Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

FAST SPECTRAL METHODS FOR RATIO CUT PARTITIONING AND CLUSTERING

Lars Hagen
Andrew B. Kahng

April 1991
CSD-910012

Fast Spectral Methods for Ratio Cut Partitioning and Clustering

Lars Hagen and Andrew Kahng

UCLA Department of Computer Science
Los Angeles, CA 90024-1596

Abstract

Partitioning of circuit netlists is important in many phases of VLSI design, ranging from layout to testing and hardware simulation. The *ratio cut* objective function [27], though NP-complete, has received much attention since it naturally embodies both min-cut and equipartition, the two traditional goals of partitioning. Fiduccia-Mattheyses style ratio cut heuristics have resulted in average cost savings of 39% for circuit partitioning and over 50% savings for hardware simulation applications [29]. In this paper, we present several new results for ratio cut computation. First, we show a new theoretical correspondence between the optimal ratio cut partition cost and the second smallest eigenvalue of a particular matrix derived from the netlist. This yields a provably good approximation of the *optimal* ratio cut value. Second, we demonstrate that fast Lanczos-type methods for the sparse symmetric eigenvalue problem are a robust basis for computing heuristic ratio cuts based on the eigenvector of the second eigenvalue. We have tested our algorithm, EIG1, on standard-cell and gate-array industry benchmarks. Results improve those of the RCut1.0 algorithm of Wei and Cheng [27], e.g., by an average of 17% for the Primary MCNC benchmarks, while using similar computational resources. For example, we derive the heuristic eigenvector-based partition of the 3014-module PrimGA2 netlist in 83 seconds on a Sun-4/60 workstation. We also tested additional benchmark netlists using uniform module weights, reflecting applications of partitioning to test or hardware simulation; our method gave an average of 9% improvement over RCut1.0 on the MCNC Primary/Test netlists and two benchmarks from Hughes. Finally, an efficient *clustering* method, also based on the second eigenvector, is very successful on all of the "difficult" input classes that have been proposed in the CAD literature. The spectral method is appealing because it uses global information while iterative methods rely on local information; modules also in some sense make a continuous rather than discrete choice of location. Finally, we use a *single* numerical computation rather than multiple computations from random starting points. The paper concludes by describing several types of algorithm speedups and directions for future work.

1 Preliminaries

As system complexity increases, the divide-and-conquer approach is used to keep the circuit design process tractable. The recursive decomposition of the synthesis problem is reflected in the hierarchical organization of boards, multi-chip modules, integrated circuits, macros, etc. Since early decisions will constrain all succeeding decisions, the high-level layout phases are critical to the quality of the final layout. In particular, without a successful partitioning algorithm, good solutions to the placement, global routing and detailed routing problems will be impossible. As noted in, e.g., [7], partitioning

comprises the essence of many basic CAD problems, including

- **Packaging of designs:** logic is partitioned into modules, subject to constraints on module area as well as I/O bounds; this is the canonical partitioning application at all levels of the design process, and it also arises whenever technology improves and existing designs must be repackaged onto higher-capacity modules.
- **Clustering analysis:** in many layout approaches, partitioning is used to derive a sparse, clustered netlist which is then used as the basis of constructive module placement.
- **Partition analysis for high-level synthesis:** accurate prediction of layout area and wireability is crucial to high-level synthesis and floorplanning, and predictive layout models are made by fitting analysis of the partitioning structure of netlists to models of the output characteristics of particular place/route algorithms.

Because signal delays will typically decrease as we move downward in the design hierarchy (e.g., on-chip communication is faster than inter-chip communication), the traditional metric for the decomposition is the number of signal nets which cross between layout subproblems. Minimizing this number is the essence of partitioning.

1.1 Basic Partitioning Formulations

A standard mathematical model in VLSI layout associates a graph $G = (V, E)$ with the circuit netlist; vertices in V represent modules and edges in E represent signal nets. The vertices and edges of G may be weighted to reflect module area and the multiplicity or importance of a connection. Because nets often have more than two pins, the netlist is more generally represented by a *hypergraph* $H = (V, E')$, where hyperedges in E' are the subsets of V contained by each signal [23]. There are several standard transformations from the hypergraph representation of a circuit to a graph representation, and partly for this reason a large portion of the literature has treated graph partitioning, rather than hypergraph partitioning. Section 3 treats the hypergraph transformation in more detail.

Two basic (graph) formulations for circuit partitioning are the following:

- **Minimum Cut:** Given $G = (V, E)$, find the ***min-cut*** partition of V into disjoint U and W such that the number of edges $e = \{u, w\}$, $u \in U$ and $w \in W$, is minimized.

- **Minimum-Width Bisection:** Given $G = (V, E)$, find the partition of V into disjoint U and W, with $|U| = |W|$, such that the number of edges $e = \{u, w\}$, $u \in U$ and $w \in W$, is minimized.

By the max-flow min-cut theorem of Ford and Fulkerson [10], a minimum cut separating prescribed nodes s and t can be found by flow techniques in $O(n^3)$ time, where $n = |V|$. Cut-tree techniques [5] yield the global min cut using $n - 1$ minimum cut computations in $O(n^4)$ time. This time complexity is rather high, and furthermore the minimum cut can divide modules very unevenly (Figure 1).

Because the minimum-width bisection divides module area equally, it is a more desirable metric, particularly with a hierarchical layout approach. Unfortunately, minimum-width bisection is NP-complete [13], so heuristic methods must be used. Approaches in the literature fall naturally into several classes. The top-down recursive bipartitioning method of such authors as Charney, Breuer and Schweikert [4] [7] [23] repeatedly divides the logic until subproblems become small enough for layout. Clustering and aggregation algorithms map logic to a prescribed floorplan in a bottom-up fashion, using seeded modules or analytic methods (e.g., Vijayan [26]). It is also possible to look for natural clusters in the circuit graph, as in the recent work of Garbers et al. [12].

In production software, iterative improvement is a nearly universal approach, either as a postprocessing refinement to other methods or as a method in itself. The iterative improvement is based on *local* perturbation of the current solution and can be either greedy (the Kernighan-Lin method [16] [23] and its algorithmic speedups by Fiduccia and Mattheyses [9] and Krishnamurthy [18]), or hill-climbing (the simulated annealing approach of Kirkpatrick et al. [17], Sechen [24] and others). Virtually all implementations will also use multiple random starting configurations [20] [27] in order to adequately search the solution space and yield some measure of "stability", i.e., predictable performance.

An important class of partitioning approaches, particularly in relation to the present work, consists of "spectral" methods which use eigenvalues or eigenvectors of matrices that are derived from the netlist graph. Recall that the circuit netlist may be represented by the simple undirected graph $G = (V, E)$ with $|V| = n$ vertices $v_1, \ldots, v_n$. Often, we use the $n \times n$ *adjacency matrix* $A = A(G)$, where $a_{vw} = 1$ if $(v, w) \in E$ and $a_{vw} = 0$ otherwise. If G has weighted edges, then a_{vw} is equal to the weight of $(v, w) \in E$, and by convention $a_{vv} = 0$ for all $v \in V$. If we let $d(v)$ denote the degree of node v (i.e., the sum of the weights of all edges incident to v), we get the $n \times n$ *diagonal matrix* D defined by $D_{ii} = d(v_i)$. The eigenvalues and eigenvectors of such matrices are the subject of the relatively recent subfield of graph theory dealing with *graph spectra* [6].

Early theoretical work connecting graph spectra and partitioning is due to Barnes, Donath and

Hoffman [1] [7] [8]. More recent eigenvector and eigenvalue methods have dealt with both module placement (Frankle and Karp [11] and Tsay and Kuh [25]) and graph min-cut bisection (Boppana [2]). In general, these previous works formulate the partitioning problem as the assignment or placement of nodes into bounded-size clusters or chip locations. The problem is then transformed into a quadratic optimization, and Lagrangian relaxation is used to derive an eigenvector formulation.

A prototypical example is the work of Hall [15], which we now outline. This work is particularly relevant since it uses eigenvectors of the same graph-derived matrix $Q = D - A$ (the same D and A defined above) that we discuss in Section 2 below. Donath and Hoffman, Boppana, and others use different matrices derived from the netlist graph, but exploit similar mathematical properties (e.g., symmetry, positive-definiteness) to derive alternate eigenvalue formulations and relationships to partitioning.

Hall's result [15] was that the eigenvectors of the matrix $Q = D - A$ solve the quadratic placement problem of finding the vector $\underline{x} = (x_1, x_2, \ldots, x_n)$ which minimizes

$$z = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2 a_{ij}$$

subject to the constraint $|\underline{x}| = (\underline{x}^T \underline{x})^{1/2} = 1$, with a_{ij} again equal to the strength of the connection between modules i and j.

It can be shown that $z = \underline{x}^T Q \underline{x}$, so that to minimize z we may form the Lagrangian

$$L = \underline{x}^T Q \underline{x} - \lambda(\underline{x}^T \underline{x} - 1).$$

Taking the first partial derivative of L with respect to $\underline{x}$ and setting it equal to zero yields

$$2Q\underline{x} - 2\lambda\underline{x} = 0,$$

and this can rewritten as

$$(Q - \lambda I)\underline{x} = 0$$

where I is the identity matrix. This is readily recognizable as an eigenvalue formulation for λ, and the eigenvectors of Q are the only nontrivial solutions for $\underline{x}$. The minimum eigenvalue 0 gives the uninteresting solution $\underline{x} = (1/\sqrt{n}, 1/\sqrt{n}, \ldots, 1/\sqrt{n})$, and hence the eigenvector corresponding to the second smallest eigenvalue λ is used. Note the several transformations inherent in this type of derivation, e.g., the requirement that $|x| = 1$. the relationship between minimum-cut bisection and the second eigenvector is hence somewhat indirect.

1.2 Ratio Cuts

One easily notices that requiring an exact bisection, rather than, say, a 60% - 40% partition of module area, is unnecessarily restrictive. In the instance of Figure 1, even the optimal bisection will not yield a very sensible partitioning. Penalty functions as in the r-bipartition method of [9] have been used to permit not-quite-perfect bisections. However, these methods can require rather ad hoc thresholds and penalties. This leads us to what is perhaps the most natural metric for partitioning, the *ratio cut* (also known as a *sparse* cut or *flux* cut) recently developed by Wei and Cheng [27] [28], and separately by Leighton and Rao [19]. This is formulated for bipartitioning as follows.

- **Minimum Ratio Cut:** Given $G = (V, E)$, find the partition of V into disjoint U and W such that $\frac{e(U,W)}{|U| \cdot |W|}$ is minimized, where $e(U, W)$ is the number of edges $e = \{u, w\}$, $u \in U$ and $w \in W$.

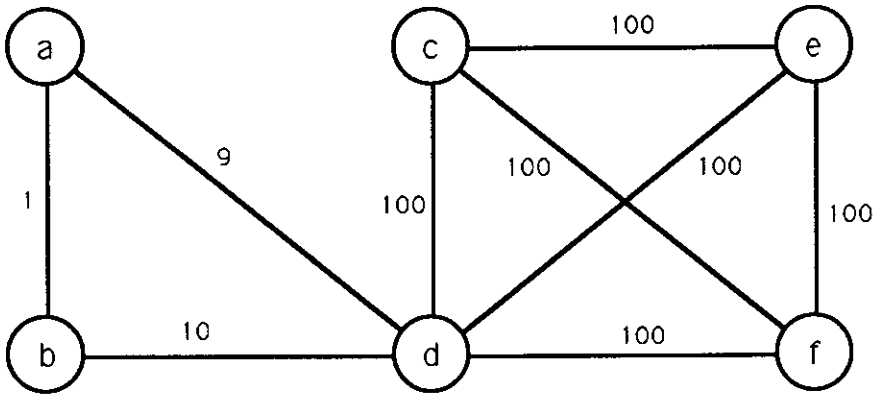

Figure 1: The minimum cut $a|bcdef$ will have cost = 10, but gives a very uneven partition. The optimal bisection $abd|cef$ has cost = 300, much worse than the more natural partitioning $ab|cdef$ which has cost = 19 and is the optimal ratio cut.

The ratio cut metric gives the "best of both worlds" in the sense that the numerator embodies minimum-cut, while the denominator favors an even partition (see Figure 1). Recent work shows that this metric is extremely useful; [27] reports average cost improvements of 39% over results from the standard Fiduccia-Mattheyses method [9] on industry benchmarks. The ratio cut also has important advantages in other areas of CAD: (i) in design for testability, a sparse partition will result in subcircuits that have fewer I/O's, thus requiring fewer test vectors; and (ii) in hardware simulation, runtime is proportional to the number of interconnections at a given level of the hierarchy, so a sparse cut reduces simulation costs. Wei and Cheng [27] [28] and the recent Ph.D. thesis of Wei [29] report

extraordinary cost savings of up to 70% for such applications in a number of industry settings, and ratio cut partitioning has indeed attracted widespread interest. Unfortunately, one can show that finding the minimum ratio cut of a graph G is NP-complete by reduction from Bounded Min-Cut Graph Partition in [13]. Multicommodity flow based approximation methods [5] [19] have been proposed, but are prohibitively expensive for large problems. Wei and Cheng [27] [28] therefore use an adaptation of the shifting and group swapping iterative improvement scheme of Fiduccia-Mattheyses, which was originally employed for min-cut bisection.

In this paper, we present new results indicating that spectral heuristics based on matrix eigenvalues are extremely useful for computing provably good ratio cuts. This conclusion is based on new theoretical results as well as fast numerical algorithms for directly computing a heuristic ratio cut in the circuit graph. Our approach exhibits a number of desired attributes, including:

- *speed*, i.e., low-order complexity compatible with user requirements; the method also parallelizes perfectly and allows tradeoffs between solution quality and CPU cost;
- *provability*, i.e., a lower bound on the optimal solution cost; note that current iterative and annealing methods cannot provide such a bound and indeed have unbounded error, particularly for "difficult" instances [3];
- *stability*, i.e., predictable performance, which in our case does not require any of the standard devices such as taking the best of several solutions derived using multiple random starting points; and
- *scaling* of solution quality with increasing problem size; by contrast, current iterative methods suffer from the "error catastrophe" that is common to local search heuristics for combinatorial problems [20].

We find that such features are becoming more crucial in a partitioning algorithm: the user requires rapid and correct evaluation of implementation choices during synthesis, and the rapidly growing number of design alternatives is making this harder to achieve.

The remainder of our paper is organized as follows. In Section 2, we show a new theoretical connection between graph spectra and the optimal ratio cut. Section 3 presents EIG1, our basic spectral heuristic for minimum ratio cut partitioning and clustering analysis. We derive a good ratio cut partition directly from the eigenvector associated with the second eigenvalue of $Q = D - A$. The spectral approach, since it uses *global* information, will give a qualitatively different result than

the standard iterative methods which rely on *local* information. Furthermore, modules in some sense make a continuous, rather than discrete, choice of location within the partition. Section 4 gives performance results and comparisons with previous work, using benchmarks from the MCNC suite and other industry sources, as well as classes of "difficult" inputs from the literature. Our method yields significant improvements over the previous ratio-cut partitioning methods of Wei and Cheng [27] [28], and for both partitioning and clustering applications we derive essentially optimal results for the difficult problem classes of [3] and [12]. Section 5 presents extensions and directions for future work, and our conclusions are contained in Section 6.

2 A New Connection: Graph Spectra and Ratio Cuts

In this section, we develop the theoretical basis of the method. Our notation is for graphs, but it is straightforward to model netlist hypergraphs as graphs via the usual hyperedge models. The experimental results reported below reflect such transformations of the input hypergraph.

Recall that two standard matrices derivable from the circuit netlist are the adjacency matrix A and the diagonal degree matrix D. We use the matrix $Q = D - A$ mentioned above, which we may view as the discrete analog of the Laplace Δ operator. Following are several basic properties of Q:

- (1) Q is symmetric.
- (2) Q is non-negative definite, so that: (i) $xQx = \sum_{i,j} q_{ij} x_i x_j \geq 0$; (ii) all eigenvalues of Q are ≥ 0; (iii) $\exists\ B$ such that $Q = B^T B$.
- (3) The smallest eigenvalue of Q is 0 with eigenvector $1 = (1, 1, ..., 1)$, since $Q1_i = (D - A)(1)_i = d_i - \sum_j a_{ij} \cdot 1 = d_i - d_i = 0$. Defining λ to be the second smallest eigenvalue of Q, we note that G is connected iff $\lambda > 0$ (by a theorem of Gershgorin [21], the multiplicity of the zero eigenvalue gives the number of connected components of G).

We may also show:

- (4) Using the notation $\sum' \equiv \sum_{(i,j) \in E}$ to denote summation over all edges,

$$(x, Qx) = xDx - xAx = \sum_i d_i x_i^2 - \sum_{i,j} a_{ij} x_i x_j = \sum_i d_i x_i^2 - 2 \sum{}' x_i x_j$$

and completing the square we have $(x, Qx) = \sum'(x_i - x_j)^2$.

- (5) Finally, the Rayleigh Principle [14] implies $\lambda = \min_{x \perp \mathbf{1}, x \neq 0} \frac{xQx}{|x|^2}$.

We use properties (4) and (5) to establish the main theoretical contribution of this paper, which is a new relationship between the *optimal ratio cut cost* and the second eigenvalue λ of $Q = D - A$.

Theorem One: Given a netlist graph $G = (V, E)$ with adjacency matrix A, diagonal degree matrix D, and $|V| = n$, the second smallest eigenvalue λ of $Q = D - A$ yields a lower bound on the cost c of the optimal ratio cut partition, with $c \geq \frac{\lambda}{n}$.

Proof: The optimal partition will divide V into disjoint U and W such that the ratio cut cost $\frac{e(U,W)}{|U| \cdot |W|}$ is minimum. We may rewrite this as $|U| = pn$, $|W| = qn$, with $p, q \geq 0$ and $p + q = 1$, and define the vector x by

$$x_i = \begin{cases} q & \text{if } v_i \in \text{U} \\ -p & \text{if } v_i \in \text{W} \end{cases}$$

Then we have the following:

- x is perpendicular to $\mathbf{1}$, since by construction $x \cdot \mathbf{1} = 0$.
- Since $x_i - x_j = q - (-p) = 1$ for edges e_{ij} crossing the partition, we will have

$$(x, Qx) = \sum_{i,j}{}' (x_i - x_j)^2 = \sum_{i,j \in U}{}' (x_i - x_j)^2 + \sum_{i,j \in W}{}' (x_i - x_j)^2 + \sum_{i \in U, j \in W}{}' (x_i - x_j)^2 = e(U, W)$$

- We further note that $|x|^2 = q^2 pn + p^2 qn = pqn(p + q) = pqn = \frac{|U| \cdot |W|}{n}$.
- Since $\min_{x \perp \mathbf{1}} \frac{xQx}{|x|^2} = \lambda$ from property (5) above, we have $\frac{(x,Qx)}{|x|^2} = \frac{e(U,W)n}{|U| \cdot |W|} \geq \lambda$, implying $\frac{e(U,W)}{|U| \cdot |W|} \geq \frac{\lambda}{n}$, and this gives the lower bound of Theorem One. □

We note the following: (i) the $\frac{\lambda}{n}$ lower bound in Theorem One for the optimal partition cost under the ratio cut metric is a tighter result than can be obtained using the early techniques of Donath et al. (which are essentially based on the Hoffman-Wielandt inequality [1]); and (ii) if we restrict the partition to be an *exact* bisection, Theorem One implies the same bound shown by Boppana [2], but our direct derivation from the optimal ratio cut partition allows our result to subsume the result of Boppana.

Given the result of Theorem One, the basic idea is to compute $\lambda(Q)$ and the corresponding eigenvector v, then use v to construct a heuristic ratio cut.

3 New Heuristics for Ratio Cut Partitioning

A basic heuristic algorithm template is shown in Figure 2.

$H = (V, E')$ = input netlist hypergraph;
Transform H into graph $G = (V, E)$;
Compute A = adjacency matrix, D = diagonal matrix of G;
Compute second-smallest eigenvalue $\lambda(Q)$ of $Q = D - A$;
Compute v, the real eigenvector associated with $\lambda(Q)$;
Map v into a heuristic ratio cut partition of H.

Figure 2: High-level description of basic spectral approach for ratio cut partition of netlist hypergraph H.

Clearly, a practical implementation of this approach requires closer examination of four main issues: (i) the transformation of the netlist hypergraph into a graph G; (ii) the calculation of the second eigenvector v; (iii) the construction of a heuristic ratio cut partition from v; and (iv) a possible post-processing stage to improve the heuristic ratio cut. The following subsections address these aspects in greater detail.

3.1 Hypergraph Model

We have examined two heuristic mappings from hyperedges in the netlist to graph edges in G. The first mapping is via the standard clique model (where a k-pin net is represented by a complete graph on its k modules, with each edge weight equal to $1/(k-1)$). The second mapping is given by using the standard clique model followed by an added sparsifying heuristic: we have considered two methods of sparsifying Q: (i) ignoring less significant (e.g., non-critical or very large) nets, and (ii) thresholding small Q_{ij} to 0 until the matrix has sufficiently few nonzeros.

The second type of transformation is important because our numerical algorithms can be much more efficient on sparse input. Preliminary experiments with the sparsifying heuristics, as well as a new *cycle* net model which also yields a sparser Q matrix, are quite promising. However, the results reported below are based on the standard weighted clique model; this already gives very good results and it is furthermore consistent with usual CAD practice.

3.2 Numerical Methods

With regard to algorithm implementation, it may at first appear that eigenvalue computations are too complicated to be practical. However, there are significant algorithmic speedups based on our need to calculate only a *single* (the second-smallest) eigenvalue of a *symmetric* matrix. Furthermore, while tradeoffs between netlist sparsity and runtime are implicit in the Lanczos method, note that netlist graphs tend to be very sparse due to degree bounds imposed by the technology and the hierarchical circuit organization. This allows us to apply sparse numerical techniques, particularly the block Lanczos algorithm [14].

We use an adaptation of an existing Lanczos implementation [21]. There are two interesting points to consider. First, the code calculates the second-*largest* eigenvalue and the corresponding eigenvector of the matrix $Q' = A - D$. This is equivalent to computing the second-smallest eigenvector of $Q = D - A$, i.e., we compute $-\lambda$ and $-v$, and is preferable by theoretical results of Kaniel-Paige-Saad [14] which show that the Lanczos algorithm converges faster to the largest eigenvalues. The numerical code is portable Fortran77; all other code in our system is written in C. Our experiments have been run on Sun-4 hardware.

3.3 Constructing the Ratio Cut

We have also considered a number of heuristics for constructing the ratio cut partition from the second eigenvector v: (i) construct $\overline{x} = sgn(v)$, i.e., $U = \{i : v_i \geq 0\}$ and $W = \{i : v_i < 0\}$; (ii) partition the nodes around the median v_i value, putting the first half in U and the second half in W; (iii) exploit the relation between the eigenvector computation and quadratic placement, whereby a "large" gap in the sorted list of v_i values indicates a natural division (cf. Section 3.4 below); and (iv) sort the x_i, then determine the splitting rank r, $1 \leq r \leq n-1$, that yields the best ratio cut cost when nodes with rank $> r$ are placed in U and nodes with rank $\leq r$ are placed in W. We use (iv) as the basis for the EIG1 algorithm below, since it subsumes the other three methods. Method (iv) requires a factor of $O(n)$ more effort than the other methods, but this cost of evaluating the partitions is asymptotically dominated by the cost of the Lanczos computation.

With these implementation decisions, our algorithm EIG1 is as follows:

As shown in the next section, EIG1 generates initial partitions which are already significantly better than the output of RCut1.0, the iterative Fiduccia-Mattheyses style algorithm of Wei and Cheng [27]. In fact, using the single sorted eigenvector we often find *many* partitions that are better than

Algorithm EIG1
$H = (V, E')$ = input netlist hypergraph;
Transform each k-pin hyperedge of H into a clique in $G = (V, E)$
    with uniform edge weight $\frac{1}{k-1}$;
Compute A = adjacency matrix, D = diagonal matrix of G;
Compute second-largest eigenvalue of $Q' = A - D$ by block
    Lanczos algorithm (equivalent to $\lambda(Q)$);
Compute associated real eigenvector v;
Sort components of v and find best splitting point for indices
    (i.e., modules) using ratio cut metric.

Figure 3: High-level outline of **Algorithm EIG1**.

the Fiduccia-Mattheyses result. Therefore, we have deferred investigation of iterative post-processing improvements.[1]

3.4 Clustering Is "Free"

A number of authors have noted that finding natural clusters is useful, e.g., (i) when multi-way partitioning is desired or when the sizes of the partitions are not known a priori; (ii) for purposes of floorplanning or constructive placement; and (iii) when the input is so large that clustering must be used to reduce the size of input to a partitioning algorithm. We conclude this section with the observation that clustering is "free" with our approach, in that the second eigenvector v contains both partitioning *and clustering* information. In Section 4 below, we demonstrate that the sorted second eigenvector by itself can effectively identify natural clusters in the classes of "difficult" inputs proposed in Bui et al. [3] and Garbers et al. [12].

More sophisticated interpretations of the second eigenvector are currently being examined. For example, because of the well-known correspondence between $\lambda(Q)$ and quadratic placement formulations, it is reasonable to interpret large *gaps* between adjacent components of the sorted eigenvector as delimiters of natural circuit clusters. We have considered using "local minimum" partitions in the sorted eigenvector, i.e., those with lower cost than either of their neighbor partitions, to delineate clusters. In other words, all node indices between consecutive local minima are placed together into a cluster. This is very reasonable since, as noted above, there are usually many distinct, good (local minimum) partitions that are derivable from the second eigenvector. Initial experiments show this approach,

[1] **Note to Reviewers:** Since absolutely no iterative improvement was attempted in our tests, our current results are at some disadvantage and should be interpreted accordingly. Indeed, preliminary experiments show that the eigenvector partitions can be improved by iterative post-processing. Partition improvement code for Fiduccia-Mattheyses group-shifting and swapping is being developed and should be complete within several weeks.

which is distinct from previous methods that required multiple eigenvectors, to be very promising.

4 Experimental Results

4.1 Ratio Cut Partitioning

In this section, we present computational results using the EIG1 algorithm. We initially ran this heuristic on the MCNC Primary1 and Primary2 standard-cell and gate-array benchmarks. Table 1 compares our results with the *better* of the results of the RCut1.0 program as reported in the Ph.D. thesis of Wei [29] and in the earlier paper of Wei and Cheng [27]. (Those reported results are already an average of 39% better than Fiduccia-Mattheyses output in terms of the ratio cut metric; [27] compared the best of 10 Rcut1.0 runs to the best of 20 F-M runs, all with random starting seeds.)

Test problem	Number of elements	Wei-Cheng (RCut1.0)			Hagen-Kahng (EIG1)			R_{HK}/R_{WC} Ratio
		Areas	Nets cut	Ratio cut	Areas	Nets cut	Ratio cut	
PrimGA1	833	502:2929	11	7.48×10^{-6}	751:2681	15	7.45×10^{-6}	.989
PrimGA2	3014	2488:5885	89	6.08×10^{-6}	2522:5852	78	5.29×10^{-6}	.855
PrimSC1	833	1071:1682	35	1.94×10^{-5}	588:2166	15	1.18×10^{-5}	.602
PrimSC2	3014	2332:5374	89	7.10×10^{-6}	2361:5345	78	6.18×10^{-6}	.859

Table 1: Comparison with best values reported in [27][29] on standard-cell and gate-array benchmark netlists (area sums may vary due to rounding). On average, the EIG1 results are 17.6% better. Note that EIG1 results are completely unrefined: no local improvement has been performed on the eigenvector partition.

The CPU times required by our algorithm were very competitive with those cited in [27]: for example, the eigenvector computation for PrimSC2, using our default convergence tolerance of 10^{-4}, required 83 seconds of CPU time on a Sun4/60, versus 204 seconds of CPU for 10 runs of RCut1.0.

A critical aspect of the current eigenvector computation is that one cannot naturally force a good *area* partition since graph nodes are unweighted; by contrast, the Fiduccia-Mattheyses method inherently exploits module area information. Our results were obtained by applying the Lanczos code to the netlist, then using the actual module areas [2] to determine the best split of the sorted eigenvector under the ratio cut metric. Thus, the approach is well-suited to standard-cell and gate-array designs, as is confirmed by the experimental results of Table 1. To make EIG1 more generally applicable, we are investigating netlist transformations which "granularize", e.g., macro-cell designs.

[2] We followed the method in [27], where I/O pads are assumed to have area = 1.

As our current spectral algorithm is oblivious to node weights, it should be well suited to large-scale partitioning applications in CAD, e.g., partition for testability or hardware simulation, where the input is simply a netlist hypergraph with uniform node weights. For example, [29] reports that ratio cut partitioning saved 50% of hardware simulation costs of a 5-million gate circuit as part of the Very Large Scale Simulator Project at Amdahl; similar savings were obtained for test vector costs. With such applications in mind, we compared our method to RCut1.0 on ***unweighted*** netlists, including the MCNC Test02 - Test06 benchmarks and two circuits from Hughes [27]. The RCut1.0 code was obtained from its authors, and was run for ten consecutive trials on each netlist, following the experimental protocol in [27]. Our spectral algorithm output averaged 9% better than the best RCut1.0 outputs (mostly because the Test06 result was not good). The results are summarized in Table 2.

Test problem	Number of elements	Wei-Cheng (RCut1.0)			Hagen-Kahng (EIG1)			R_{HK}/R_{WC} Ratio
		#Mods	Nets cut	Ratio cut	#Mods	Nets cut	Ratio cut	
bm1	882	9:873	1	1.27×10^{-4}	21:861	1	5.5×10^{-5}	.434
19ks	2844	1011:1833	109	5.9×10^{-5}	387:2457	64	6.7×10^{-5}	1.144
Prim1	833	152:681	14	1.35×10^{-4}	150:683	15	1.46×10^{-4}	1.082
Prim2	3014	1132:1882	123	5.8×10^{-5}	725:2289	78	4.7×10^{-5}	.814
Test02	1663	372:1291	95	1.98×10^{-4}	213:1450	60	1.94×10^{-4}	.982
Test03	1607	147:1460	31	1.44×10^{-4}	794:813	61	9.4×10^{-5}	.654
Test04	1515	401:1114	51	1.14×10^{-4}	71:1444	6	5.9×10^{-5}	.512
Test05	2595	1204:1391	110	6.6×10^{-5}	429:2166	57	6.1×10^{-5}	.933
Test06	1752	145:1607	18	7.7×10^{-5}	9:1743	2	1.27×10^{-4}	1.649

Table 2: Comparison on benchmark netlist graphs with uniform node weights. Results reported for RCut1.0 are best of 10 consecutive runs on each input. Again, the EIG1 results do not involve any local improvement of the initial eigenvector partition.

4.2 Clustering

Finally, we outline results showing that a straightforward interpretation of the second eigenvector yields good clusterings on the two classes of "difficult" inputs in the literature.

The first type of input is given by the random graph model $G_{Bui}(2n, d, b)$, developed by Bui et al. [3] in analyzing graph bisection algorithms. Here, the random graphs have $2n$ nodes, are d-regular and have minimum bisection width almost certainly equal to b. We generated random graphs with between 100 and 800 nodes, and with parameters $(2n, d, b)$ as in Bui's experiments (Table I, p. 188 of [3]).[3] In all cases, the second eigenvector immediately yielded the correct clustering. Table 3 gives the sorted eigenvector for a random graph in the class $G_{Bui}(100, 3, 6)$. The expected clustering places

[3] A very slight modification of Bui's construction was made: to avoid self-loops, we superposed d random matchings of the n nodes in a cluster, rather than making a single matching on dn nodes and then condensing into n nodes.

nodes $0-49$ in one half, and $50-99$ in the other. The eigenvector in Table 3 clearly shows this.

Node	Component	Node	Component	Node	Component	Node	Component
0	-0.215306	2	-8.18021E-02	86	1.32846E-02	51	9.56038E-02
7	-0.211889	18	-7.70074E-02	98	2.38492E-02	63	9.86033E-02
6	-0.206269	45	-7.41216E-02	93	3.22976E-02	90	9.93633E-02
42	-0.199676	5	-7.39643E-02	55	4.34522E-02	82	1.00400E-01
28	-0.189419	37	-7.38216E-02	91	4.75056E-02	56	1.00670E-01
30	-0.188609	9	-7.37229E-02	73	5.39718E-02	99	1.02075E-01
23	-0.145966	11	-7.32770E-02	71	5.84973E-02	58	1.04506E-01
8	-0.142736	10	-6.22291E-02	65	5.93850E-02	77	0.105093
3	-0.129631	17	-6.12589E-02	92	6.69472E-02	67	0.105109
15	-0.120541	40	-5.85974E-02	83	6.71358E-02	81	0.107045
39	-0.118946	49	-5.52888E-02	53	6.71782E-02	70	0.108180
38	-0.114429	24	-5.44673E-02	78	6.79049E-02	95	0.108719
27	-0.112974	32	-5.43197E-02	60	7.61767E-02	59	0.109054
46	-0.108921	48	-5.35167E-02	87	7.69490E-02	68	0.109451
19	-0.108893	44	-5.28886E-02	89	7.92344E-02	79	0.109522
12	-0.107840	43	-4.95200E-02	66	8.66147E-02	72	0.109647
35	-0.107710	26	-4.36413E-02	54	8.74412E-02	84	0.110152
33	-0.107241	1	-4.12196E-02	97	8.80222E-02	74	0.111162
31	-9.93970E-02	36	-3.99069E-02	61	8.87784E-02	50	0.112822
4	-9.78949E-02	22	-2.75223E-02	76	8.95560E-02	94	0.117317
16	-9.29926E-02	34	-2.64148E-02	69	9.02742E-02	62	0.122495
29	-9.22521E-02	20	-2.37243E-02	64	9.33758E-02	57	0.125205
14	-9.10469E-02	25	-1.92634E-02	88	9.49801E-02	85	0.132424
13	-8.83968E-02	21	-2.00013E-03	75	9.51491E-02	80	0.138341
41	-8.40547E-02	47	1.02043E-02	96	9.55961E-02	52	0.139806

Table 3: Sorted eigenvector for random graph in $G_{Bui}(100,3,6)$. "Expected" clustering is nodes $0-49$, $50-99$.

The second type of input is given by the $G_{Gar}(n,m,p_{int},p_{ext})$ random model of Garbers et al. [12], which prescribes n clusters of m nodes each, with all edges inside clusters independently present with probability p_{int} and all edges between clusters independently present with probability p_{ext}. We have tested a number of 1000-node examples of such clustered inputs, using the same values (n,m,p_{int},p_{ext}) as in Table 1 of [12]. In all cases, quite accurate clusterings were immediately evident from the eigenvector, with most clusters completely contiguous in the sorted list, and occasionally pairs of clusters being intermingled. Table 4 gives the sorted eigenvector for a smaller random graph, from the class $G_{Gar}(4,25,0.167,0.0032)$. The p_{int} and p_{ext} are of the same order as in the examples from [12], e.g., $p_{int}=O(n^{-\frac{1}{2}})$ (here implying expected degree 4 for each node). The expected clusters contain nodes $0-24$, $25-49$, $50-74$ and $75-99$. Again, the eigenvector in Table 4 strongly reflects this. Because of the random construction, the "correct" clustering may deviate slightly from expectation; in the instance of Table 4, the switch of nodes 47 and 73 may in fact correspond to the "correct" clustering.

Node	Component	Node	Component	Node	Component	Node	Component
19	-0.122405	58	-8.02980E-02	74	-4.69732E-02	88	9.72816E-02
23	-0.120929	53	-7.90911E-02	47	-1.09157E-02	79	0.124942
21	-0.118306	70	-7.86159E-02	36	1.97259E-02	86	0.128686
1	-0.115762	71	-7.78550E-02	45	2.39392E-02	87	0.132406
17	-0.115104	60	-7.68319E-02	44	2.64149E-02	97	0.133227
10	-0.114198	54	-7.55931E-02	48	2.79049E-02	85	0.135566
6	-0.114083	63	-7.47548E-02	49	2.90957E-02	75	0.138987
5	-0.112286	59	-7.44015E-02	42	3.74555E-02	98	0.140964
9	-0.111083	62	-7.41008E-02	35	3.74606E-02	76	0.141252
8	-0.110700	72	-7.38784E-02	34	3.80184E-02	81	0.141296
12	-0.110309	56	-7.30905E-02	28	3.85021E-02	82	0.142007
20	-0.110163	55	-7.13742E-02	32	3.85118E-02	91	0.144033
0	-0.110106	57	-7.12579E-02	38	3.87157E-02	80	0.149802
22	-0.109779	50	-7.03936E-02	33	3.88343E-02	95	0.150069
4	-0.108896	65	-6.99085E-02	30	4.00060E-02	92	0.151593
11	-0.108043	66	-6.96821E-02	26	4.00622E-02	84	0.152478
15	-0.107415	51	-6.86737E-02	31	4.04879E-02	94	0.152605
18	-1.04816E-01	61	-6.81345E-02	46	4.09548E-02	90	0.153824
2	-1.04262E-01	69	-6.77330E-02	41	4.24597E-02	83	0.154241
14	-1.03383E-01	64	-6.76836E-02	29	4.30074E-02	89	0.155534
24	-1.01804E-01	52	-6.75918E-02	27	4.36473E-02	99	0.155569
7	-1.01130E-01	67	-6.72584E-02	39	4.41580E-02	96	0.160441
13	-1.00931E-01	43	-5.86593E-02	37	4.65159E-02	77	0.163642
16	-9.68630E-02	68	-5.10627E-02	40	4.74509E-02	78	0.165712
3	-9.17668E-02	73	-5.05522E-02	25	7.44758E-02	93	0.176925

Table 4: Sorted eigenvector for random graph in $G_{Gar}(4, 25, .167, .0032)$. "Expected" clusters are $0-24$, $25-349$, $50-74$ and $75-99$.

5 Extensions and Future Work

Many research questions are still under investigation.

- **Speedups of the Eigenvector Computation.** We are actively investigating several obvious improvements to the basic approach of EIG1. A promising variant uses a *condensing* strategy first proposed by Bui et al. [3]: we reduce problem size by finding a random maximal matching on the graph, then using the edges of the matching to condense node pairs into single nodes. After solving the condensed problem, which has $|V|/2$ nodes, the condensed nodes can be re-expanded and an iterative improvement stage may follow. Although there is the drawback of yielding a denser input to the eigenvector computation, the sparsifying heuristics mentioned above may be applied so that a net speedup is still obtained. A second variant weakens *convergence criteria* in the Lanczos implementation, reducing the accuracy of the eigenvector calculation; preliminary experiments indicate that for, e.g., the PrimGA2 benchmark, we can speed up our standard Lanczos computation by a factor of between 1.3 and 1.7 without any loss of solution quality. Parallel implementations of the Lanczos on medium- and large-scale vector processors are also of interest.

- **Iterative Improvement Postprocessing** With any of these heuristics, the ratio cuts so obtained may optionally be improved by using standard iterative techniques. As noted above, we are now starting to apply the Fiduccia-Mattheyses partition improvement method to initial partitions generated by heuristic EIG1. This phase of experimentation will determine the quality of eigenvector partitions as initial partitions for F-M style improvement, and possibly shed new light on the nature of the ratio cut cost surface for large problems.

- **Other CAD Applications.** Following Wei and Cheng, we will apply EIG1 and ratio cut partitioning to other CAD applications such as design for testability and the mapping of logic for hardware simulation. The results in Section 4 suggest that for applications where the ratio cut has already been successful, our spectral construction will provide further improvements. Finally, we will also consider extensions to yield multi-way partitioning, e.g., using locally minimum partitions in the sorted eigenvector, as well as to allowing arbitrary node weights in the netlist.

6 Conclusions

We have presented new theoretical analysis showing that a second-eigenvalue construction yields good partitions under the ratio cut metric. In conjunction with sparse-matrix techniques, this leads to demonstrably effective ratio cut algorithms in terms of both CPU cost and solution quality. The overall approach is certainly competitive with the fastest current methods for circuit partitioning, and it scales well with growing problem sizes [14]. On standard-cell and gate-array industry benchmarks, our solution quality was significantly improved over that of [27]. As expected, our method was also effective for ratio cut partitioning of unweighted graphs, e.g., for test and simulation purposes. Because our result is derived from a single deterministic execution of the algorithm, stability is not a problem, and we do not need to take the best of multiple runs as with other methods. Finally, we note that for all practical purposes, the numerical Lanczos computation is perfectly parallelizable, in contrast to traditional partitioning heuristics. The spectral approach to ratio cut partitioning can thus be seen to satisfy virtually all of the desirable traits listed in Section 1 above.

No previous work applies numerical algorithms to ratio cut partitioning, mostly because the mathematical basis of ratio cuts has only recently been developed. However, from a historical perspective it is intriguing that spectral methods have not been more popular for other problem formulations such as bisection or k-partition, despite the early results of Barnes, Donath and Hoffman and the availability of standard packages for matrix computations. We speculate that this is for several reasons. First, progress in numerical methods and progress in VLSI CAD have followed more or less disjoint paths: only recently have the paths met in the sense that large-scale numerical computations have become reasonable tasks on VLSI CAD workstation platforms. Second, early theoretical bounds and empirical performance of spectral methods for bisections were not generally encouraging. By contrast, Theorem One gives a more natural correspondence between graph spectra and the *ratio cut* metric, and our results confirm that second-eigenvector heuristics are indeed better suited to the ratio cut metric. Finally, it has only been with growth in problem complexity that possible scaling weaknesses of iterative approaches have been exposed. In any case, we strongly believe that the spectral approach to partitioning, first developed by Barnes, Donath and Hoffman twenty years ago, merits renewed interest in the context of a number of basic CAD applications.

7 Acknowledgements

We are grateful to Professor C. K. Cheng of UCSD, Dr. Arthur Wei of Cadence Design Systems and Mr. Chingwei Yeh of UCSD and Amdahl for providing RCut1.0 code [29], and to Dr. Horst Simon of NASA Ames Research Center for providing an early version of his Lanczos implementation [21].

References

[1] E. R. Barnes, "An Algorithm for Partitioning the Nodes of a Graph", *SIAM J. Alg. Disc. Meth.* 3(4) (1982), pp. 541-550.

[2] R.B. Boppana, "Eigenvalues and Graph Bisection: An Average-Case Analysis", *IEEE Symp. on Foundations of Computer Science*, 1987, pp. 280-285.

[3] T. N. Bui, S. Chaudhuri, F. T. Leighton and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior", *Combinatorica* 7(2) (1987), pp. 171-191.

[4] H.R. Charney and D.L. Plato, "Efficient Partitioning of Components", *IEEE Design Automation Workshop*, 1968, pp. 16-0 - 16-21.

[5] C.K. Cheng and T.C. Hu "Maximum Concurrent Flow and Minimum Ratio Cut", Technical Report CS88-141, Univ. of California, San Diego, Dec. 1988.

[6] D. Cvetkovic, M. Doob, I. Gutman and A. Torgasev, *Recent Results in the Theory of Graph Spectra*, North-Holland, 1988.

[7] W.E. Donath, "Logic Partitioning", in *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti, eds., Benjamin/Cummings, 1988, pp. 65-86.

[8] W.E. Donath and A.J. Hoffman, "Lower Bounds for the Partitioning of Graphs", *IBM J. Res. Dev.* (1973), pp. 420-425.

[9] C.M Fiduccia and R.M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", *ACM/IEEE Design Automation Conf.*, 1982, pp. 175-181.

[10] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

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

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

[13] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.

[14] G. Golub and C. Van Loan, *Matrix Computations*, Baltimore, Johns Hopkins University Press, 1983.

[15] K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Management Science* 17(1970), pp. 219-229.

[16] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning of Electrical Circuits", *Bell System Technical J.*, Feb. 1970.

[17] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by Simulated Annealing", *Science* 220 (1983), pp.671-680.

[18] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", *IEEE Trans. on Computers* 33(5) (1984), pp. 438-446.

[19] T. Leighton and S. Rao, "An Approximate Max-Flow Min-Cut Theorem for Uniform Multicommodity Flow Problems with Applications to Approximation Algorithms", *IEEE Annual Symp. on Foundations of Computer Science*, 1988, pp. 422-431.

[20] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, 1990.

[21] A. Pothen, H. D. Simon and K. P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs", *SIAM J. Matrix Analysis and its Applications* 11 (1990), pp. 430-452.

[22] L.A. Sanchis, "Multiple-Way Network Partitioning", *IEEE Trans. on Computers* 38 (1989), pp. 62-81.

[23] D.G. Schweikert and B.W. Kernighan, "A Proper Model for the Partitioning of Electrical Circuits", *ACM/IEEE Design Automation Conf.*, 1972.

[24] C. Sechen, Placement and Global Routing of Integrated Circuits Using Simulated Annealing, Ph.D. Thesis, Univ. of California, Berkeley, 1986.

[25] R.S. Tsay and E.S. Kuh, "A Unified Approach to Partitioning and Placement", *Princeton Conf. on Inf. and Comp.*, 1986.

[26] G. Vijayan, "Partitioning Logic on Graph Structures to Minimize Routing Cost", *IEEE Trans. on CAD* 9(12) (1990), pp. 1326-1334.

[27] Y.C. Wei and C.K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning", *IEEE Intl. Conf. on Computer-Aided Design*, 1989, pp. 298-301.

[28] Y.C. Wei and C.K. Cheng, "A Two-Level Two-Way Partitioning Algorithm", *IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 516-519.

[29] Y. C. Wei, "Circuit Partitioning and Its Applications to VLSI Designs", Ph.D. Thesis, UCSD CSE Dept., September 1990.