Spectral Partitioning: The More Eigenvectors, The Better*

Charles J. Alpert and So-Zen Yao†

UCLA Computer Science Department, Los Angeles, CA 90024-1596
† Cadence Design Systems, San Jose, CA 94135

Abstract

A *spectral* partitioning method uses the eigenvectors of a graph's adjacency or Laplacian matrix to construct a geometric representation (e.g., a linear ordering) which is then heuristically partitioned. We map each graph vertex to a vector in d-dimensional space, where d is the number of eigenvectors, such that these vectors constitute an instance of the *vector partitioning* problem. When all the eigenvectors are used, graph partitioning *exactly* reduces to vector partitioning. This result motivates a simple ordering heuristic that can be used to yield high-quality 2-way and multi-way partitionings. Our experiments suggest the vector partitioning perspective opens the door to new and effective heuristics.

1 Introduction

Given a netlist hypergraph, one may apply a transformation (e.g., replacing each hyperedge by a clique of weighted edges) to derive a *graph partitioning* instance. Such an instance consists of a weighted graph $G(V, E)$, i.e., vertex set $V = \{v_1, v_2, \ldots, v_n\}$, and symmetric $n \times n$ *adjacency matrix* $A = (a_{ij})$, where $a_{ij} > 0$ is the cost of $(v_i, v_j) \in E$, and $a_{ij} = 0$ if no (v_i, v_j) edge exists. Let $deg(v_i) = \sum_{j=1}^{n} a_{ij}$ be the *degree* of v_i. The $n \times n$ *degree matrix* $D = (d_{ij})$ is given by $d_{ii} = deg(v_i)$ and $d_{ij} = 0$ if $i \neq j$. The $n \times n$ *Laplacian* matrix of G is defined as $Q = D - A$.

Definition: A *k-way partitioning* of G is a set of *clusters* (subsets of V) $P^k = \{C_1, C_2, \ldots, C_k\}$ such that each $v_i \in V$ is a member of exactly one C_h, $1 \leq h \leq k$.

Almost all of the proposed multi-way partitioning objectives involve some combination of the number of cut edges and cluster size balance. We assume that the objective is to minimize the total cost of cut edges while satisfying cluster size constraints; however, the following discussion is applicable to other objectives as well.

Min-Cut Graph Partitioning: Given the adjacency matrix A corresponding to G, and cluster size lower and upper bounds L_h and W_h, find P^k that satisfies $L_h \leq |C_h| \leq W_h$ for each $1 \leq h \leq k$ and minimizes

$$f(P^k) = \sum_{h=1}^{k} E_h \quad where \quad E_h = \sum_{v_i \in C_h} \sum_{v_j \notin C_h} a_{ij}. \quad (1)$$

In other words, each E_h is the edge cut of cluster C_h.

Min-cut graph partitioning is NP-complete, and many types of heuristic methods have been proposed (see the recent netlist partitioning survey [4]). *Spectral* methods [1] [2] [6] [7] [8] [10] [12] [15] have been successful in recent years; these methods all use eigenvectors of the Laplacian or adjacency matrix to construct some type of geometric representation of the graph. We note examples of four such representations:

- **Linear orderings:** Hall [12] showed that the second eigenvector of the Laplacian yields the optimum 1-dimensional placement in terms of squared wirelength. Barnes [6] proposed a multi-eigenvector method that reduces to *spectral bipartitioning* (sorting the coordinates of the largest eigenvector of A) when $k = 2$. Hagen and Kahng [10] applied this approach to the ratio-cut objective and Riess et al. [15] used analytic methods to construct linear orderings.
- **Points in d-dimensional space:** Hall [12] also proposed constructing 2-dimensional graph placements with vertex coordinates derived from two eigenvectors. Alpert and Kahng [1] [2] extended this approach to higher dimensions, i.e., the i^{th} entries of d eigenvectors yield the coordinates of v_i in d-space. Geometric partitioning heuristics in [1] and a linear ordering heuristic in [2] were applied; dynamic programming was used to split the ordering into a k-way partitioning.
- **d-dimensional vectors:** Chan et al. [7] use the same representation as [12] [1] [2], but view each vertex as a *vector* rather than as a point in space. Their KP algorithm constructs partitioning solutions using the directional cosine between two vectors as a similarity measure between vertices.
- **Indicator vectors:** Any bipartitioning can be represented as an n-dimensional 0-1 *indicator vector*. Frankle and Karp [8] proposed sending *probes* into the d-dimensional space spanned by the best d eigenvectors; for a given probe, they find the indicator vector that maximally projects onto the probe in $O(n \log n)$ time.

*This work was performed during Charles J. Alpert's summer 1994 internship at Cadence Design Systems.

32nd ACM/IEEE Design Automation Conference ®

Our geometric representation is similar to that of the third approach, but we scale each coordinate by a function of the eigenvalue. The resulting vectors comprise a *vector partitioning* instance in which n vectors are partitioned into k subsets. The objective is to maximize the sum of the squared magnitudes of the k vectors that result by adding the vectors in each subset. Unlike directional cosines [7], this objective captures both vector magnitude and direction. Our representation is also similar to the indicator vector representation [8] since our n vectors are the indicator vectors for bipartitioning solutions of the form $P^2 = \{\{v\}, \{V - v\}\}$; hence, we extend the basic approach of [8] from 2-way to multi-way partitioning.

Our main contribution is the result that when n eigenvectors are used, the min-cut graph partitioning and max-sum vector partitioning objectives are identical. When $d < n$ eigenvectors are used, the vector partitioning instance is an approximation of graph partitioning, but as d gets larger, the instance provides a progressively better approximation, until it is exact for $d = n$. Hence, unlike works [1] [6] [7] that propose using $d = k$ eigenvectors for k-way partitioning, we believe that d should always be as large as practically possible.

The rest of our paper is organized as follows. Section 2 develops the relationship between eigenvectors and the Min-Cut objective. Section 3 introduces the vector partitioning problem and shows the equivalence between graph and vector partitioning. Section 4 presents a simple ordering heuristic. Section 5 presents experimental results for multi-way and 2-way partitioning, and Section 6 concludes with directions for future research.

2 Eigenvectors and Min-Cuts

In this section, we review terminology and key results that have appeared in the literature.

Definition: Given P^k, the corresponding $n \times k$ *assignment* matrix $X = (x_{ih})$ has $x_{ih} = 1$ if $v_i \in C_h$ and $x_{ih} = 0$ otherwise. $\vec{X_h}$, the h^{th} column of X, is the *indicator vector* for cluster C_h. Consequently, each row of X has sum one, and column h has sum $|C_h|$.

Theorem 1: $f(P^k) = trace(X^T Q X)$.

This theorem is a well-known extension to multi-way partitioning (e.g., [14]) of the bipartitioning result $\frac{1}{2} f(P^2) = \vec{X_1}^T Q \vec{X_1}$. Each $\vec{X_h}$ can be viewed as a linear combination of the n indicator vectors corresponding to singleton clusters, e.g., $[11001]^T = [10000]^T + [01000]^T + [00001]^T$. Alternatively, we can express $\vec{X_h}$ in terms of a different set of basis vectors, namely, the eigenvectors of the Laplacian.

Definition: An n-vector $\vec{\mu}$ is an *eigenvector* of Q with *eigenvalue* λ if and only if $Q\vec{\mu} = \lambda\vec{\mu}$. The eigenvectors of Q are given by $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_n}$ and have corresponding eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. The $n \times d$ *eigenvector matrix* $U_d = (\mu_{ij})$ has columns $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_d}$ (we use U for U_n) and the $n \times n$ *eigenvalue matrix* $\Lambda = (\Lambda_{ij})$ has diagonal entries $\Lambda_{ii} = \lambda_i$ and 0 entries elsewhere.

We assume that the eigenvectors are normalized, i.e., for $1 \leq j \leq n$, $\vec{\mu_j}^T \vec{\mu_j} = 1$. The eigenvectors of Q have many desirable properties, including [13]:

- The eigenvectors are all mutually orthogonal, hence they form a basis in n-dimensional space.
- $\lambda_1 = 0$ and $\vec{\mu_1} = [\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}]^T$.
- If G is connected then $\lambda_2 > 0$.

For simplicity, we assume that G is connected. Because the eigenvectors form a basis, any n-vector $\vec{x}$ can be expressed in terms of this basis. Since $UU^T = I$, we may write $\vec{x} = U(U^T\vec{x})$, or equivalently $\vec{x} = \sum_{j=1}^{n} (\vec{\mu_j}^T \vec{x}) \vec{\mu_j}$.

Definition: The $n \times k$ *projection matrix* $\Gamma = (\alpha_{jh})$ is given by $\alpha_{jh} = \vec{\mu_j}^T \vec{X_h}$, and thus $\Gamma = U^T X$. The h^{th} column of Γ is given by $\vec{\Gamma_h} = U^T \vec{X_h}$. We say that α_{jh} is the *magnitude* of the projection of $\vec{X_h}$ onto $\vec{\mu_j}$.

We can now write each indicator vector $\vec{X_h}$ in terms of the eigenvector basis, i.e., $\vec{X_h} = U\vec{\Gamma_h} = \sum_{j=1}^{n} \alpha_{jh} \vec{\mu_j}$. Notice that $|C_h| = ||\vec{X_h}||^2 = \sum_{j=1}^{n} \alpha_{jh}^2 = ||\vec{\Gamma_h}||^2$. Thus, $\vec{X_h}$ and $\vec{\Gamma_h}$ give two representations of the same indicator vector. It is easy to show (since $Q = U\Delta U^T$) that

Theorem 2: $trace(X^T Q X) = trace(\Gamma^T \Lambda \Gamma)$.

By combining Theorems 1 and 2 and performing the matrix multiplication, we observe:

Corollary 1:

$$f(P^k) = \sum_{h=1}^{k} \sum_{j=1}^{n} \alpha_{jh}^2 \lambda_j . \qquad (2)$$

This corollary is the k-way extension of the bipartitioning result [8]: $E_1 = \vec{X_1}^T Q \vec{X_1} = \sum_{j=1}^{n} \alpha_{j1}^2 \lambda_j$. This corollary gives the key insight behind our vector partitioning formulation. We now define the vector partitioning problem and show how graph partitioning reduces to vector partitioning.

3 The Vector Partitioning Problem

Definition: A k-way *vector partitioning* of a set of vectors Y is a set of k subsets $S^k = \{S_1, S_2, \ldots, S_k\}$ such that each $\vec{y} \in Y$ is a member of exactly one S_h, $1 \leq h \leq k$.

Max-Sum Vector Partitioning: Given a set Y of n vectors, and subset cardinality bounds L_h and W_h, find S^k that satisfies $L_h \leq |S_h| \leq W_h$ for each $1 \leq h \leq k$ and maximizes

$$g(S^k) = \sum_{h=1}^{k} ||\vec{Y_h}||^2 \quad where \quad \vec{Y_h} = \sum_{\vec{y} \in S_h} \vec{y}. \qquad (3)$$

We call $\vec{Y_h}$ the *subset vector* for S_h. Intuitively, the goal of vector partitioning is to find subsets of vectors that point in the same direction. Observe that Equation (3) induces an obvious pairwise similarity measure for vector partitioning, i.e., $||\vec{y_i} + \vec{y_j}||^2$ for $\vec{y_i}$

and $\vec{y_j}$. In contrast, graph partitioning has no natural analogous similarity measure (cf. the many ad hoc means in the literature such as all-pairs shortest paths, $k-l$ connectivity [9], etc.).

Because the min-cut objective of Equation (1) is to *minimize* f and the max-sum objective of Equation (3) is to *maximize* g, the two objectives appear incompatible. However, following [8], we may transform f into a maximization objective. Let $H \geq \lambda_n$ be some constant. Corollary 1 allows us to formulate the new maximization objective as

$$\begin{aligned} nH - f(P^k) &= \sum_{h=1}^{k}(H|C_h| - \sum_{j=1}^{n}\alpha_{jh}^2\lambda_j) \quad (4) \\ &= \sum_{h=1}^{k}\sum_{j=1}^{n}\alpha_{jh}^2(H-\lambda_j). \end{aligned}$$

The choice of $H \geq \lambda_n$ ensures that $nH - f(P^k) \geq 0$.

Definition: The $n \times d$ *scaled eigenvector matrix* $V_d = (\nu_{ij})$ is given by $\nu_{ij} = \mu_{ij}\sqrt{H-\lambda_j}$, i.e., by U_d with each column $\vec{\mu_j}$ scaled by $\sqrt{H-\lambda_j}$.

Let $\vec{y_i^d}$ denote the i^{th} row of V_d. Consider the d-dimensional vector partitioning instance with the vector set $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$, in which each graph vertex v_i corresponds to a vector $\vec{y_i^d}$. For this Y, we say that a graph partitioning $P^k = \{C_1, C_2, \ldots, C_k\}$ *corresponds to* a vector partitioning $S^k = \{S_1, S_2, \ldots, S_k\}$ if and only if $v_i \in C_h$ whenever $\vec{y_i^d} \in S_h$. Our main result shows that the min-cut graph partitioning and max-sum vector partitioning objectives are identical.

Theorem 3: If P^k corresponds to S^k, then for $d = n$,

$$nH - f(P^k) = g(S^k).$$

Proof: For a given $S_h \in S^k$, let $\vec{Y_h^d} = \sum_{\vec{y_i^d} \in S_h} \vec{y_i^d}$. First we establish that $||\vec{Y_h^d}||^2 = \sum_{j=1}^{d}\alpha_{jh}^2(H-\lambda_j)$, and the theorem will follow. For every i, the j^{th} component of $\vec{y_i^d}$ is ν_{ij}, hence the j^{th} component of $\vec{Y_h^d}$ is $\sum_{\vec{y_i^d} \in S_h} \nu_{ij}$. Therefore, computing the norm of $\vec{Y_h^d}$ by summing over components yields

$$||\vec{Y_h^d}||^2 = \sum_{j=1}^{d}(\sum_{\vec{y_i^d} \in S_h} \nu_{ij})^2$$

and since membership of $\vec{y_i^d} \in S_h$ corresponds to membership of $v_i \in C_h$,

$$= \sum_{j=1}^{d}(\sqrt{H-\lambda_j}\sum_{v_i \in C_h}\mu_{ij})^2 = \sum_{j=1}^{d}(\sqrt{H-\lambda_j}(\vec{\mu_j}^T\vec{X_h}))^2$$

$$= \sum_{j=1}^{d}(\sqrt{H-\lambda_j}(\alpha_{jh}))^2 = \sum_{j=1}^{d}\alpha_{jh}^2(H-\lambda_j)$$

Recalling that $\sum_{j=1}^{n}\alpha_{jh}^2 = |C_h|$, we have

$$||\vec{Y_h^n}||^2 = \sum_{j=1}^{n}\alpha_{jh}^2(H-\lambda_j) = H|C_h| - E_h. \quad (5)$$

Combining this result with Equations (3) and (4) yields

$$g(S^k) = \sum_{h=1}^{k}(H|C_h| - E_h) = nH - f(P^k). \quad \square$$

Corollary 2: Min-cut graph partitioning reduces to max-sum vector partitioning, hence max-sum vector partitioning is NP-Hard.

Corollary 3: $||\vec{y_i^n}||^2 = H - deg(v_i)$.

Equation (5) states that the magnitude of a subset vector gives the cost of the edges cut by its corresponding cluster. Corollary 3 is just a special case of this equation for $C_h = \{v_i\}$, i.e., the degree of each vertex can be computed from the magnitude of its corresponding vector.

Finally, we show that ignoring the first eigenvector $\vec{\mu_1}$ leads to the same vector partitioning formulation, but with a nice geometric property. The magnitude of the projection of C_h onto $\vec{\mu_1}$ is $\alpha_{1h} = \vec{\mu_1}^T\vec{X_h} = \frac{|C_h|}{\sqrt{n}}$. Hence, for any P^k

$$\sum_{h=1}^{k}\alpha_{1h}^2(H-\lambda_1) = \sum_{h=1}^{k}\frac{H|C_h|^2}{n}, \quad (6)$$

which is a constant, i.e., if the $\alpha_{1h}^2(H-\lambda_1)$ terms are discarded in Equation (2), and if the first component of each $\vec{y_i^n}$ is discarded, Theorem 3 still holds. Each eigenvector $\vec{\mu_j}$ $(2 \leq j \leq n)$ is orthogonal to $\vec{\mu_1}$, so for each such $\vec{\mu_j}$, we have $\sum_{i=1}^{n}\mu_{ij} = 0$. Thus, ignoring the first eigenvector yields the following desirable result.

Theorem 4: For any vector partitioning S^k of $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$, if the first component of each $\vec{y_i^d} \in Y$ is discarded, then $\sum_{h=1}^{k}\sum_{\vec{y} \in S_h}\vec{y} = \vec{0}$.

Theorem 4 states that the sum of all the subset vectors is the zero vector, e.g., for $k = 2$, the subset vectors $\vec{Y_1}$ and $\vec{Y_2}$ have the same magnitude and point in opposite directions.

An Example

Consider the 5-vertex graph with adjacency matrix

$$A = \begin{bmatrix} 0 & 4 & 0 & 0 & 1 \\ 4 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 4 \\ 1 & 0 & 0 & 4 & 0 \end{bmatrix}$$

The eigenvalues for the Laplacian of this graph are $\lambda_1 = 0.000, \lambda_2 = 2.298, \lambda_3 = 2.469, \lambda_4 = 8.702$ and $\lambda_5 = 10.531$, and the eigenvector matrix is given by

$$U_5 = \begin{bmatrix} 0.447 & 0.309 & 0.530 & 0.452 & 0.467 \\ 0.447 & 0.131 & 0.468 & -0.532 & -0.530 \\ 0.447 & -0.880 & 0.000 & 0.159 & 0.000 \\ 0.447 & 0.131 & -0.468 & -0.532 & 0.530 \\ 0.447 & 0.309 & -0.530 & 0.452 & -0.467 \end{bmatrix}$$

We choose $H = \lambda_5$, so that the last component of each vector is zero. The scaled eigenvector matrix is

$$V_5 = \begin{bmatrix} 1.451 & 0.886 & 1.505 & 0.612 & 0.000 \\ 1.451 & 0.377 & 1.329 & -0.719 & 0.000 \\ 1.451 & -2.526 & 0.000 & 0.215 & 0.000 \\ 1.451 & 0.377 & -1.329 & -0.719 & 0.000 \\ 1.451 & 0.886 & -1.505 & 0.612 & 0.000 \end{bmatrix}$$

By Theorem 4, we can ignore the first column of V_5, and our choice of H permits us to ignore the last column. The subset vectors corresponding to the bipartitioning $\{\{v_1, v_2, v_3\}\{v_4, v_5\}\}$ are $\vec{Y_1} = \vec{y_1} + \vec{y_2} + \vec{y_3} = [-1.263, 2.834, 0.108]^T$ and $\vec{Y_2} = \vec{y_4} + \vec{y_5} = [1.263, -2.834, -0.108]^T$. We have $||\vec{Y_1}||^2 = ||\vec{Y_2}||^2 = 9.637$. The constant from Equation (6) is $\sum_{h=1}^{2} \frac{10.531}{5}|C_h|^2 = 27.378$, hence $g(S^2) = 9.637 + 9.637 + 27.378 = 46.652$. The maximization objective from Equation (4) evaluates to $nH - f(P^2) = 5 \cdot 10.531 - (3 + 3) = 46.655$, and we see that the vector and graph partitioning objectives (except for rounding errors) are identical.

Other Objectives

We note that different graph partitioning objectives will induce corresponding vector partitioning objectives.

Chan et al. [7] proposed to minimize the *Scaled Cost* objective $f(P^k) = \sum_{h=1}^{k} \frac{E_h}{|C_h|}$ with no size constraints, i.e., a "generalized ratio-cut". Our maximization objective becomes $kH - f(P^k) = \sum_{h=1}^{k} \sum_{j=1}^{n} \alpha_{jh}^2 \frac{H-\lambda_j}{|C_h|}$. From Equation (5), we have $\frac{||\vec{Y}_h^n||^2}{|C_h|} = H - \frac{E_h}{|C_h|}$, which is the contribution of cluster C_h to our new objective. The vector partitioning objective becomes to maximize $g(S^k) = \sum_{h=1}^{k} \frac{||\vec{Y}_h||^2}{|S_h|}$, which captures an "average vector" formulation.

FPGA partitioning might require minimizing the maximum degree of any cluster, i.e., minimizing $f(P^k) = \max_{1 \le h \le k} E_h = \max_{1 \le h \le k} \alpha_{jh}^2 \lambda_j$, or equivalently, maximizing $\min_{1 \le h \le k} \alpha_{jh}^2 (H - \lambda_j)$. The corresponding vector partitioning objective becomes to maximize $g(S^k) = \min_{1 \le h \le k} ||\vec{Y}_h^n||^2$.

4 Linear Ordering Algorithm

We now propose a simple, greedy graph partitioning heuristic that utilizes the corresponding vector partitioning instance. Instead of explicitly solving vector partitioning, we construct a *linear ordering* of the vectors (and hence the vertices). Previous work [2] [10] [15] has shown the effectiveness of this approach.

In a good vector partitioning solution S^k, any given subset of vectors S_h will generally consist of vectors that sum to a vector of large magnitude. Thus, our MELO (Multiple Eigenvector Linear Orderings) algorithm in Figure 1 greedily constructs such a subset. MELO begins by constructing the d-dimensional vector partitioning instance and initializes the set of vectors S to empty (Steps 1-2). MELO then adds to S the vector with largest magnitude, removes this vector from Y, and labels the vector's corresponding vertex first in the ordering. Each subsequent iteration through Steps 4-6 chooses the "best vector" $\vec{y}_i^d$ remaining in Y, i.e., the one that maximizes the magnitude of the sum of the vectors in S plus $\vec{y}_i^d$. Throughout MELO's execution, S should consist of a reasonably good subset for vector partitioning. MELO's complexity is no worse that $O(dn^2)$, although speedups are possible (see [5]).

This idea of iteratively constructing a cluster (i.e., subset of vectors) seems to be the most direct method for constructing a linear ordering. MELO may not seem much different from a greedy graph traversals (cf. [3]); however, with our approach each added vector incorporates global partitioning information while a greedy graph traversal can only make local decisions.

The MELO Algorithm

Input: Graph $G(V, E)$, number of eigenvectors d
Output: A linear ordering of V

1. Construct scaled eigenvector matrix V_d, set $S = \emptyset$.
2. $Y = \{\vec{y}_i^d\}$, $1 \le i \le n$, where $\vec{y}_i^d$ is row i of V_d.
3. **for** $j = 1$ **to** n **do**
4. Find $\vec{y}_i^d \in Y$ that maximizes $||\sum_{\vec{y} \in S} \vec{y} + \vec{y}_i^d||$.
5. Add $\vec{y}_i^d$ to S and remove $\vec{y}_i^d$ from Y.
6. Label v_i as the j^{th} vertex in the ordering.

Figure 1: The MELO Algorithm

We may also motivate MELO in the context of Equation (4) for $k = 2$. In this case, the degree of the first cluster is the same as the degree of the second cluster (i.e., $E_1 = E_2$) and the objective equivalently becomes $H - \frac{f(P^2)}{2} = \sum_{j=1}^{n} \alpha_{j1}^2 (H - \lambda_j)$. Since in practice, we have only d of the eigenvectors, we cannot evaluate all the terms in this summation; however, the first d terms should provide a reasonable approximation, i.e., $H - \frac{f(P^k)}{2} \propto \sum_{j=1}^{d} \alpha_{j1}^2 (H - \lambda_j)$. If we can find a cluster C_1 that maximizes this summation, then $H - \frac{f(P^k)}{2}$ should be close to optimal. A reasonable algorithm for constructing C_1 might be to begin with $C_1 = \emptyset$, and then iteratively add the vertex to C_1 that optimizes this approximation. Such an approach is exactly equivalent to MELO.

We have not yet discussed how to choose H. When $d = n$, H is inconsequential; however, when $d \neq n$, the choice of H can affect the ordering. We tried four different schemes for scaling the eigenvectors (see [5] for more details): $H = \infty$, $H = \lambda_d + \lambda_2$, H chosen to minimize $\sum_{j=1}^{d} \frac{\alpha_{j1}^2}{\lambda_j}$, and $H = \frac{E_1 - \sum_{j=1}^{d} \alpha_{j1}^2 \lambda_j}{|C_h| - \sum_{j=1}^{d} \alpha_{j1}^2}$ for a given P^2 [8]. Our multi-way partitioning results indicate the second scheme is slightly better, averaging 1.70%, 0.01% and 1.41% improvement over the first, third and fourth schemes respectively. Our experiments in the next section use the second scheme for multi-way partitioning and the best results derived from the second, third, and fourth schemes for bipartitioning.

Test Case	Scheme	Number of Clusters - k									Avg % improv
		10	9	8	7	6	5	4	3	2	
19ks	MELO	9.85	9.10	8.31	7.87	6.84	6.14	5.32	4.99	4.79	+0.00
	EIG1	13.9	11.6	9.15	8.74	8.87	7.00	6.51	6.45	6.35	+18.9
	KP	9.09	9.37	10.7	9.52	9.00	9.00	6.95	6.58	6.20	+17.8
	SFC	15.1	14.3	13.8	13.2	12.2	11.1	8.37	7.48	5.44	+35.7
bm1	MELO	25.5	23.4	21.8	18.6	16.0	12.5	8.63	6.61	5.53	+0.00
	EIG1	33.3	31.1	26.9	22.7	17.0	11.3	8.63	6.61	5.53	+8.94
	KP	27.5	23.1	18.9	18.2	12.5	10.7	8.67	6.61	5.53	-4.97
	SFC	24.8	22.8	20.7	18.0	14.4	11.5	8.89	6.61	5.53	-3.17
prim1	MELO	44.6	41.9	39.7	37.0	34.0	29.4	22.5	17.1	13.4	+0.00
	EIG1	53.1	44.6	40.5	38.1	32.4	28.1	23.9	17.0	13.4	+2.51
	KP	44.7	41.3	32.3	33.2	31.3	29.9	21.2	14.7	13.5	-6.16
	SFC	38.9	36.7	35.2	31.7	28.8	26.0	21.8	14.6	13.4	-10.6
prim2	MELO	13.7	12.7	12.0	11.2	10.1	9.18	7.95	6.76	4.71	+0.00
	EIG1	11.2	10.9	10.1	9.73	9.33	8.33	7.85	7.69	5.55	-5.33
	KP	15.0	15.2	13.5	11.0	10.5	10.1	9.23	7.25	4.64	+7.38
	SFC	13.7	13.3	12.8	12.1	11.0	9.43	7.95	6.86	5.05	+4.14
test02	MELO	21.1	19.9	18.5	17.0	15.4	13.9	12.4	10.7	8.07	+0.00
	EIG1	31.3	31.4	20.2	29.8	28.6	28.8	18.4	18.2	12.4	+36.4
	KP	24.4	22.6	22.1	19.1	19.3	18.7	17.4	12.0	9.26	+16.8
	SFC	25.5	24.1	22.8	20.9	18.5	16.1	13.4	10.9	8.07	+12.4
test03	MELO	19.0	17.6	16.7	15.3	14.6	13.7	12.5	11.6	9.29	+0.00
	EIG1	20.3	19.9	17.7	17.0	16.7	17.6	16.0	14.3	11.9	+14.6
	KP	20.3	22.4	19.1	17.3	18.4	22.8	19.9	14.7	9.45	+19.2
	SFC	22.6	21.1	19.2	17.1	16.2	15.2	14.3	13.0	10.2	+12.0
test04	MELO	13.2	12.3	11.5	10.8	9.97	9.32	8.21	6.83	5.78	+0.00
	EIG1	14.3	12.5	11.8	11.1	10.2	10.3	9.08	8.65	5.85	+6.46
	KP	18.3	16.3	15.0	12.2	13.7	12.5	8.98	12.0	6.74	+22.9
	SFC	22.2	19.9	17.8	17.6	16.5	15.1	11.6	8.19	5.78	+30.7
test05	MELO	7.42	7.03	6.53	6.11	5.79	5.50	4.85	4.35	3.09	+0.00
	EIG1	7.50	6.82	6.70	6.12	5.38	4.97	4.26	4.06	3.09	-3.85
	KP	10.8	10.6	9.28	6.80	6.60	6.69	7.81	7.34	4.52	+27.2
	SFC	9.88	8.66	8.06	7.84	7.32	6.56	5.49	4.90	3.09	+16.1
test06	MELO	21.3	20.2	18.5	16.7	14.7	13.5	11.3	9.54	8.80	+0.00
	EIG1	17.8	17.3	16.0	15.6	16.2	17.3	19.9	15.6	14.3	+10.3
	KP	21.0	20.9	19.7	18.5	19.1	19.1	18.0	18.2	28.6	+24.9
	SFC	27.1	25.1	23.7	20.2	18.4	16.5	13.7	11.3	9.21	+17.3
balu	MELO	54.0	50.1	46.5	43.2	40.0	36.7	32.3	24.4	17.6	+0.00
	EIG1	72.2	76.6	81.8	91.3	84.6	74.5	57.8	55.4	47.2	+46.9
	KP	45.7	58.6	56.8	47.9	45.9	35.9	33.2	32.9	48.7	+14.2
	SFC	82.0	79.1	74.1	70.3	64.9	62.2	49.4	47.3	17.6	+34.3
struct	MELO	12.9	12.0	10.9	9.82	8.46	7.56	6.53	5.54	4.25	+0.00
	EIG1	9.52	9.15	8.72	7.96	8.03	6.60	6.15	5.68	4.85	-10.7
	KP	14.9	15.7	13.8	14.4	11.4	9.32	7.91	7.51	6.60	+23.8
	SFC	12.1	11.2	10.5	9.41	8.65	7.93	7.05	6.42	4.85	+2.04
biomed	MELO	1.87	1.73	1.62	1.49	1.34	1.23	1.11	0.89	0.61	+0.00
	EIG1	1.68	1.67	1.60	1.58	1.60	1.20	1.27	1.28	0.85	+8.29
	KP	3.22	3.03	2.65	1.75	1.63	1.36	1.83	1.16	0.84	+24.3
	SFC	1.84	1.69	1.59	1.47	1.51	1.48	1.25	1.15	0.85	+9.22
Average	EIG1	+4.39	+5.47	+2.64	+8.30	+11.8	+9.60	+13.9	+19.7	+20.0	+10.6
	KP	+10.2	+16.0	+13.4	+9.35	+12.8	+15.8	+20.1	+21.1	+23.2	+15.8
	SFC	+13.6	+13.5	+14.0	+14.0	+15.2	+15.4	+14.0	+13.2	+6.06	+13.2

Table 1: Scaled cost ($\times 10^{-5}$) comparisons for the MELO, EIG1 [10], KP [7], and SFC [2] algorithms.

5 Experimental Results

Our experiments use the set of ACM/SIGDA benchmarks listed in Table 2 (available via the World Wide Web at http://ballade.cs.ucla.edu/~cheese). Each netlist was first transformed into a graph using a clique net model that adds weight $\frac{4}{p(p-1)} \cdot \frac{2^p-2}{2^p}$ to each possible edge in a p-pin net [1], and the eigenvectors were then computed using LASO2 code.

To generate multi-way partitionings from MELO orderings, we apply the "DP-RP" algorithm of [2]. DP-RP accepts a vertex ordering and returns the optimal k-way partitioning such that each cluster is a contiguous subset of the ordering. We choose to minimize Scaled Cost since it has no size constraints, it provides a single quantity that measures the quality of a linear ordering, and permits easy comparison to previous algorithms. With the Scaled Cost objective and no cluster size bounds, DP-RP has $O(kn^2)$ time complexity.

Table 1 compares MELO with the multi-way partitioning algorithms EIG1 [10], KP [7], and SFC [2]. The rightmost column of the Table reports the average percent improvement of MELO versus the other schemes. Positive improvement is indicated with a +; if the improvement is negative, we report the percent improvement of the superior algorithm versus MELO and indicate this with a −. Note that the EIG1 and KP results reported in Table 1 are considerably improved from the values reported in [7] due to two factors: a different net model, and no thresholding of large nets (see [5]). The SFC results are quoted from [2]. The MELO results are the best observed from the ten linear orderings generated using the d best eigenvectors of Q (discarding $\vec{\mu}_1$) for $1 \leq d \leq 10$. These experiments were not performed for the larger benchmarks because DP-RP's $O(n^2)$ space requirement makes it infeasible on small workstations for netlists with around 7000-8000 modules. MELO averages 10.6%, 15.8% and 13.2% improvement over EIG1, KP and SFC respectively. MELO also seems to perform consistently well over the range of benchmarks and values of k.

In separate experiments [5], we compared the quality of MELO partitionings run on a single eigenvector ($d = 1$), on a 10-dimensional vector partitioning instance ($d = 10$), and the best combined of the ten orderings ($d = 1 - 10$), reported as MELO in Table 1. Overall, the $d = 1 - 10$ orderings averaged 32.6% improvement over the $d = 1$ orderings but only 6.43% improvement over the $d = 10$ orderings, i.e., the $d = 10$ orderings significantly outperformed the single eigenvector orderings. Indeed, the $d = 10$ orderings alone still yield lower average Scaled Cost than the EIG1,

Test Case	# modules	# nets	# pins	EIG1 cuts	EIG1 RC	PARABOLI cuts	PARABOLI RC	MELO cuts	MELO RC	Runtimes(s) $d=2$	Runtimes(s) $d=10$	improv % EIG1	improv % PARABOLI
19ks	2844	3282	10547	179	8.92			119	5.89	40	79	+34.0	
bm1	882	903	2910	75	38.9			48	30.0	4	9	+22.9	
prim1	833	902	2908	75	43.6	53	30.6	64	36.9	3	8	+15.4	-17.1
prim2	3024	3029	11219	254	11.3	146	6.47	169	7.48	26	89	+33.9	-13.6
test02	1663	1720	6134	196	28.4			106	15.4	9	29	+46.6	
test03	1607	1618	5807	85	13.2			60	9.29	9	27	+29.7	
test04	1515	1658	5975	207	36.2			61	10.6	8	24	+70.8	
test05	2595	2750	10076	167	9.93			102	6.07	20	67	+39.0	
test06	1752	1541	6638	295	38.5			90	11.7	10	31	+69.7	
balu	801	735	2697	110	69.2	41	25.8	28	17.6	3	7	+74.6	+31.8
struct	1952	1920	5471	49	5.16	40	4.20	38	4.00	12	38	+22.5	+4.77
biomed	6514	5742	21040	286	2.69	135	1.28	115	1.08	132	496	+59.9	+15.7
s9234	5866	5844	14065	166	1.95	74	0.86	79	0.92	108	516	+52.8	-6.53
s13207	8772	8651	20606	110	0.57	91	0.48	104	0.54	186	710	+5.27	-12.1
s15850	10470	10383	24712	125	0.46	91	0.33	52	0.19	308	1197	+58.7	+42.4
industry2	12637	13419	48404	525	1.32	193	0.49	319	0.81	478	1855	+38.6	-39.5

Table 2: Min-cut and ratio cut ($\times 10^{-5}$) comparisons for MELO bipartitionings versus PARABOLI [15] and EIG1 [10] such that each cluster contains at least 45% of the total modules. Missing entries indicate data unavailable.

SFC and KP heuristics.

Finally, we used MELO orderings to construct bipartitionings by choosing the one with lowest ratio cut (i.e., Scaled Cost for $k = 2$) from all possible splits of the ordering while ensuring that each cluster contains at least 45% of the modules. We quote the PARABOLI results of [15] for comparison, and additionally compare against EIG1.[1] Table 2 also reports Sun Sparc-10 runtimes required for MELO to construct and split orderings using two and ten eigenvectors, after the eigenvectors have been computed. Despite MELO's $O(dn^2)$ complexity, these runtimes seem quite reasonable (see [1] for detailed runtimes for eigenvector computations). The results from Table 2 do not suggest that MELO is a superior bipartitioning approach, but rather that multiple eigenvectors can be used to yield high-quality balanced bipartitionings. We hypothesize that directly solving the vector bipartitioning problem will improve these results even further.

6 Future Work

We note possible directions for future research:

- Modify MELO to run in sub-$O(n^2)$ time, e.g., by adding vectors to S only from a candidate set of fixed size.
- Find and apply lower bounds, e.g., we would like to be able to bound the quality of a d-dimensional approximation of the n-dimensional vector partitioning instance.
- Most importantly, vector partitioning heuristics seem worth exploring, e.g., an FM-type of local improvement method could be applied.

Acknowledgments

We thank Pak Chan, Martine Schlag and Jason Zien for giving us the LASO2, KP, and EIG1 codes and their entire set of experimental data. We also thank Jon Frankle for his useful discussions and insight.

[1] The circuits s9234, s13207, and s15850 contain multiple connected components, so we ran MELO and EIG1 on the largest component and added the remaining components to the smaller cluster, while ensuring that the 45% constraint was satisfied.

References

[1] C. J. Alpert and A. B. Kahng, "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning," *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 743-748.

[2] C. J. Alpert and A. B. Kahng, "Multi-way Partitioning Via Spacefilling Curves and Dynamic Programming," *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 652-657.

[3] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering," *IEEE Intl. Conf. on Computer-Aided Design*, 1994, pp. 63-67.

[4] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey," to appear in *Integration: the VLSI Journal*, 1995.

[5] C. J. Alpert and S.-Z. Yao, "Spectral Partitioning: The More Eigenvectors, the Better," *UCLA CS Dept. Technical Report*, #940036, October 1994.

[6] E. R. Barnes, "An Algorithm for Partitioning the Nodes of a Graph," *Siam J. Algorithms and Discrete Methods* (3)4, 1992, pp. 541-549.

[7] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD* 13(9), 1994, pp. 1088-1096.

[8] J. Frankle and R. M. Karp, "Circuit Placements and Cost Bounds by Eigenvector Decomposition," *IEEE Conf. Computer Aided Design*, 1986, pp. 414-417

[9] J. Garbers, H. J. Promel and A. Steger, "Finding Clusters in VLSI Circuits" *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 520-523.

[10] L. Hagen and A. B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", *Proc. IEEE Intl. Conf. Computer-Aided Design*, 1991, pp. 10-13.

[11] L. Hagen and A. B. Kahng, "A New Approach to Effective Circuit Clustering", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1992, pp. 422-427.

[12] K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Manag. Sci.*, 17(1970), pp. 219-229.

[13] B. Mohar, "The Laplacian Spectrum of Graphs", *Proc. 6th Quadrennial Intl. Conf. on Theory and Applications of Graphs*, 1988, pp. 871-898.

[14] F. Rendl and H. Wolkowicz, "A Projection Technique for Partitioning the Nodes of a Graph", *Univ. Waterloo Technical Report*, May 1994.

[15] B. M. Riess, K. Doll, and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 646-651.