and $||P_h||$. We will show that by taking vertex sizes into account, the constraint $R^T R = I$ is replaced by $R^T M R = I$. Let $\mathcal{D}_{n,k}$ be the set of $n \times k$ matrices which have a single non-zero entry in every row and for each column exactly one non-zero value among its entries.

Theorem 1: *R is a ratioed partition matrix if and only if $R \in \mathcal{D}_{n,k}$ and $R^T M R = I$.*

PROOF:
Suppose R is a ratioed partition matrix. By definition each row of a ratioed partition matrix has a single non-zero entry and the non-zero entries within a column are identical. The gh^{th} element of $R^T M R$ is:

$$(R^T M R)_{gh} = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} r_{ig} m_{ij} \right) r_{jh} \tag{2.4}$$

$$= \sum_{j=1}^{n} r_{jg} m_{jj} r_{jh} \tag{2.5}$$

since $m_{ij} = 0$ for all $i \neq j$. Equation (2.5) is zero when $g \neq h$ due to the orthogonality and Boolean structure of the columns of R. In the case where g equals h Equation (2.5) is:

$$\sum_{j=1}^{n} r_{jh} m_{jj} r_{jh} = \sum_{j=1}^{n} m_{jj} r_{jh}^2 = \sum_{v_j \in P_h} m_{jj} r_{jh}^2 = \frac{\sum_{v_j \in P_h} m_{jj}}{||P_h||} = 1$$

where v_j is vertex j.

On the other hand, if R is an arbitrary matrix in $\mathcal{D}_{n,k}$ satisfying $R^T M R = I$, then we can show that R is the ratioed partition matrix for the partition $P_1, P_2, \cdots, P_k$ where P_h is the set of vertices whose rows have their non-zero entry in column h. Every vertex appears in exactly one P_h since every row has a single non-zero entry. Let r_h be the non-zero value appearing in column h. Since $R^T M R = I$ we have

$$1 = \sum_{j=1}^{n} r_{jh} m_{jj} r_{jh} = \sum_{j=1}^{n} m_{jj} r_{jh}^2 = \sum_{v_j \in P_h} m_{jj} r_h^2 = r_h^2 \sum_{v_j \in P_h} m_{jj} = r_h^2 ||P_h||$$

from which we get $r_h = \frac{1}{\sqrt{||P_h||}}$. Thus R is the ratioed partition matrix for $P_1, P_2, \cdots, P_k$. □

3 Relaxed Problem Formulation

Our objective is to find the matrix which minimizes $R^T Q R$ subject to the constraint $R^T M R = I$ and $R \in \mathcal{D}_{n,k}$. This problem is equivalent to the ratio-cut partitioning problem, and hence, there is no known optimal polynomial-time solution. We can, however, relax the problem by removing the $R \in \mathcal{D}_{n,k}$ constraint. The relaxed problem turns out to be a quadratic placement problem [12], which can be solved in polynomial time.

The relaxed problem is defined as:

$$\text{minimize } z = tr(X^T Q X) \text{ subject to the constraint } X^T M X = I. \tag{3.1}$$

In the past, it was typical to use the constraint: $X^T X = I$. Our new constraint, $X^T M X = I$, utilizes the vertex size information that is available. This constraint reduces to $X^T X = I$ when all of the vertices are unit size.

By taking advantage of the special (diagonal) form of M, we can transform this into a standard eigenvalue problem. Let $M = S^T S$.[1] Assume that all vertices have positive size and let $\hat{Q} = {S^{-1}}^T Q S^{-1}$. We consider the modified problem,

$$\text{minimize } z = tr(\hat{X}^T \hat{Q} \hat{X}) \text{ subject to the constraint } \hat{X}^T \hat{X} = I. \tag{3.2}$$

Theorem 2:

$$\min\{tr(X^T Q X) \mid X^T M X = I\} = \min\{tr(\hat{X}^T \hat{Q} \hat{X}) \mid \hat{X}^T \hat{X} = I\},$$

PROOF:

[1] S is the diagonal matrix with $\sqrt{m_{ii}}$ in the ii^{th} entry.

$$\begin{aligned}
\min\{tr(X^TQX) \mid X^TMX = I\} &= \min\{tr(X^TQX) \mid X^TS^TSX = I\} \\
&= \min\{tr(X^TQX) \mid \hat{X}^T\hat{X} = I \text{ and } SX = \hat{X}\} \\
&= \min\{tr(X^TQX) \mid \hat{X}^T\hat{X} = I \text{ and } X = S^{-1}\hat{X}\} \\
&= \min\{tr(\hat{X}^T{S^{-1}}^TQS^{-1}\hat{X}) \mid \hat{X}^T\hat{X} = I\} \\
&= \min\{tr(\hat{X}^T\hat{Q}\hat{X}) \mid \hat{X}^T\hat{X} = I\}
\end{aligned}$$

□

Moreover, there are solutions X and $\hat{X}$ for the minimum values for Equations (3.1) and (3.2) that are related by $SX = \hat{X}$.

Equation (3.2) has the same form as previous spectral partitioning formulations, except that $\hat{Q}$ is no longer the Laplacian of the graph. By using the method of Lagrange multipliers or Fan's Theorem as shown in previous literature [12, 7], Equation (3.2) leads to the standard eigenvalue problem.

$$\hat{Q}\hat{X} = \hat{X}\Lambda \tag{3.3}$$

Theorem 3: Fan's Theorem
Let the eigenvalues λ_i of a symmetric matrix Q be so arranged that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. For any positive integer $k \leq n$, the sums $\sum_{i=1}^{k} \lambda_i$ and $\sum_{i=1}^{k} \lambda_{n+1-i}$ are respectively the minimum and maximum of $\sum_{j=1}^{k} x_j^T Q x_j$ when k orthonormal vectors x_j $(1 \leq j \leq k)$ vary in the space.

PROOF: See [9].

There are many solutions for $\hat{X}$, but by applying Fan's Theorem, we find that the eigenvectors associated with the smallest k eigenvalues of $\hat{Q}$ yield an optimal solution to Equation (3.2), since when $\hat{X}$ is composed of the k eigenvectors associated with the smallest k eigenvalues of $\hat{Q}$, $\hat{X}^T\hat{Q}\hat{X} = \Lambda$. Λ is the $k \times k$ diagonal matrix composed of the the smallest k eigenvalues of $\hat{Q}$. Thus, we now have an optimal solution for the quadratic assignment problem which incorporates vertex sizes. We can obtain the answer to our original problem, Equation (3.1) using $X = S^{-1}\hat{X}$.

3.1 Lower bound on the cost function

By putting together Theorem 3 with Equation (2.3), we establish a lower bound for our cost function. Note, $\lambda_i(\hat{Q})$ denotes the i^{th} smallest eigenvalue of $\hat{Q}$. This provides a tie between the continuous solution and the feasible solutions (matrices restricted to $\mathcal{D}_{n,k}$), since continuous space solutions which have a lower cost will produce a lower bound on the optimal feasible solution cost.

Theorem 4: *Let $P_1, P_2, \ldots, P_k$ be any partition and R its ratioed partition matrix.*

$$\sum_{i=1}^{k} \lambda_i(\hat{Q}) = \min\{tr(\hat{X}^T\hat{Q}\hat{X}) \mid \hat{X}^T\hat{X} = I\} \leq tr\left(R^TQR\right) = \sum_{h=1}^{k} \frac{E_h}{||P_h||} \tag{3.4}$$

PROOF:
By applying Theorem 3 with Equation (3.2) we obtain the left hand side of Equation (3.4). The right hand side comes from Equation (2.3). □

3.2 What is being optimized?

The solution $X = S^{-1}\hat{X}$ derived from $\hat{X}$, the standard eigenvalue problem for $\hat{Q}$ in Equation (3.3), is actually the solution to a generalized eigenvalue problem,

$$QX = MX\Lambda. \tag{3.5}$$

The generalized eigenvalue problem has been well studied, and turns up in problems such as the oscillation of springs with point masses [21].

Consider our modified eigenvalue/eigenvector problem, $\hat{Q}\hat{X} = \Lambda\hat{X}$. We might speculate that the solution for $\hat{X}$ is a variation of Hall's [12] quadratic placement problem. With some calculations, we show that this is indeed the case.

Theorem 5: *If $\hat{X}$ is an $n \times k$ matrix, then $\hat{X}^T\hat{Q}\hat{X}$ is a $k \times k$ matrix whose gh^{th} component is:*

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(\frac{\hat{x}_{ig}}{\sqrt{m_i}} - \frac{\hat{x}_{jg}}{\sqrt{m_j}}\right)\left(\frac{\hat{x}_{ih}}{\sqrt{m_i}} - \frac{\hat{x}_{jh}}{\sqrt{m_j}}\right) \tag{3.6}$$

PROOF:

the gh^{th} component of $\hat{X}^T\hat{Q}(G)\hat{X}$ is

$$\begin{aligned}
&= \sum_{i=1}^{n} \hat{x}_{ig}\hat{x}_{ih}\frac{\sum_{j=1}^{n} a_{ij}}{m_i} - \sum_{i=1}^{n}\hat{x}_{ig}\sum_{j=1}^{n}\frac{a_{ij}}{\sqrt{m_i m_j}}\hat{x}_{jh} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{m_i}\hat{x}_{ig}\hat{x}_{ih} - \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sqrt{m_i m_j}}\hat{x}_{ig}\hat{x}_{jh} \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{m_i}\hat{x}_{ig}\hat{x}_{ih} - \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sqrt{m_i m_j}}\hat{x}_{ig}\hat{x}_{jh} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{m_i}\hat{x}_{ig}\hat{x}_{ih} \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{m_i}\hat{x}_{ig}\hat{x}_{ih} - \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sqrt{m_i m_j}}\hat{x}_{ig}\hat{x}_{jh} + \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}\frac{a_{ji}}{m_j}\hat{x}_{jg}\hat{x}_{jh} \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(\frac{\hat{x}_{ig}\hat{x}_{ih}}{m_i} - \frac{2\hat{x}_{ig}\hat{x}_{jh}}{\sqrt{m_i m_j}} + \frac{\hat{x}_{jg}\hat{x}_{jh}}{m_j}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(\frac{\hat{x}_{ig}}{\sqrt{m_i}} - \frac{\hat{x}_{jg}}{\sqrt{m_j}}\right)\left(\frac{\hat{x}_{ih}}{\sqrt{m_i}} - \frac{\hat{x}_{jh}}{\sqrt{m_j}}\right)
\end{aligned}$$

□

3.3 Application

The results of the modified eigenvalue problem may be used directly in *any* spectral partitioning algorithm which forms partitions from the eigenvectors of the Laplacian. The KP algorithm [8] forms k partitions by using the magnitude and orthogonality of the rows of the eigenvalue matrix. The MKP algorithm is our modified KP algorithm, which accounts for vertex size information, as described in this paper. Our partitioner, MP implements the KP and MKP algorithms, as well as multi-level contraction.

4 MP Implementation

```
MP(hgraph HG, int K, int Num_levels,
        algorithm Partition, boole Refine?)
{
    HG_0 ← HG
    for (i = 1; i ≤ Num_levels; i++)
        HG_i ← Contract(HG_{i-1})
    P_{Num_levels} ← Partition(HG_{Num_levels}, K)
    for (i = Num_levels; i > 0; i--)
    {   If (Refine? == Yes) P_i ← Improve(P_i, K)
        P_{i-1} ← Expand(P_i)
    }
    If (Refine? == Yes) P ← Improve(P_0, K)
    else P ← P_0
    return(P)
}
```

Figure 4.1: MP Partitioner.

Our partitioner, MP, has been implemented in C++. MP interfaces with the LASO library by D.S. Scott[19], which performs the sparse matrix eigenvalue/eigenvector computation. Figure 4.1 illustrates how MP integrates a k-way partitioning algorithm with contraction and iterative improvement. The k-way partitioning algorithms we implemented include

- a reimplementation of the KP partitioning algorithm [7] which uses actual vertex sizes in forming the partitions and computing the ratio-cut cost,
- the MKP partitioning algorithm, which amounts to our new KP modified to use the eigenvectors from the generalized eigenvalue problem, Equation (3.5), and
- an algorithm which generates a random k-way partition.

The last algorithm was used to evaluate the benefit of the spectral partitioning algorithm when used in conjunction with iterative improvement methods.

4.1 Contraction and Iterative Improvement

We wanted to evaluate MP's performance in three scenarios:

- on graphs where vertices were of non-unit size and with no hierarchical clustering,
- on graphs with non-unit size vertices and multiple levels of contraction,

- and finally, on graphs whose vertices were initially unit size, but became non-unit size through multiple levels of contraction.

Our focus was not to find the best contraction algorithm nor the best iterative refinement algorithm. Our main goal was to provide a framework to test our size-aware spectral algorithm. Other researchers have conducted more detailed studies of different contraction and improvement algorithms and their relative effects [17].

```
Contract(HGraph & HG) {
    numcolors=0;
    Convert HG into a graph G using Frankle clique
    expansion, ignoring nets with degree > 99
    Insert all edges of G into heap, ordered by weight
    for (i=0; i < num_vertices/2; i++) {
        Remove highest weight edge, e from heap
        v1 = e.vertex1; v2=e.vertex2;
        if (v1 and v2 are uncolored) {
            numcolors++;
            v1.color=numcolors; v2.color=numcolors;
        }
        else if (v1 is uncolored) v1.color=v2.color;
        else if (v2 is uncolored) v2.color=v1.color;
    }
}
```

Figure 4.2: Conglomerative graph contraction algorithm.

The contraction algorithm we used is shown in Figure 4.2. The edges of the hypergraph are clique expanded to obtain a graph. The algorithm orders the edges of the graph using a heap based on the weight of an edge. The edges are iteratively removed, and the vertices are merged into clusters (based on the color of a vertex). This algorithm is similar to Kruskal's minimum spanning tree algorithm, except that when $\frac{n}{2}$ edges have been encountered, the algorithm terminates.

Figure 4.3 shows the iterative improvement heuristic. Our improvement algorithm is modeled after the two-way ratio-cut algorithm [22]. We have extended it to perform k-way partitioning in the following way. In turn, we select each of the k partitions as the SINK (resp. SOURCE) partition, and the remaining partitions together form the SOURCE (resp. SINK). Vertices are moved one by one from the SOURCE to the SINK based on the gain of a vertex (the total weight of the nets that would become uncut if a vertex is moved to a partition). This process is repeated until there is no more improvement. The best k-way ratio-cut solution encountered is retained. In practice only a few passes of the outer improvement loop are performed before a local minima is reached. For our experiments, we terminated the improvement step after three passes.

4.2 Implementation Issues

MP works directly with hypergraphs, only transforming the hypergraphs into graphs for the eigenvector computation and edge matching algorithms. Hypergraphs are converted into graphs by performing a clique expansion on the hyperedges. We chose a weighting of the edges proposed by Frankle [10], in which each edge of the clique formed by hyperedge, e_i is given a weight of $(\frac{2}{deg(e_i)})^{\frac{3}{2}}$. For the eigenvalue/eigenvector computations, we chose to perform clique expansions even on very large fanout nets, although in some cases it may be more practical to set an upper threshold on the nets chosen for clique expansion so that sparse matrix

```
Improve(HGraph & HG, int k, int maxiterations) {
    do {
        save initial HGraph
        for (h=1; h < k; h++) {
            Flow(INTO,BestHG,HG, h);
            Flow(OUTOF,BestHG,HG, h);
        }
    } while (BestHG is better than HG) and
        (iterations < maxiterations);
}

Flow(int direction, HGraph & BestHG, HGraph &
        Starting, int target) {
    if (direction==INTO) {
        Build k − 1 src heaps with vertices not
        in target. Order heaps by the gain in
        moving vertex to INTO.
        While there is a non-empty heap {
            Select highest gain vertex, v, breaking
            ties using the move from largest source
            partition.
            Move v into target.
            Evaluate k-way ratio-cut cost, save if best.
        }
    }
    if (direction==OUTOF) {
        Build k − 1 dest heaps, using vertices in
        target. Order heaps by the gain in moving
        vertex from OUTOF.
        While there is a non-empty heap {
            Select highest gain vertex, v, breaking
            ties with the move from the smallest dest
            partition.
            Move v into target.
            Evaluate k-way ratio-cut cost, save if best.
        }
    }
}
```

Figure 4.3: Iterative improvement heuristic.

computations performed on the graph can be carried out efficiently. For efficiency, in the conglomerative contraction heuristic, we chose to only perform clique expansion on nets of degree smaller than 100 since these nets are unlikely to affect the clustering.

# of levels	Iterative Improvement?	Algorithm		
		RND5	KP	MKP
0	No	43.4	4.92	2.36
0	Yes	3.01	2.61	1.38
2	No	18.0	11.4	1.91
2	Yes	2.21	1.84	1.26

Table 5.1: Geometric mean of scaled cost multiplied by 10^8 over all tests with actual vertex sizes.

# of levels	Iterative Improvement?	Algorithm		
		RND5	KP	MKP
0	No	163.0	17.3	17.3
0	Yes	19.3	13.6	13.6
2	No	69.2	29.0	16.2
2	Yes	13.3	14.5	12.1

Table 5.2: Geometric mean of scaled costs multiplied by 10^5 over all tests with unit size vertices.

5 Results

We ran experiments using seven MCNC benchmarks (number of vertices is shown in parenthesis): p1_ga(833), p2_ga(3014), t2(1663), t3(1607), t4(1515), t5(2595), and t6(1752). These benchmarks are netlists with vertex sizes. The actual distribution of vertex sizes is given in Figure A.1 in the appendix.

A number of different parameters for the benchmarks were run to analyze the performance of the new method. Results under the heading MKP used the solution to the generalized eigenvalue while results under KP used the eigenvectors of the Laplacian. (With unit-size vertices, no contraction and no iterative improvement this is equivalent to the KP algorithm in [7].) In order to evaluate the benefit of using the spectral information in a multi-level partitioning scheme we also used an algorithm which generates random k-way partitions. The best results out of 5 obtained from random partitions are listed under the heading RND5.

Partitioning results were generated for $2, 4, 8$, and 16-way partitions. The results were reported using the scaled cost function [7]:

$$\frac{1}{n(k-1)} \sum_{h=1}^{k} \frac{E_h}{||P_h||}.$$

Tables 5.1 and 5.2 show the overall performance of the algorithms. We used the geometric mean of the results over all seven benchmarks for $k = 2, 4, 8$, and 16. In every test group, MKP gives the best answer. In the case where there are unit size vertices with no contraction, KP and MKP give exactly the same answer.

The detailed results are given in the following eight tables. The first four tables give the k-way ratio-cut cost after generating the partitions before any iterative improvement. We refer to these partitions as initial partitions. In Tables 5.3 and 5.4 MKP is superior to the other two partitioning methods in almost every test case. For these tests the actual circuit cell areas were used as the vertex sizes in MKP. In Tables 5.5 and 5.6 all vertices were made unit size (the actual cell areas were ignored). In Table 5.5 KP and MKP generate exactly the same answer, which is exactly what we would expect. In Table 5.6, two levels of contraction were performed, allowing vertices to become non-unit size. Here again, we see that MKP is clearly superior

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	305.08	289.87	8.62	21.76	19.39	10.92	26.72
KP	2	17.63	21.43	1.08	1.40	1.94	0.66	0.81
MKP	2	23.68	3.55	0.37	0.73	0.76	0.66	0.81
RND5	4	310.38	303.48	10.02	24.79	23.15	10.11	28.56
KP	4	19.25	34.39	1.25	2.16	2.03	0.99	1.03
MKP	4	23.14	11.28	0.46	0.81	1.38	0.59	0.79
RND5	8	340.97	338.65	12.12	27.68	31.08	12.27	30.84
KP	8	38.27	60.38	1.10	2.57	2.82	2.82	3.46
MKP	8	23.16	13.36	0.84	1.46	1.52	0.79	1.32
RND5	16	358.67	366.93	14.06	30.51	30.19	17.62	34.60
KP	16	53.78	127.90	3.54	7.50	6.02	2.80	19.33
MKP	16	40.24	23.73	11.46	2.62	2.22	0.80	1.62

Table 5.3: Scaled ratio-cut costs multiplied by 10^8 of solutions for actual vertex sizes with 0 levels of contraction without iterative improvement.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	130.99	133.37	2.17	9.10	8.44	2.92	10.24
KP	2	38.28	17.50	0.72	1.38	34.00	0.50	0.81
MKP	2	26.88	3.55	0.37	0.73	2.44	0.50	0.81
RND5	4	137.86	144.20	3.58	9.92	9.02	4.22	10.89
KP	4	29.22	22.99	0.59	2.24	12.15	0.80	79.79
MKP	4	23.73	8.94	0.50	0.66	1.86	0.32	0.88
RND5	8	156.03	164.57	4.64	12.18	11.63	5.19	13.36
KP	8	46.87	84.08	16.64	1.91	30.17	17.48	42.02
MKP	8	27.86	11.62	0.50	1.04	1.47	0.44	0.87
RND5	16	162.94	185.60	6.44	13.93	13.96	6.99	17.24
KP	16	39.80	94.46	18.83	32.08	32.39	15.29	21.53
MKP	16	23.80	16.28	0.88	1.72	1.43	0.75	1.44

Table 5.4: Scaled ratio-cut costs multiplied by 10^8 of solutions for actual vertex sizes with 2 levels of contraction without iterative improvement.

to the other methods.

Each of the last four tables show the results after the iterative improvement for the corresponding benchmarks of the first four tables. Of the 112 different benchmark tests presented in the tables, MKP produced the best answer or tied for the best answer in 109 of those tests. The weakest performance of MKP was in Table 5.10, which gives the results after iterative improvement using two levels of contraction for unit size vertices. The iterative improvement performed on two levels of contracted graphs allows the solutions of the other initial partitioning algorithms to catch up to MKP.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	332.12	87.42	159.69	156.91	172.19	96.88	130.61
KP	2	13.39	5.43	8.95	10.07	8.22	11.34	28.57
MKP	2	13.39	5.43	8.95	10.07	8.22	11.34	28.57
RND5	4	355.96	91.42	162.33	164.29	183.39	103.42	136.61
KP	4	25.62	9.40	15.45	23.57	11.13	6.40	18.12
MKP	4	25.62	9.40	15.45	23.57	11.13	6.40	18.12
RND5	8	369.24	101.17	171.34	174.67	197.43	111.87	149.18
KP	8	37.88	13.75	21.48	18.74	14.80	14.51	24.08
MKP	8	37.88	13.75	21.48	18.74	14.80	14.51	24.08
RND5	16	390.10	109.17	184.70	187.03	214.72	120.88	162.00
KP	16	58.37	21.19	36.36	41.11	22.90	18.47	28.65
MKP	16	58.37	21.19	36.36	41.11	22.90	18.47	28.65

Table 5.5: Scaled ratio-cut costs multiplied by 10^5 of solutions for unit size vertices with 0 levels of contraction without iterative improvement.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	148.85	38.92	57.03	61.69	72.71	37.88	46.72
KP	2	80.32	10.34	40.16	9.92	66.05	11.41	28.57
MKP	2	17.39	9.71	14.43	9.91	6.93	4.46	28.57
RND5	4	147.45	42.24	62.25	66.25	75.51	40.50	50.76
KP	4	68.34	13.57	28.26	17.38	31.77	8.76	31.74
MKP	4	21.91	9.79	18.36	14.86	12.85	7.25	17.69
RND5	8	163.68	48.72	72.36	78.01	86.19	46.16	62.25
KP	8	68.75	18.75	34.45	24.70	32.27	18.12	33.47
MKP	8	36.24	12.64	24.83	17.94	22.24	8.06	24.07
RND5	16	194.74	55.31	81.74	86.63	93.61	52.22	76.80
KP	16	102.10	20.50	46.97	38.20	42.36	19.00	47.97
MKP	16	56.06	16.20	26.42	22.23	23.76	9.82	32.12

Table 5.6: Scaled ratio-cut costs multiplied by 10^5 of solutions for unit size vertices with 2 levels of contraction without iterative improvement.

6 Conclusion and Future Research

In this paper we have presented a modified eigenvalue formulation to account for vertex sizes, and studied its use on circuits with varying vertex sizes and within a multi-level partitioning scheme. From the benchmarks presented in this paper, it is clear that MKP provides excellent results.

Future research will focus on adding practical constraints (input/output and partition capacity) to MKP, rather than using the k-way ratio-cut cost function.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	16.94	3.55	0.32	0.73	0.61	0.19	0.27
KP	2	17.63	3.55	0.32	1.24	0.61	0.19	0.81
MKP	2	9.53	3.55	0.32	0.73	0.39	0.19	0.81
RND5	4	28.76	30.83	0.36	0.99	1.57	0.54	0.62
KP	4	18.28	32.56	0.94	1.88	1.72	0.38	0.80
MKP	4	10.83	8.29	0.34	0.66	0.59	0.28	0.61
RND5	8	28.67	36.36	0.96	1.99	3.57	1.38	2.26
KP	8	30.86	42.51	0.73	2.11	2.34	0.71	1.71
MKP	8	18.00	11.56	0.46	1.02	0.98	0.44	0.87
RND5	16	34.07	70.03	3.61	4.66	8.88	3.22	11.99
KP	16	43.03	49.03	1.67	2.92	1.79	1.28	2.50
MKP	16	22.40	15.19	0.65	1.86	1.31	0.43	0.89

Table 5.7: Scaled ratio-cut costs multiplied by 10^8 of solutions for actual vertex sizes with 0 levels of contraction with iterative improvement.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	19.98	3.55	0.32	0.50	0.57	0.19	0.80
KP	2	19.98	3.55	0.25	1.29	0.39	0.19	0.81
MKP	2	19.98	3.55	0.32	0.73	0.57	0.19	0.81
RND5	4	20.79	8.72	0.36	1.04	1.14	0.40	1.91
KP	4	20.01	14.36	0.36	1.33	1.05	0.59	0.82
MKP	4	19.22	8.21	0.36	0.66	0.68	0.28	0.81
RND5	8	21.04	37.80	0.49	0.83	1.65	1.04	4.35
KP	8	20.69	17.66	0.38	1.02	0.86	0.70	0.81
MKP	8	19.92	9.57	0.37	0.71	0.63	0.32	0.83
RND5	16	19.62	45.47	1.00	1.75	2.30	1.99	4.46
KP	16	20.69	27.51	1.00	1.88	2.05	1.03	1.78
MKP	16	19.99	10.22	0.38	0.76	0.66	0.32	0.82

Table 5.8: Scaled ratio-cut costs multiplied by 10^8 of solutions for actual vertex sizes with 2 levels of contraction with iterative improvement.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	13.91	4.58	12.97	14.34	11.66	5.95	11.10
KP	2	13.39	5.43	8.85	8.98	7.95	8.94	11.63
MKP	2	13.39	5.43	8.85	8.98	7.95	8.94	11.63
RND5	4	22.97	15.63	19.50	16.58	12.22	6.75	13.30
KP	4	22.32	8.74	14.41	16.42	10.52	5.87	10.38
MKP	4	22.32	8.74	14.41	16.42	10.52	5.87	10.38
RND5	8	40.30	19.84	36.76	23.44	22.64	16.63	24.29
KP	8	30.78	11.48	19.05	16.80	13.08	7.84	15.12
MKP	8	30.78	11.48	19.05	16.80	13.08	7.84	15.12
RND5	16	68.18	26.32	48.07	32.01	52.90	23.04	41.59
KP	16	45.82	16.93	26.77	23.50	18.96	12.29	22.17
MKP	16	45.82	16.93	26.77	23.50	18.96	12.29	22.17

Table 5.9: Scaled ratio-cut costs multiplied by 10^5 of solutions for unit size vertices with 0 levels of contraction with iterative improvement.

partitioner	k	p1_ga	p2_ga	t2	t3	t4	t5	t6
RND5	2	24.45	5.39	10.24	11.45	5.93	4.87	9.51
KP	2	13.39	5.39	12.41	9.29	5.70	5.47	7.96
MKP	2	13.39	4.58	7.96	9.29	6.40	3.31	10.34
RND5	4	17.45	7.99	12.94	13.95	8.50	5.20	11.12
KP	4	19.86	8.11	12.06	15.15	10.44	7.69	11.84
MKP	4	17.45	9.08	11.39	11.47	9.24	5.70	11.09
RND5	8	30.16	11.56	18.56	15.78	10.49	8.50	18.68
KP	8	37.59	15.34	22.58	16.29	15.61	8.38	17.77
MKP	8	29.03	10.99	20.61	15.86	14.44	7.00	16.19
RND5	16	52.91	18.94	25.95	20.59	21.42	11.95	27.86
KP	16	65.25	14.62	31.48	31.64	23.34	12.35	42.45
MKP	16	48.39	13.53	23.69	20.58	18.73	8.96	23.41

Table 5.10: Scaled ratio-cut costs multiplied by 10^5 of solutions for unit size vertices with 2 levels of contraction with iterative improvement.

References

[1] C. J. Alpert and A. B. Kahng. Geometric embeddings for faster and better multi-way netlist partitioning. In *Proc. ACM/IEEE Design Automation Conference*, pages 743–748, 1993.

[2] C. J. Alpert and So-Zen Yao. Spectral partitioning: The more eigenvectors, the better. In *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, June 1995.

[3] Charles. J. Alpert and Andrew. B. Kahng. Multiway partitioning via geometric embeddings, orderings, and dynamic programming. *IEEE Trans. on CAD*, 14(11):1342–1358, November 1995.

[4] Stephen T. Barnard and Horst D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Technical Report RNR-92-033, NASA Ames Research Center, NAS Systems Division, Moffet Field, CA, November 1992.

[5] E. R. Barnes. Partitioning the nodes of a graph. *Proceedings of Graph Theory with Applications to Algorithms and Computer Science*, pages 57–72, 1985.

[6] E.R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algorithm and Discrete Method*, 3:541–550, December 1982.

[7] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Trans. on CAD*, 13(9):1088–1096, September 1994.

[8] Pak K. Chan, Martine D. F. Schlag, and Jason Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *Proceedings of the Symposium on Integrated Systems, University of Washington, Seattle*. The MIT Press, Cambridge, March 1993.

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

[10] Jonathan Alexander Frankle. Circuit placement methods using multiple eigenvectors and linear probe techniques. Technical Report UCB/ERL M87/32, University of California, Berkeley, Electronics Research Laboratory, Berkeley, CA, May 1987.

[11] Lars Hagen and Andrew Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *Proceedings of the ICCAD*, pages 10–12, 1991.

[12] Kenneth M. Hall. An r-dimensional quadratic placement algorithm. *Management Sciences*, 17(3):219–229, November 1970.

[13] Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460 * UC-405, Sandia National Laboratories, Albuquerque, New Mexico, 1992.

[14] Bruce Hendrickson and Robert Leland. Multidimensional spectral load balancing. Technical Report SAND93-0070 * UC-405, Sandia National Laboratories, Albuquerque, New Mexico, 1993.

[15] Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301 * UC-405, Sandia National Laboratories, Albuquerque, New Mexico, 1993.

[16] George Karypis and Vipin Kumar. Analysis of multilevel graph partitioning. Technical Report 95-037, University of Minnesota, Dept. of Computer Science, August 1995.

[17] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, University of Minnesota, Dept. of Computer Science, July 1995.

[18] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. Technical Report 95-064, University of Minnesota, Dept. of Computer Science, August 1995.

[19] B. N. Parlett and D. S. Scott. The Lanczos algorithm with selective orthogonalization. *Mathematics and Computations*, 33(11):217–238, 1979.

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

[21] Gilbert Strang. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, Inc., San Diego, CA, 1988.

[22] Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 298–301, 1989.

[23] Yen-Chuen Wei and Chung-Kuan Cheng. A two-level two-way partitioning algorithm. In *Proceedings of the IEEE ICCAD*, pages 516–519, 1990.

A Vertex Size distribution in benchmarks

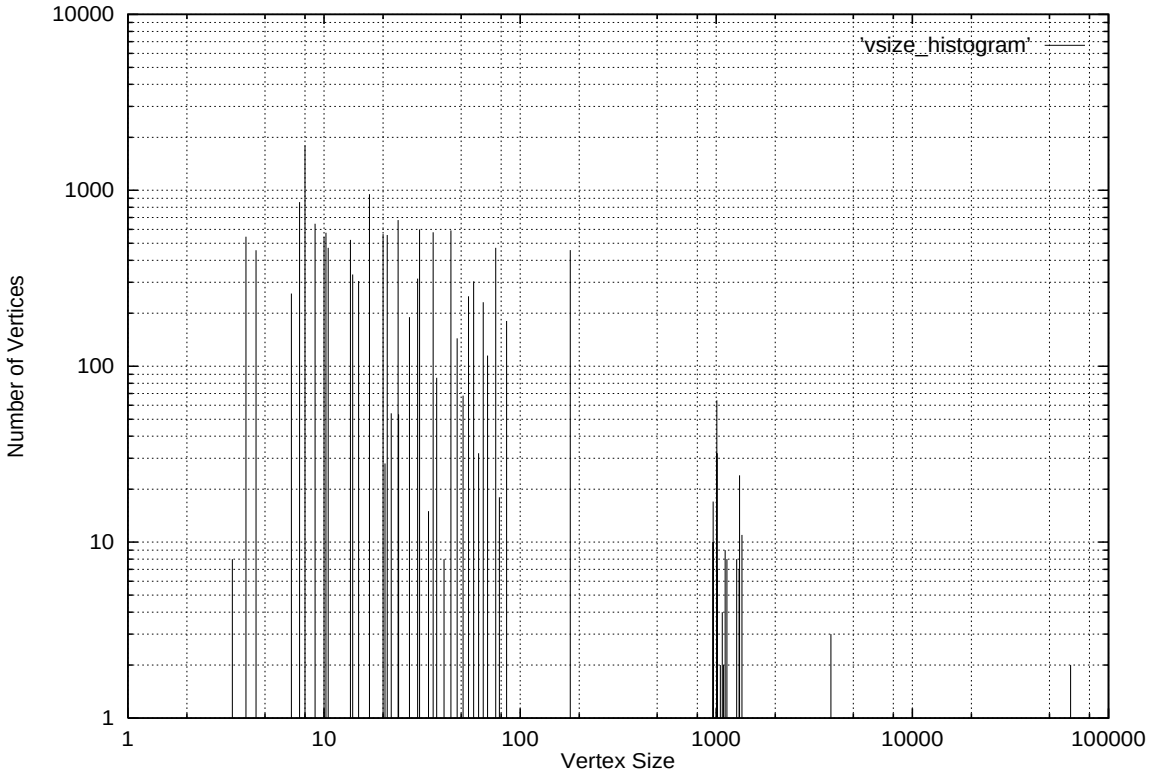

Figure A.1: Histogram of actual vertex sizes (scaled by 1000.0) in all seven benchmarks.