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

NEW SPECTRAL METHODS FOR RATIO CUT PARTITIONING AND CLUSTERING

L. Hagen
A. Kahng

October 1991
CSD-910073

Submitted June 11, 1991
Accepted October 9, 1991
To appear in IEEE Trans. on CAD

New 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] by an average of 17% for the Primary MCNC benchmarks, while using less computational resources. Effective *clustering* methods are an immediate by-product of the second eigenvector computation. The clustering methods are very successful on all of the "difficult" input classes that have been proposed in the CAD literature, and yield optimal solutions for instances that are pathological for the Kernighan-Lin and simulated annealing approaches. Finally, we discuss statistical analyses of netlist structure and other motivations (e.g., fanout and port limits in cell-based design) for using the very natural *intersection graph* representation of the input as the basis for partitioning. A second new heuristic, EIG1-IG, is proposed, based on spectral ratio-cut partitioning of the netlist intersection graph. EIG1-IG results are an average of 24% better than RCut1.0 results, over the entire MCNC benchmark suite. Furthermore, runtimes are faster than those of EIG1 due to additional sparsity in the eigenvector computation. Overall, we find that the spectral method is appealing for several reasons: (i) it uses global information while iterative methods rely on local information; (ii) modules in some sense make a continuous rather than discrete choice of location; and (iii) 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 by such authors as Donath [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.

Note that signal delays typically decrease as we move downward in the design hierarchy; for example, on-chip communication is faster than inter-chip communication. Therefore, 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 wiring 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]. A large portion of the literature has treated graph partitioning instead of hypergraph partitioning: not only is the formulation simpler, but many algorithms are applicable only to the graph restriction of the partitioning problem. There are several standard transformations from the hypergraph representation of a circuit to a graph representation. Thus, in this section we will discuss graph partitioning, and defer detailed discussion of the hypergraph-to-graph transformation to Section 3 below.

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 minimum 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 $a_{ij} = 1$ if $\{v_i, v_j\} \in E$ and $a_{ij} = 0$ otherwise. If G has weighted edges, then a_{vw} is equal to the weight of $\{v_i, v_j\} \in E$, and by convention $a_{ii} = 0$ for all $i = 1, \ldots, n$. If we let $d(v_i)$ denote the degree of node v_i (i.e., the sum of the weights of all edges incident to v_i), we obtain the $n \times n$ *diagonal matrix* D defined

by $D_{ii} = d(v_i)$. (When no confusion may arise, we may also use d_i to denote $d(v_i)$.) 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 $w \in W$.

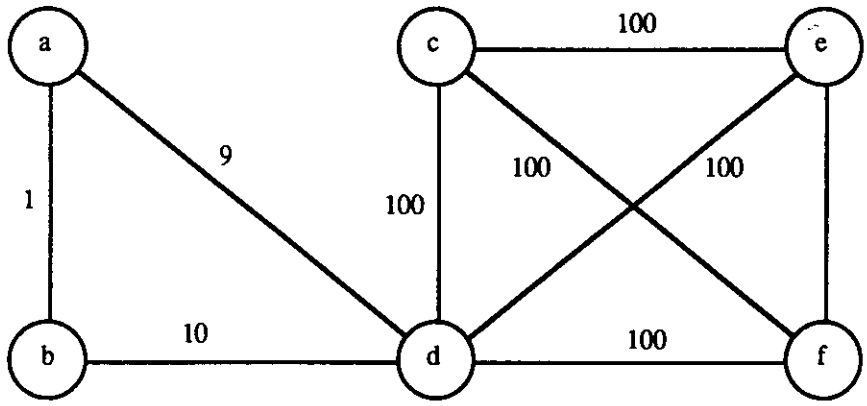

Figure 1: The minimum cut $a|bcdef$ will have cost = 10, but gives a very uneven partition. The optimal bisection $abd|cef$ has cost = 300, much worse than the more natural partitioning $ab|cdef$ which has cost = 19 and is the optimal ratio cut.

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 testability applications, 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 the implementation of 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 shows that the spectral method for minimum ratio cut can be extended to the dual *intersection graph* of the netlist hypergraph. Because of technological limits on fanout and module degree, particularly for cell-based design, the intersection graph is quite sparse and is therefore even better suited to sparse-matrix algorithms than the graph obtained from the netlist hypergraph through the usual clique net model. Results of the EIG1-IG variant, which uses a ratio-cut partition of the intersection graph to construct a module partition, are significantly better than those of EIG1: 24% average improvement over RCut1.0 is obtained over the entire MCNC benchmark suite. Section 6 discusses extensions and directions for future work, and our conclusions are contained in Section 7.

2 A New Connection: Graph Spectra and Ratio Cuts

In this section, we develop the theoretical basis of the method in the context of graph partitioning. Although the discussion concerns graphs rather than hypergraphs, it is straightforward to model the netlist hypergraph as a graph via standard hyperedge models, which we discuss in Section 3 below. The experimental results reported below reflect such transformations of the input hypergraph into a graph representation.

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) $xQx = \sum_{i,j} q_{ij}x_ix_j \geq 0 \ \ \forall x$, and (ii) all eigenvalues of Q are ≥ 0.

- (3) The smallest eigenvalue of Q is 0 with eigenvector $\mathbf{1} = (1, 1, ..., 1)$, since $Q\mathbf{1}_i = (D - A)(1)_i = d_i - \sum_j a_{ij} \cdot 1 = d_i - d_i = 0$. We define λ to be the second smallest eigenvalue of Q. (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, the inner product

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

  which we may simply write as a complete square, i.e., $(x, Qx) = \sum'(x_i - x_j)^2$.

- (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 $p + q = 1$. Now, construct the vector x by letting

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

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 $U|W$ partition, we will have

$$(x, Qx) = \sum_{i,j}{}' \ (x_i - x_j)^2 (from (4)) = \sum_{i,j\in U}{}' \ (x_i - x_j)^2 + \sum_{i,j\in W}{}' \ (x_i - x_j)^2 + \sum_{i\in U, j\in W}{}' \ (x_i - x_j)^2$$

$$= 0 + 0 + \sum_{i \in U, j \in W}{}' (x_i - x_j)^2 = e(U, W).$$

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

We note two important points. First, 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]. Second, 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, a straightforward partitioning method 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.

Input $H = (V, E')$ = 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 [20], where each k-pin net is represented by a complete graph on its k modules, with each edge weight equal to $1/(k-1)$. Thus, the node adjacency matrix A is constructed as follows: For each pair of nodes v_i and v_j with $p \geq 1$ signal nets in common, let $|s_1|, |s_2|, \cdots, |s_p|$ be the number of nodes in the common signal nets $s_1, s_2, \cdots, s_p$, respectively. Each element of A is then given by

$$A_{ij} = \sum_{k=1}^{p} \frac{1}{(|s_k| - 1)}.$$

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. This second class of hyperedge transformation is important because most numerical algorithms will have faster runtimes on sparse input. Preliminary experiments with the sparsifying heuristics, as well as a new *cycle* net model which also yields a sparser Q matrix, have been quite promising. However, we confine the results we report below to those derived using the standard weighted clique model. This is for two main reasons: (i) the weighted clique method already yields a fast and effective algorithm, and (ii) the method is consistent with the usual net modeling practice in VLSI layout [20].

3.2 Numerical Methods

The theoretical results of Section 2 notwithstanding, it at first seems that the computational complexity of the eigenvalue calculation is too great for any practical application. However, significant algorithmic speedups stem from our need to calculate only a *single* (the second-smallest) eigenvalue of a *symmetric* matrix. Furthermore, netlist graphs tend to be very sparse, due to hierarchical circuit organization and degree bounds imposed by the technology fanout limits. This allows us to apply sparse numerical techniques, in particular the block Lanczos algorithm. The field of matrix algorithms is well-studied due to its tremendous importance in a number of applications. For a standard treatment of the symmetric eigenvalue problem, and the Lanczos method in particular, the reader is referred to Golub and Van Loan [14]. (Because tradeoffs between sparsity and runtime are implicit in the class of Lanczos methods, the sparsifying approaches mentioned in Section 3.1 are indeed of interest, since the standard clique model will represent a large k-pin net by $C(k,2)$ nonzeros.)

We use an adaptation of an existing Lanczos implementation [21]. It is important to note that our code actually 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, with the entire software package currently running 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 v_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 of the EIG1 algorithm below, since it subsumes the other three methods and since the cost of evaluating all the partitions is asymptotically dominated by the cost of the Lanczos computation. Although method (iv) requires more effort than the other methods, its evaluation of all $n-1$ partitions is simplified by using data structure techniques similar to those used by Fiduccia and Mattheyses [9], so that cutsize may be quickly updated with each node shift in the partition. Our conversion of the second eigenvector to a heuristic node partition is summarized as shown in Figure 3.

Node Assignment to Partitions
Compute eigenvector v of second eigenvalue $\lambda(Q)$;
Sort entries of v, yielding sorted vector x of node indices;
Place all node indices in partition U;
for $i = 1$ to n
    move node v_i from partition U to partition W;
    calculate and output ratio cut cost for (U, W) partition
Return lowest-cost partition so obtained.

Figure 3: High-level description of conversion from sorted eigenvector to node partition.

With these implementation decisions, our Algorithm EIG1 is summarized as in Figure 4:

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;
Output best ratio cut partition found.

Figure 4: High-level outline of **Algorithm EIG1**.

As we now describe, EIG1 generates *initial* partitions which are already significantly better than the output of the iterative Fiduccia-Mattheyses style algorithm RCut1.0 of Wei and Cheng [27]. In fact, using the single sorted eigenvector we often find *many* partitions that are better than the Fiduccia-Mattheyses result. Therefore, in this paper we do not consider iterative improvement methods, although this puts our current results at some disadvantage. Certainly, post-processing methods are very useful and will be quite appropriate, e.g., in a production implementation.

3.4 Experimental Results: Ratio Cut Partitioning Using EIG1

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]. Note that the results reported in [27] [29] are already an average of 39% better than Fiduccia-Mattheyses output in terms of the ratio cut metric; the experiments run by the authors of [27] compared the best of 10 Rcut1.0 runs to the best of 20 F-M runs, all with random starting seeds.

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.

One readily observes that the current eigenvector computation 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,

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.

then using the actual module areas[1] to determine the best split of the sorted eigenvector under the ratio cut metric. Because module areas are ignored, our approach is somewhat better suited to standard-cell and gate-array designs, as is reflected by the experimental results in Tables 1 and 2. It is possible to make EIG1 more generally applicable via preprocessing netlist transformations which "granularize" building-block designs into more uniformly sized pseudo-modules. This is an open area for future work; preliminary implementations have been promising.

The fact that the EIG1 spectral algorithm is oblivious to node weights is not a difficulty for other large-scale partitioning applications in CAD, e.g., for test or hardware simulation, where the input is simply the 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% improvement over the best RCut1.0 outputs (mostly because the Test06 result was not good). The results are summarized in Table 2.

4 EIG1 Gives Clustering For "Free"

A number of authors have noted that finding natural clusters in the netlist is useful for several applications. Examples include: (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

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

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.

netlist is so large that clustering must be used to reduce the size of the input to a partitioning algorithm. In this section, we make the important observation that clustering is "free" with our approach, in that the second eigenvector v contains both partitioning *and clustering* information. The results below demonstrate that a straightforward interpretation of the sorted second eigenvector can immediately identify natural clusters in the classes of "difficult" partitioning inputs proposed by Bui et al. [3] and Garbers et al. [12]. Such inputs have optimal cutsize that is significantly smaller than the optimal cutsize of a random graph with similar node and edge cardinalities. Furthermore, on these difficult inputs the Kernighan-Lin and simulated annealing algorithms usually return solutions that are an unbounded factor worse than optimal [3]. (For graph bisection, these standard approaches give results that are essentially no better than *random* solutions, an observation which again leads us to question the future effectiveness of iterative techniques as problem sizes become large.) In view of this, it is noteworthy that our approach can easily deal with such "difficult" partitioning instances.

We first consider the class of inputs 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 $(2n, d, b)$ exactly as in Bui's experiments (Table I, p. 188 of [3]).[2] In all cases, the module ordering given by the sorted second eigenvector immediately yielded the expected clustering. Table 3 gives the sorted eigenvector for a random graph in the class $G_{Bui}(100, 3, 6)$. The expected clustering places nodes $0 - 49$ in one half, and $50 - 99$ in the other. The eigenvector in Table 3 clearly shows this.

[2] 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.

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 $n \cdot C(m, 2)$ edges inside clusters independently present with probability p_{int} and all $m^2 \cdot C(n, 2)$ 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. As with the G_{Bui} class of inputs, the Kernighan-Lin and simulated annealing algorithms fail on the G_{Gar} input class, especially as $p_{int} \gg p_{ext}$, but note that this is exactly when the eigenvector method will work well.[3]

[3]Furthermore, for completely pathological inputs where the cut size is zero (i.e., the graph is disconnected), Bui et al. [3] note that Kernighan-Lin and simulated annealing will almost never discover the disconnection. In contrast, our method will immediately yield $\lambda = 0$ by the theorem of Gershgorin mentioned above. (Of course, a disconnected input can also be recognized by simpler means.)

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 $0-24$, $25-49$, $50-74$ and $75-99$.

It is easy to envision more sophisticated interpretations of the sorted second eigenvector. 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 in the sorted eigenvector as delimiters of natural circuit clusters. We have also considered using "local minimum" partitions in the sorted eigenvector, i.e., those with lower ratio cut cost than either of their neighbor partitions, to delineate clusters: in other words, all node indices between consecutive locally minimum ratio cut partitions are placed together into a cluster. This is intuitively reasonable since, as noted above, there are usually many distinct, high-quality (locally minimum) partitions that are derivable as splits of the sorted second eigenvector. Initial experiments show these clustering approaches, which are distinct from previous methods in the literature that utilize multiple eigenvectors, to be very promising. Indeed, we believe that such new heuristics will grow in importance as problem sizes become extremely large and clustering becomes a more critical application.

5 Using the Second Eigenvector of the Intersection Graph

In examining the performance of EIG1 versus iterative ratio cut and bisection algorithms, we noticed several interesting relationships between the size of a net and the probability that the net is cut in

the heuristic (ratio-cut or min-width bisection) circuit partition. These observations led to an alternate application of the spectral construction which significantly improved partition quality for virtually every input that we tested.

5.1 The Intersection Graph

We begin this section with a simple thought experiment: Given a 2-pin net and a 14-pin net in a circuit netlist, which is more likely to be cut in the optimal ratio-cut partition? A simple random model would indicate that it is much less likely for all 14 modules of the larger net to be on a single side of the partition than it is for both modules of the smaller net to be on a single side of the partition. Therefore, we might guess that the 14-pin net is much more likely to be cut, and in fact that the *cut probability* for a k-pin net would be roughly $1 - O(2^{-k})$. This rough relationship has indeed been confirmed for heuristic minimum-width bisections of various small netlists from industry and academia, including, e.g., ILLIAC IV printed circuit boards.

However, our analysis of EIG1 and RCut1.0 output for both minimum-width bisection and minimum ratio-cut metrics has shown that this intuitive model does not necessarily remain correct, particularly as circuit sizes grow large. For example, a typical locally minimum ratio cut for the MCNC Primary2 netlist yields the following statistics, shown in Table 5.

The obvious interpretation of these statistics is that while a random model may suffice for small circuits, larger netlists have strong hierarchical organization, reflecting the high-level functional partitioning imposed by the designer. Thus, nets themselves may very well contain "useful" partitioning information. Furthermore, if we consider the partitioning problem from a slightly different perspective, we realize that the minimum (ratio) cut metric is not only asking for an assignment of *modules* to the two sides of the partition, but is equivalently asking us to assign nets to the two sides of the partition, with the objective being maximization of the number of *nets* that are *not* cut by the partition. In other words, we want to assign the greatest possible number of nets *completely* to one side or the other of the partition. This objective can be captured using the graph dual of the input netlist hypergraph, also known as the *intersection graph* of the hypergraph.

The dualization of the problem is as follows. Given a netlist hypergraph $H = (V, E')$ with $|V| = n$ and $|E'| = m$, we consider the graph $G' = (V'', E'')$ which has $|V''| = m$, i.e., G' has m vertices, one for each hyperedge of H (that is to say, each signal in the netlist). Two vertices of G' are adjacent if and only if the corresponding hyperedges in H have at least one module in common. G' is called

Net Size	Number of Nets	Number Cut
2	1835	21
3	365	29
4	203	18
5	192	26
6	120	5
7	52	12
8	14	0
9	83	5
10	14	1
11	35	0
12	5	0
13	3	0
14	10	0
15	3	0
16	1	0
17	72	22
18	1	1
23	1	0
26	1	1
29	1	0
30	1	0
31	1	0
33	14	4
34	1	0
37	1	0

Table 5: Cut statistics for k-pin nets. Note that the probability of a net being cut in the best heuristic partition does not necessarily increase monotonically with net size, counter to intuition.

the *intersection graph* of the hypergraph H. For any given H, the intersection graph G' is uniquely determined; however, there is no unique reverse construction. An example of the intersection graph is shown in Figure 5.

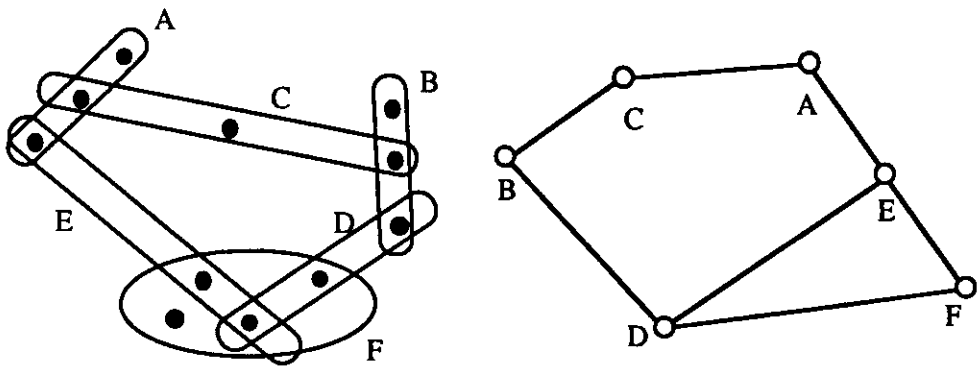

Figure 5: Left: The hypergraph for a netlist with six signal nets (each node represents a module). Right: The intersection graph of the hypergraph (each node represents a signal net).

Given this definition, the adjacency matrix A' of the intersection graph has nonzero elements A'_{ab} exactly when signal nets s_a and s_b share at least one module. As with the earlier mapping of the

netlist hypergraph to a graph formulation via the weighted clique net model (Section 3.1), there are a number of possible heuristic edge weighting methods for the intersection graph. We have tried several approaches. Surprisingly, most of the methods we tried lead to extremely similar, high-quality results, implying that the intersection graph is a very robust, natural representation. The results below are derived through the following construction:

For each pair of signal nets s_a and s_b with $q \geq 1$ nodes $v_1, \ldots, v_q$ in common, let $|s_a|$ and $|s_b|$ be the number of nodes in s_a and s_b respectively. The element A'_{ab} is then given by

$$A'_{ab} = \sum_{k=1}^{q} \frac{1}{d_k} \left(\frac{1}{|s_a|} + \frac{1}{|s_b|} \right)$$

where d_k is again equal to the degree of the k^{th} common node v_k, i.e., the number of nets incident to module k. This net weighting scheme is designed so that overlaps between large nets are accorded somewhat lower significance than overlaps between small nets. The diagonal degree matrix D' is constructed analogously to the matrix D described in Section 1.1 above, with the D'_{jj} entry equal to the sum of the entries in the j^{th} row of A'. Thus, D'_{jj} indicates the total strength of connections between signal net s_j and all other nets which share at least one module with s_j, i.e.,

$$D'_{jj} = \sum_{i=1}^{m} A'_{ij}.$$

Given A' and D', we then find the eigenvector v' corresponding to the second eigenvalue λ' of $Q' = D' - A'$, using the same Lanczos code as in the EIG1 implementation. We may sort the eigenvector to obtain an ordering x' of the *signal net* indices, which we then use to derive a heuristic *node* partition.

The procedure that we adopt is patterned after the method of Section 3.3, but with additional features. Note that it is too simplistic to construct the $(U|W)$ partition by merely assigning signal net s_{x_1} to U, signal net s_{x_m} to W, signal net s_{x_2} to U, etc. This is because such assignments will soon begin to conflict: for example, a net assigned to U will contain some module that also belongs to a net already assigned to W. If we stop the assignment of nodes to U and W exactly when no further nets can be *completely* assigned, then some nodes may remain unattached to either side of the partition. To avoid such difficulties, we adopted the following strategy: assign a node v_i to, e.g., U only when *enough* of the nets containing v_i have been assigned to U. This is accomplished with a heuristic weighting function, where each net exerts "weight" on its component modules inversely proportional to the size of the net. In practice, to guarantee that every node is assigned to a partition, we put all nets/nodes in U, then move the nets one by one to W (beginning with s_{x_1} and continuing through s_{x_m}). A node will move to W only when enough of its total incident *net-weight* w_i (i.e., more than some threshold

proportion) has been shifted to W. (In the pseudocode below, as well as in the reported experiments, we use a threshold of $1/2 \cdot$*net-weight*.) Symmetrically, we also start with all nets/nodes in W, and shift nets beginning with s_{x_m}, since this yields a different set of heuristic node partitions. We then output the best partition among the up to $2 \cdot (n-1)$ distinct heuristic partitions so generated. The conversion of the sorted second eigenvector to a heuristic node partition is summarized in Figure 6:

```
Node Assignment to Partitions – Via Intersection Graph
w ≡ array containing the total net weight of each node
z ≡ array containing the moved net weight of each node
Compute eigenvector v' of second eigenvalue λ(Q');
Sort entries of v', yielding sorted vector x' of net indices;

{* initialize net-weight vector *}
w = 0
for i = 1 to n = number of nodes
    for each signal net s_j containing node v_i
        add 1/|s_j| to w_i

{* begin with all nets/nodes assigned to partition U *}
z = 0
for j = 1 to m = number of nets
    for each node v_i in net s_{x'_j}
        add 1/|s_{x'_j}| to z_i
        if z_i ≥ (w_i/2)
            move node i from partition U to partition W
    calculate and output the ratio cut cost for (U, W) partition

{* begin with all nets/nodes assigned to partition W *}
z = 0
for j = m down to 1
    for each node v_i in net s_{x'_j}
        add 1/|s_{x'_j}| to z_i
        if z_i ≥ (w_i/2)
            move node i from partition W to partition U
    calculate and output the ratio cut cost for (U, W) partition

Output best ratio cut partition found.
```

Figure 6: High-level description of conversion from sorted eigenvector (derived from intersection graph) to node partition.

With these implementation decisions, our algorithm EIG1-IG, based on the second eigenvector of the netlist intersection graph, has the high-level description shown in Figure 7:

Algorithm EIG1-IG
$H = (V, E')$ = input netlist hypergraph;
Compute intersection graph $G'' = (V'', E'')$ of H
Compute A' = weighted adjacency matrix, D' = diagonal matrix of G'';
Compute second-largest eigenvalue of $Q'' = A' - D'$ by block
    Lanczos algorithm (equivalent to $\lambda(Q')$, $Q' = D' - A'$);
Compute associated real eigenvector v';
Sort components of v' to yield vector x' of ordered net indices;
Compute vector $\underline{w}$ of net-weights for nodes in V;
Put all nets and nodes in U; shift nets one by one, moving node v_i
    to W when $\geq w_i/2$ of its net-weight has been shifted;
Put all nets and nodes in W; shift nets one by one, moving node v_i
    to U when $\geq w_i/2$ of its net-weight has been shifted;
Output best ratio cut partition found.

Figure 7: High-level outline of **Algorithm EIG1-IG**.

5.2 Computational Results: EIG1-IG

Table 6 shows computational results obtained using the EIG1-IG algorithm on a number of MCNC benchmarks, as well as the two benchmarks from Hughes tested in [29]. The results show an *average* of over 23% improvement in ratio cut cost over the best results obtained using 10 runs of Rcut1.0.

Test problem	Number of elements	Wei-Cheng (RCut1.0)			Hagen-Kahng (EIG1-IG)			Percent improvement
		Areas	Nets cut	Ratio cut	Areas	Nets cut	Ratio cut	
bm1	882	9:873	1	12.73×10^{-5}	21:861	1	5.53×10^{-5}	57
19ks	2844	1011:1833	109	5.88×10^{-5}	662:2182	92	6.37×10^{-5}	-8
Prim1	833	152:681	14	1.35×10^{-4}	154:679	14	1.34×10^{-4}	1
Prim2	3014	1132:1882	123	5.77×10^{-5}	730:2284	87	5.22×10^{-5}	10
Test02	1663	372:1291	95	1.98×10^{-4}	228:1435	48	1.47×10^{-4}	26
Test03	1607	147:1460	31	14.44×10^{-5}	787:820	64	9.92×10^{-5}	31
Test04	1515	401:1114	51	11.42×10^{-5}	71:1444	6	5.85×10^{-5}	49
Test05	2595	1204:1391	110	6.57×10^{-5}	103:2492	8	3.12×10^{-5}	53
Test06	1752	145:1607	18	7.72×10^{-5}	143:1609	19	8.26×10^{-5}	-7

Table 6: Output from EIG1-IG algorithm: partitioning using sorted second eigenvector of the intersection graph of netlist hypergraph. Results are 24% better on average than those of RCut1.0.

We note that runtimes are significantly faster with the EIG1-IG implementation, since the input to the Lanczos computation is often much sparser than that obtained with the original clique-based hypergraph-to-graph transformation. For example, the Test05 intersection graph results in a Q' matrix that is over *ten times* sparser than the Q matrix created by EIG1 (19935 nonzeros versus 219811 nonzeros). Even though convergence can be slowed by the relative uniformity of the Q'_{ij} entries, the gains

in sparsity result in a net runtime reduction. At the same time, the EIG1-IG results are significantly better than EIG1. Recall that the original Wei and Cheng RCut1.0 algorithm partitions are an average of 39% better than the output of traditional methods; the 24% average improvement that EIG1-IG achieves *over RCut1.0* is thus quite noteworthy. In practice, it is possible to run both EIG1 and EIG1-IG, then choose the better result; however, we believe that a production tool might very well rely on only the EIG1-IG algorithm.

6 Extensions and Future Work

A number of research directions are still open.

- **Speedups of the Eigenvector Computation.** There are several obvious speedups to the basic approaches used in EIG1 and/or EIG1-IG. 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 an overall speedup is still obtained. Sparsifying heuristics can also be used by themselves, e.g., simply thresholding small nonzero elements of Q to zero. A second type of speedup occurs through weakening the *convergence criteria* in the Lanczos implementation, thus reducing the accuracy of the eigenvector calculation; preliminary experiments indicate that for, e.g., the PrimGA2 benchmark, we can speed up our current Lanczos computation by a factor of between 1.3 and 1.7 without any loss of solution quality. Parallel implementations of the Lanczos algorithm on medium- and large-scale vector processors are also of interest, since the algorithm exhibits near-perfect parallel speedup.

- **Iterative Improvement Post-processing** With any of these heuristics, the ratio cuts so obtained may optionally be improved by using standard iterative techniques. As discussed in Section 3 above, we have deferred such post-processing since our initial partitions are already so much better than previous results. However, in practice the Fiduccia-Mattheyses style partition improvement methods should be applied to initial partitions generated by the EIG1 or EIG1-IG heuristics. Such investigations could shed more complete light on the quality of eigenvector partitions as initial partitions for F-M style improvement. In addition, such work might shed new

light on the nature of the ratio cut cost surface for large VLSI partitioning problems.

- **Other CAD Applications.** Following the successes reported by Wei and Cheng [29] [30], EIG1 and EIG1-IG should be applied to ratio cut partitioning for other CAD applications, especially test and the mapping of logic for hardware simulation. The results above certainly suggest that for applications where the ratio cut has already been successful, our spectral construction will provide further improvements. Extensions to multi-way partitioning are relatively straightforward, e.g., using locally minimum partitions in the sorted eigenvector. Finally, devising a net model or a netlist transformation which accounts for arbitrary node weights is an important open issue.

7 Conclusions

We have presented theoretical analysis showing that the second eigenvalue of the matrix $Q = D - A$ yields a new lower bound on the cost of the optimum graph partition under the ratio cut metric. The ratio cut metric has been recently shown to be highly useful in a number of CAD partitioning applications ranging from simulation to physical layout. The very natural derivation of our new lower bound, directly from the definition of the ratio cut metric, suggests that heuristic node partitions can be constructed from the second eigenvector of Q. In conjunction with sparse-matrix (Lanczos) techniques, this leads to new algorithms for ratio cut partitioning which are significantly superior to previous methods in terms of both solution quality and CPU cost. The overall approach is certainly competitive with the fastest current methods for circuit partitioning, and the computational complexity of the Lanczos implementation scales well with increasing problem sizes [14]. On standard-cell and gate-array industry benchmarks, the solutions produced by our algorithm EIG1, which uses a traditional clique net model, were significantly (an average of 17%) better than those of [27]. As expected, our method was also effective for ratio cut partitioning of unweighted graphs, e.g., in test and simulation applications, where the ratio cut metric has been shown to yield tremendous CPU savings [29]. In addition, high-quality circuit *clustering* is a "free" by-product of the second eigenvector construction: indeed, EIG1 gives *optimal* solutions for the difficult input classes of [3] and [12], which are pathological for the Kernighan-Lin and simulated annealing algorithms. Finally, statistical analysis of net cut probabilities as a function of net size, as well as sparsity considerations in the numerical computation, led to algorithm EIG1-IG, which constructs a node partition from the second eigenvector of the netlist *intersection graph*. On the MCNC benchmark test suite, the EIG1-IG algorithm yielded an average of 24% improvement over the previous methods of Wei and Cheng. We note that the EIG1-IG algorithm is generally faster than EIG1, due to technological fanout limits which yield a sparse intersection graph.

Both the EIG1 and EIG1-IG algorithms derive a solution from a single, deterministic execution of the algorithm, i.e., the spectral approach is inherently stable, and we do not need to take the best of multiple runs as with other approaches. Finally, 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 the graph bisection problem 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 well-suited to the ratio cut objective. Finally, it has only been with growth in problem complexity that possible scaling weaknesses of iterative approaches, such as the Kernighan-Lin method, 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.

8 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.

[30] Y. C. Wei and C. K. Cheng, "Ratio Cut Partitioning for Hierarchical Designs" to appear in *IEEE Trans. on CAD*, October 1991.