New Spectral Methods for Ratio Cut Partitioning and Clustering

Lars Hagen, *Student Member, IEEE,* and Andrew B. Kahng, *Member, IEEE*

***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 [29] has received much attention since it naturally captures both min-cut and equipartition, the two traditional goals of partitioning. In this paper, we show that the second smallest eigenvalue of a matrix derived from the netlist gives a provably good approximation of the *optimal* ratio cut partition cost. We also 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 this second eigenvalue. Effective *clustering* methods are an immediate by-product of the second eigenvector computation, and are very successful on the "difficult" input classes proposed in the CAD literature. Finally, we discuss the very natural *intersection graph* representation of the circuit netlist as a basis for partitioning, and propose a heuristic based on spectral ratio cut partitioning of the netlist intersection graph. Our partitioning heuristics were tested on industry benchmark suites, and the results compare favorably with those of Wei and Cheng [29], [32] in terms of both solution quality and runtime. This paper concludes by describing several types of algorithmic speedups and directions for future work.**

I. Preliminaries

As system complexity increases, a divide-and-conquer approach is used to keep the circuit design process tractable. This recursive decomposition of the synthesis problem is reflected in the hierarchical organization of boards, multi-chip modules, integrated circuits, and macro cells. As we move downward in the design hierarchy, signal delays typically decrease; 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.

Any decision made early in the layout synthesis procedure will constrain succeeding decisions, and hence good solutions to the placement, global routing, and detailed routing problems depend on the quality of the partitioning algorithm. As noted by such authors as Donath [7] and Wei and Cheng [32], partitioning is basic to many fundamental CAD problems, including the following:

- *Packaging of designs:* Logic is partitioned into blocks, subject to I/O bounds and constraints on block area; this is the canonical partitioning application at all levels of design, arising whenever technology improves and existing designs must be repackaged onto higher-capacity blocks.
- *Clustering analysis:* In certain 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; predictive models rely on analysis of the partitioning structure of netlists in conjunction with output models for placement and routing algorithms.
- *Hardware simulation and test:* A good partitioning will minimize the number of inter-block signals that must be multiplexed onto a hardware simulator; similarly, reducing the number of inputs to a block often reduces the number of vectors needed to exercise the logic.

A. Basic Partitioning Formulations

A standard mathematical model in VLSI layout associates a graph $G = (V, E)$ with the circuit netlist, where 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 net [25]. A large segment of the literature has treated graph partitioning instead of hypergraph partitioning since not only is the formulation simpler, but many algorithms are applicable only to graph inputs. In this section, we will discuss graph partitioning and defer discussion of standard hypergraph-to-graph transformations to Section III.

Two basic formulations for circuit partitioning are the following:

- *Minimum Cut:* Given $G = (V, E)$, partition V into disjoint U and W such that $e(U, W)$, i.e., the number

Manuscript received June 9, 1991; revised December 12, 1991. This work was supported by the National Science Foundation under Grant MIP-9110696. A. B. Kahng is supported also by a National Science Foundation Young Investigator Award. This paper was recommended by Editor A. Dunlop.

The authors are with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90024-1596.

IEEE Log Number 9200832.

of edges in $\{(u, w) \in E | u \in U \text{ and } w \in W\}$, is minimized.

- *Minimum-Width Bisection:* Given $G = (V, E)$, partition V into disjoint U and W, with $|U| = |W|$, such that $e(U, W)$ is minimized.

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

Because minimum-width bisection divides module area equally, it is a more desirable objective for hierarchical layout. Unfortunately, minimum-width bisection is NP-complete [13], so heuristic methods must be used. Approaches in the literature fall naturally into several classes. Clustering and aggregation algorithms map logic to a prescribed floorplan in a bottom-up fashion using seeded modules and analytic methods for determining circuit clusters (e.g., Vijayan [28] or Garbers *et al.* [12]). The top-down recursive bipartitioning method of such authors as Charney [4], Breuer [7], and Schweikert [25] repeatedly divides the logic until subproblems become small enough for layout.

In production software, iterative improvement is a nearly universal strategy, either as a postprocessing refinement to other methods or as a method in itself. Iterative improvement is based on local perturbation of the current solution and can be either greedy (the Kernighan–Lin method [17], [25] and its algorithmic speedups by Fiduccia and Mattheyses [9] and Krishnamurthy [19]) or hill-climbing (the simulated annealing approach of Kirkpatrick *et al.* [18], Sechen [26], and others). Virtually all implementations use multiple random starting configurations [21], [32] in order to adequately search the solution space and yield some measure of "stability," i.e., predictable performance.

For our purposes, an important class of partitioning approaches consists of "spectral" methods which use eigenvalues and eigenvectors of matrices derived from the netlist graph. Recall that the circuit netlist may be given as the simple undirected graph $G = (V, E)$ with $|V| = n$ vertices $v_1, \cdots, v_n$. Often, we represent G using 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_{ij} is equal to the weight of $(v_i, v_j) \in E$, and by convention $A_{ii} = 0$ for all $i = 1, \cdots, n$. If we let $d(v_i)$ denote the degree of node v_i (i.e., the sum of weights of all edges incident to v_i), we obtain the $n \times n$ diagonal *degree matrix* D defined by $D_{ii} = d(v_i)$. (When no confusion arises, we may also use d_i to denote $d(v_i)$.) The eigenvalues and eigenvectors of such matrices are studied in the subfield of graph theory dealing with *graph spectra* [6].

Early theoretical work by Barnes, Donath, and Hoffman [1], [7], [8] established relationships between the spectral properties and the partitioning properties of graphs. More recently, eigenvector and eigenvalue methods have been used for both module placement in CAD (Frankle and Karp [11] and Tsay and Kuh [27]) and graph minimum-width bisection (Boppana [2]). In the context of layout, these previous works formulate the partitioning problem as the assignment or placement of modules into bounded-size clusters or chip locations. The problem is then transformed into a quadratic optimization, and a Lagrangian formulation leads to an eigenvector computation.

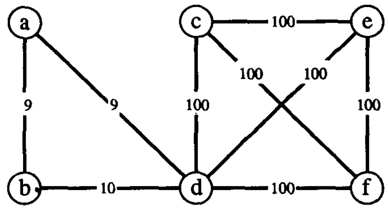


Fig. 1. The minimum cut $a|bcdef$ will have cutsize = 18, but gives a very uneven partition. The optimal bisection $abd|cef$ has cutsize = 300, much worse than the more natural partitioning $ab|cdef$ which has cutsize = 19 and gives the optimal ratio cut.

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 will discuss in Section II. Boppana, Donath and Hoffman, and others use slightly different graph-derived matrices, 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 one-dimensional *quadratic placement* problem of finding the vector $x = (x_1, x_2, \cdots, x_n)$ 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 $|x| = (x^T x)^{1/2} = 1$.

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

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

Taking the first partial derivative of L with respect to x and setting it equal to zero yields

$$2Qx - 2\lambda x = 0$$

which can be rewritten as

$$(Q - \lambda I)x = 0$$

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 x. The minimum eigenvalue 0 gives the uninteresting solution $x = (1/\sqrt{n}, 1/\sqrt{n}, \cdots, 1/\sqrt{n})$, and hence the eigenvector corresponding to the second smallest eigenvalue is used. Hall's work heuristically derived a two-dimensional clustering placement from the eigenvectors of the second and third smallest eigenvalues (i.e., two one-dimensional placements).

B. Ratio Cuts

One easily notices that requiring an exact bisection, rather than, say, allowing up to a 60%–40% imbalance in the module partition, is unnecessarily restrictive. In the instance of Fig. 1, the optimal bisection does not give a very natural partitioning. Penalty functions as in the r-bipartition method of Fiduccia and Mattheyses [9] have been used to permit not-quite-perfect bisections. However, these methods can require rather *ad hoc* thresholds and penalties. With this in mind, the *ratio cut* objective proposed by Wei and Cheng [29], [32], and separately by Leighton and Rao [20], has proved highly successful.

Minimum Ratio Cut: Given $G = (V, E)$, partition V into disjoint U and W such that $e(U, W)/(|U| \cdot |W|)$ is minimized.

The ratio cut metric intuitively allows freedom to find "natural" partitions: the numerator captures the minimum-cut criterion, while the denominator favors an even partition (see Fig. 1). Recent work shows that this metric is extremely useful: on industry benchmarks, [29] reports average cost improvement of 39% over results from a standard Fiduccia–Mattheyses implementation [9]. The ratio cut also has important advantages in other areas of CAD, most notably for test and hardware simulation applications. Wei and Cheng [29], [32] and the recent Ph.D. dissertation of Wei [3] 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, 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 approximations have been proposed [5], [20], but are prohibitively expensive for large problems. Wei and Cheng [29], [32] therefore use an adaptation of the shifting and group swapping techniques of Fiduccia–Mattheyses [9].

In this paper, we show that spectral heuristics are extremely useful for computing good ratio cuts. This conclusion is based on new theoretical results as well as implementation of fast numerical algorithms. Our approach exhibits a number of desired attributes, including the following:

- *Speed:* The low-order complexity is compatible with current CAD methodologies; our method also parallelizes well and allows a tradeoff between solution quality and CPU cost;
- *Stability:* Predictable performance is obtained without resorting to multiple solutions derived from random starting points; and
- *Scaling:* Solution quality is maintained as problem size increases, in contrast to the "error catastrophe" common to local search heuristics for combinatorial problems, particularly on "difficult" instances [3], [21].

The remainder of this paper is organized as follows. In Section II, we show a new theoretical connection between graph spectra and optimal ratio cuts. Section III 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 uses *global* information, giving a qualitatively different result than standard iterative methods which rely on *local* information. Modules in some sense make a continuous, rather than discrete, choice of location within the partition, and only a single numerical computation is required. Section IV gives performance results and comparisons with previous work, using MCNC and other industry benchmarks, as well as classes of "difficult" inputs from the literature. Our method yields significant improvements over the ratio cut partitioning program RCut1.0 of Wei and Cheng [29], [32], and for both partitioning and clustering applications we derive essentially optimal results for the difficult problem classes of [3] and [12]. Section V 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, particularly in cell-based designs, the intersection graph is much better suited to sparse-matrix algorithms than the graph obtained from the netlist through the usual clique net model. Results of our EIG1–IG algorithm are much better than those of EIG1, yielding 24% average improvement over RCut1.0 for the MCNC layout benchmark suite. We conclude in Section VI with extensions and directions for future work.

II. A New Connection: Graph Spectra and Ratio Cuts

In this section, we develop a theoretical basis for our method. 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 is known as the Laplacian of G [34]. Following are several basic properties of Q:

(a) Q is symmetric and non-negative definite, i.e., (i) $x^T Qx = \Sigma_{i,j} Q_{ij} x_i x_j \geq 0 \; \forall x$, and (ii) all eigenvalues of Q are ≥ 0.

(b) The smallest eigenvalue of Q is 0 with eigenvector $\mathbf{1} = (1, 1, \cdots, 1)$.

(c) The inner product $x^T Qx$ corresponds to "squared wire length," i.e.,

$$\begin{aligned} &= x^T Dx - x^T Ax \\ &= \sum_i d_i x_i^2 - \sum_{i,j} A_{ij} x_i x_j \\ &= \sum_i d_i x_i^2 - 2 \sum_{i,j \in E} x_i x_j \end{aligned}$$

which we simply may write as the complete square $x^TQx = \Sigma_{(i,j)\in E}(x_i - x_j)^2$.

(d) Finally, the Courant–Fischer Minimax Principle [34] implies

$$\lambda = \min_{x \perp \mathbf{1}, x \neq \mathbf{0}} \frac{x^TQx}{|x|^2}$$

where λ is the second smallest eigenvalue of Q.

Properties (c) and (d) establish a new relationship between the *optimal* ratio cut cost and the second eigenvalue λ of $Q = D - A$.

Theorem 1: Given a 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 $c \geq (\lambda/n)$.

Proof: Consider the partition which minimizes $e(U, W)/(|U| \cdot |W|)$. We may write $|U| = pn$ and $|W| = qn$, with $p, q \geq 0$ and $p + q = 1$. Construct the vector x by letting

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

Note that x is perpendicular to $\mathbf{1}$, since by construction $x \cdot \mathbf{1} = 0$. Also note that $x_i - x_j = q - (-p) = 1$ for edges e_{ij} crossing the (U, W) partition, but $x_i - x_j = 0$ if e_{ij} does not cross the partition. Property (c) then implies

$$x^TQx = \sum_{(i,j)\in E} (x_i - x_j)^2 = e(U, W).$$

Finally, note that $|x|^2 = q^2pn + p^2qn = pqn(p + q) = pqn = (|U| \cdot |W|)/n$. Since $\min_{x\perp \mathbf{1}}(x^TQx)/|x|^2 = \lambda$ from property (d) above, we have $(x^TQx)/|x|^2 = (e(U, W) \cdot n)/(|U| \cdot |W|) \geq \lambda$, implying

$$e(U, W)/(|U| \cdot |W|) \geq \lambda/n. \qquad \square$$

This is a tighter result than can be obtained using earlier techniques which are essentially based on the Hoffman–Wielandt inequality [1]. If we restrict the partition to be an *exact* bisection, Theorem 1 implies the same bound shown by Boppana [2], but our derivation subsumes that of Boppana.

III. New Heuristics for Ratio Cut Partitioning

The result of Theorem 1 immediately sugests the following partitioning method: compute $\lambda(Q)$ and the corresponding eigenvector x, then use x to construct a heuristic ratio cut.

A basic algorithm template is shown in Fig. 2.

Clearly, a practical implementation of this approach requires closer examination of three main issues: (a) the transformation of the netlist hypergraph into a graph G; (b) the calculation of the second eigenvector x; and (c) the construction of a heuristic ratio cut partition from x. The following subsections address these aspects in greater detail.

Spectral algorithm template

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

Fig. 2. Basic spectral approach for ratio cut partitioning of netlist hypergraph H.

A. Hypergraph Model

Our mapping of hyperedges in the netlist to graph edges in G is based on the standard clique model [21], where each k-pin net contributes a complete subgraph on its k modules, with each edge weight equal to $1/(k - 1)$. In other words, the adjacency matrix A is constructed as follows: For each pair of modules 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 modules in the common signal nets $s_1, s_2, \cdots, s_p$, respectively. Then

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

We have also considered several sparsifying variants, e.g., ignoring less significant (non-critical, large) nets, or thresholding small Q_{ij} to 0 until Q has sufficiently few nonzeros. Such variants are important because most numerical algorithms will have faster runtimes on sparse inputs. Preliminary experiments with these sparsifying heuristics, as well as with a new *cycle* net model which also yields a sparser Q matrix, have been quite promising. However, the results reported below are derived using only the standard weighted clique model, since the clique model is consistent with the usual net modeling practice in VLSI layout [21], and even without sparsifying Q, we obtain a fast and effective algorithm.

B. Numerical Methods

The theoretical results of Section II notwithstanding, it might appear that the computational complexity of the eigenvalue calculation precludes 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 technology constraints. This allows us to apply sparse numerical techniques, in particular, the Lanczos method.

Given an $n \times n$ matrix Q, the Lanczos algorithm iteratively computes a symmetric tridiagonal matrix T whose extremal eigenvalues will be very close to the extremal eigenvalues of Q. So long as only a few extremal eigenvalues are needed, the number of iterations needed to derive T will typically be much less than n. Since T is tridiagonal and symmetric, we can quite rapidly calculate an eigenvalue λ of T and use λ to compute the corresponding eigenvector x of Q. The Lanczos method, which originated in 1950, is well-studied and has been used in a wide variety of applications; for a more detailed exposition, the reader is referred to Golub and Van Loan [14]. Tradeoffs between sparsity and runtime are implicit in the Lanczos approach; thus, the sparsifying approaches mentioned in Section III-A are indeed of interest since the clique model introduces many nonzeros when there are large signal nets in the design.

The results below were obtained using an adaptation of an existing Lanczos implementation [23]. The numerical code is portable Fortran 77; all other code in our system is written in C, with the entire software package running on Sun–4 hardware.

C. Constructing the Ratio Cut

We have considered the following heuristics for constructing the ratio cut module partition from the second eigenvector x:

(a) partition the modules based on sgn (x), i.e., $U = \{\text{module } i: x_i \geq 0\}$ and $W = \{\text{module } i: x_i < 0\}$;
(b) partition the modules around the median x_i value, putting the first half in U and the second half in W;
(c) exploit the heuristic relationship between x and a one-dimensional quadratic placement, whereby a "large" gap in the sorted list of x_i values indicates a natural partition (cf. Section III-D); and
(d) sort the x_i to give a linear ordering of the modules, then determine the splitting index r that yields the best ratio cut.

To describe (d) more precisely: the n components x_i of the eigenvector are sorted, yielding an ordering $v = v_1, \cdots, v_n$ of the modules. The splitting index r, $1 \leq r \leq n - 1$, is then found which gives the best ratio cut cost when modules with index $> r$ are placed in U and modules with index $\leq r$ are placed in W. Because (d) subsumes the other three methods and because its cost is asymptotically dominated by the cost of the Lanczos computation, we use (d) as the basis of the EIG1 algorithm. The evaluation of the $n - 1$ partitions may be simplified by using data structure techniques similar to those of Fiduccia and Mattheyses [9], allowing cutsize to be quickly updated as each successive module is shifted across the partition. Our construction of a heuristic module partition from the second eigenvector is summarized in Fig. 3.

Module assignment to partitions

Compute eigenvector x of second eigenvalue $\lambda(Q)$
Sort entries of x, yielding sorted vector v of module indices
Place all modules in partition U
for $i = 1$ to n
    Move module v_i from partition U to partition W
    Calculate ratio cut cost of (U, W) partition
Output best ratio cut partition found

Fig. 3. Conversion from the sorted eigenvector to a module partition.

D. Experimental Results: Ratio Cut Partitioning Using EIG1

Given the above implementation decisions, our algorithm EIG1 is as summarized in Fig. 4. In this section, we present computational results using the EIG1 algorithm. Since the eigenvector formulation ignores module area information, it is more suited to cell-based and sea-of-gates methodologies. Thus, our initial experiments tested EIG1 on the MCNC Primary1 and Primary2 standard-cell and gate-array benchmarks. Table I compares our results with the best reported results for the RCut1.0 program [29], [31], [32]. Note that the results reported in [29] are already an average of 39% better than Fiduccia–Mattheyses output in terms of the ratio cut metric. Our partitioning results were obtained by applying the Lanczos code, and then using the actual module areas[1] in determining the best split of the sorted eigenvector according to the ratio cut metric.

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 $1/k - 1$
Compute A = adjacency matrix and D = degree matrix of G
Compute second-smallest eigenvalue of $Q = D - A$ by Lanczos algorithm
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

Fig. 4. Outline of Algorithm EIG1.

The fact that the EIG1 spectral approach is oblivious to module weights is not a difficulty for many large-scale partitioning applications in CAD, e.g., test or hardware simulation, where the input is simply the netlist hypergraph with uniform node weights. For example, [31] 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 also compared our method to RCut1.0 on *unweighted* netlists, including the MCNC Test02–Test06 benchmarks and two circuits from Hughes [29], [32]. The RCut1.0 program was obtained from its

[1]We followed the method in [29], where I/O pads are assumed to have area = 1.

TABLE I
COMPARISON WITH BEST VALUES REPORTED FOR THE RCUT1.0 PROGRAM IN [29], [31], AND [32] ON STANDARD-CELL AND GATE-ARRAY BENCHMARK NETLISTS. AREA SUMS MAY VARY, DUE TO ROUNDING. ON AVERAGE, EIG1 RESULTS GIVE 17.6% IMPROVEMENT. NOTE THAT EIG1 RESULTS ARE COMPLETELY UNREFINED; NO LOCAL IMPROVEMENT HAS BEEN PERFORMED ON THE EIGENVECTOR PARTITION

Test Problem	Number of Elements	Wei-Cheng (RCut1.0)			Hagen-Kahng (EIG1)			Percent Improvement
		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}	1
PrimGA2	3014	2488:5885	89	6.08×10^{-6}	2522:5852	78	5.29×10^{-6}	15
PrimSC1	833	1071:1682	35	1.94×10^{-5}	588:2166	15	1.18×10^{-5}	40
PrimSC2	3014	2332:5374	89	7.10×10^{-6}	2361:5345	78	6.18×10^{-6}	14

TABLE II
COMPARISON WITH RCUT1.0 ON BENCHMARK NETLISTS WITH UNIFORM MODULE WEIGHTS. RESULTS REPORTED FOR RCUT1.0 ARE BEST OF TEN CONSECUTIVE RUNS ON EACH INPUT. AGAIN, THE EIG1 RESULTS DO NOT INVOLVE ANY LOCAL IMPROVEMENT OF THE INITIAL EIGENVECTOR PARTITION

Test Problem	Number of Elements	Wei-Cheng (RCut1.0)			Hagen-Kahng (EIG1)			Percent Improvement
		#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}	57
19ks	2844	1011:1833	109	5.9×10^{-5}	387:2457	64	6.7×10^{-5}	−14
Prim1	833	152:681	14	1.35×10^{-4}	150:683	15	1.46×10^{-4}	− 8
Prim2	3014	1132:1882	123	5.8×10^{-5}	725:2289	78	4.7×10^{-5}	19
Test02	1663	372:1291	95	1.98×10^{-4}	213:1450	60	1.94×10^{-4}	2
Test03	1607	147:1460	31	1.44×10^{-4}	794:813	61	9.4×10^{-5}	35
Test04	1515	401:1114	51	1.14×10^{-4}	71:1444	6	5.9×10^{-5}	49
Test05	2595	1204:1391	110	6.6×10^{-5}	429:2166	57	6.1×10^{-5}	7
Test06	1752	145:1607	18	7.7×10^{-5}	9:1743	2	1.27×10^{-4}	−65

authors and run for ten consecutive trials on each netlist, following the experimental protocol in [29]. The EIG1 output averaged 9% improvement over the best RCut1.0 outputs, as summarized in Table II.

The CPU times required by EIG1 are competitive with those of RCut1.0; for example, the eigenvector computation for PrimSC2, using our default convergence tolerance of 10^{-4}, required 83 s of CPU time on a Sun4/60, versus 204 s of CPU time for ten runs of RCut1.0.

As shown by these experimental results, EIG1 generates *initial* partitions which are already significantly better than the output of the iterative Fiduccia–Mattheyses style RCut1.0 program of Wei and Cheng [29]. In fact, using the single sorted eigenvector, we often find many partitions that are better than the RCut1.0 output. Therefore in this paper we do not consider iterative improvement methods, although this puts our results at some disadvantage. Certainly, post-processing improvement would be appropriate in a production implementation.

IV. 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: a) multi-way partitioning, particularly when the sizes of the partitions are not known *a priori*; b) floorplanning or constructive placement; and c) situations when the circuit design is so large that clustering must be used to reduce the size of the partitioning input. This last application is particularly interesting: Bui *et al.* [3] and Lengauer [21] have noted that applying an iterative partitioning algorithm to such a "condensed" netlist, then reexpanding the clusters, yields a better result than if we were to have applied the iterative algorithm directly to the original netlist.

In this section, we show that clustering is "free" with our approach, in that the second eigenvector x contains both partitioning *and clustering* information. This is in line with the original observations of Hall [15], who used eigenvectors of Q to derive a two-dimensional clustering placement. Our results below demonstrate that a straightforward interpretation of the sorted second eigenvector can immediately identify the natural clusters of a graph. In particular, we obtain very good results for the classes of "difficult" partitioning inputs proposed by Bui *et al.* [3] and Garbers *et al.* [12]. For such inputs, which have optimal cutsize significantly smaller than the optimal cutsize of random graphs with similar node and edge cardinalities, the Kernighan–Lin and simulated annealing algorithms return solutions that can be an unbounded factor worse than optimal [3]. Indeed, for graph bisection, these standard approaches give results that are essentially no better than *random* solutions, an observation which again brings into question the continued utility of iterative techniques as problem sizes become large. It is therefore noteworthy that our approach can so easily deal with such "difficult" partitioning instances.

We first consider the class of random inputs $G_{\text{Bui}}(2n, d, b)$, developed by Bui *et al.* [3] in analyzing graph bisection algorithms. Such random graphs have $2n$

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

Fig. 5. Sorted second eigenvector for random graph in $G_{\mathrm{Bui}}(100, 3, 6)$. "Expected" clusters contain modules 0–49 and 50–99.

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

Fig. 6. Sorted second eigenvector for random graph in $G_{\mathrm{Gar}}(4, 25, 0.167, 0.0032)$. "Expected" clusters contain modules 0–24, 25–49, 50–74, and 75–99.

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. Fig. 5 gives the second eigenvector for a random graph in the class $G_{\mathrm{Bui}}(100,3,6)$. The expected clustering places modules 0–49 in one half, and modules 50–99 in the other. The sorted eigenvector clearly reflects this.

The second type of input is given by the random model $G_{\mathrm{Gar}}(n, m, p_{\mathrm{int}}, p_{\mathrm{ext}})$ 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 $(n, m, p_{\mathrm{int}}, p_{\mathrm{ext}})$ as in Table I of [12]. In all cases, quite accurate clusterings were immediately evident from the eigenvector, with most clusters being completely contiguous in the sorted list, and occasional pairs of clusters being intermingled. Fig. 6 depicts the second eigenvector for a smaller random graph, from the class $G_{\mathrm{Gar}}(4, 25, 0.167, 0.0032)$. The p_{int} and p_{ext} values are of the same order as in the examples from [12], with $p_{\mathrm{int}} = \Theta(m^{-1/2})$. The expected clustering groups modules 0–24, 25–49, 50–74, and 75–99. Again, the eigenvector in Fig. 6 strongly reflects this. Note that because of the random construction, the "correct" clustering may deviate slightly from expected clustering, so that the "out of place" entries x_{47} and x_{73} may in fact be optimal. As with the G_{Bui} class, the G_{Gar} inputs are pathological for the Kernighan–Lin and simulated annealing algorithms, especially as $p_{\mathrm{int}} >> p_{\mathrm{ext}}$, but this is exactly where the eigenvector method will succeed.

It is easy to envision other clustering interpretations of the sorted second eigenvector which are quite distinct from previous methods that use multiple eigenvectors. For example, the correspondence between $\lambda(Q)$ and quadratic placement suggests interpreting large *gaps* between adjacent components of the sorted eigenvector as delimiting the natural circuit clusters. We may also use "local minimum" partitions in the sorted eigenvector (those with lower ratio cut cost than either of their neighbor partitions) to delineate clusters. In other words, we cluster modules whose indices lie between consecutive locally minimum ratio cut partitions. This is intuitively reasonable since many distinct splitting indices correspond to high-quality bipartitions. Initial experiments show both of these clustering approaches to be promising. We believe that such heuristics will grow in importance as problem sizes increase and bottom-up clustering becomes a more critical precursor to a well-considered partitioning.

V. 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 given net and the probability that this net is cut by the heuristic (ratio cut or minimum-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.

A. The Intersection Graph

Consider the following 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

[2]A very slight modification of the construction in [3] 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.

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 something like $1 - 2^{-(k-1)}$. This rough relationship has indeed been confirmed for heuristic minimum-width bisections of various small netlists from industry and academia, including ILLIAC IV printed circuit boards.

However, our analysis of EIG1 and RCut1.0 outputs for both the minimum-width bisection and the 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 statistics shown in Table III.

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 reconsider partitioning from a slightly different perspective, we realize that assigning *modules* to the two sides of the partition to minimize the number of net cuts is equivalent to assigning *nets* in order to maximize the number of nets that are *not* cut. 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, 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 $G' = (V', E_{G'})$ 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 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 Fig. 7.

We note that the intersection graph has had only limited previous application in the CAD literature. Pillage and Rohrer [22] applied a "nets-as-points metric" to module placement, the idea being that a heuristic two-dimensional placement of nets could guide module placement. Their scheme attempts to place each module within the convex hull of the locations of nets to which it belongs. An iterative, *ad hoc* solution method is required because the intersection graph is not naturally suited to module placement. For partitioning, Kahng [16] used diameters of the intersection graph to yield an approximate hypergraph bisection heuristic; more recently, Yeh *et al.* [33] proposed to compute cutsize gains from a "net perspective" in an iterative multiway partitioning approach.

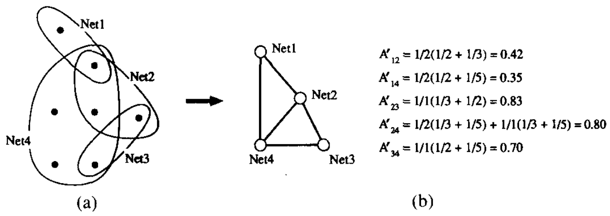


Fig. 7. (a) The hypergraph for a netlist with four signal nets (each node represents a module). (b) The intersection graph of the hypergraph (each node represents a signal net). The intersection graph edge weights A'_{ij} are also shown.

TABLE III
STATISTICS FOR k-PIN NETS CUT IN PRIMARY2 RATIO CUT. NOTE THAT THE PROBABILITY OF A NET BEING CUT IN THE HEURISTIC PARTITION DOES NOT NECESSARILY INCREASE MONOTONICALLY WITH NET SIZE, COUNTER TO INTUITION

Net Size	Number of Nets	Number Cut
2	1836	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

Given the above definition of G', the adjacency matrix $A'(G')$ of the intersection graph has nonzero elements A'_{ab} exactly when signal nets s_a and s_b share at least one module. 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 led to extremely similar, high-quality results: the intersection graph seems to be a very robust, natural representation. The results reported below are derived using the following construction: For each pair of signal nets s_a and s_b with $q \geq 1$ modules $v_1, \cdots, v_q$ in common, let $|s_a|$ and $|s_b|$ be the number of modules in s_a and s_b, respectively. The element A'_{ab} is then given by

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

where d_l is the degree of the l^{th} common module v_l. This 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 I-A above, with the D'_{jj} entry equal to the sum of the entries in the jth 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.

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

Our method is patterned after the method of Section III-C, but has additional features. Note that it is too simplistic to construct the (U, W) partition merely by assigning all the modules of signal net $s_{v'_1}$ to U, all the modules of signal net $s_{v'_m}$ to W, etc. Such assignments will soon conflict, with a net assigned to U containing some module that also belongs to a net already assigned to W. If we stop the module assignment when no more nets can be *completely* assigned, then many modules may remain unattached to either side of the partition. To avoid such difficulties, we assign a module v_i to 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 imposes a "weight" on its component modules inversely proportional to the size of the net. In practice, to guarantee that every module is assigned to a partition, we put all nets/modules in U, then move the nets one by one to W, beginning with $s_{v'_1}$ and continuing through $s_{v'_m}$. A module 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 $(1/2) \cdot w_i$. Symmetrically, we also start with all nets/modules in W, and shift nets beginning with $s_{v'_m}$, since this yields a different set of heuristic module partitions. We then output the best ratio cut partition among the up to $2 \cdot (n - 1)$ distinct heuristic partitions so generated. The conversion of the sorted second eigenvector to a heuristic module partition is summarized in Fig. 8.

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 Fig. 9.

Module assignment to partitions using intersection graph

$w \equiv$ array containing the total net weight of each module
$z \equiv$ array containing the moved net weight of each module
Compute eigenvector x' of second eigenvalue $\lambda(Q')$
Sort entries of x', yielding sorted vector v' of net indices

{* *initialize net-weight vector* *}
$w = \mathbf{0}$
for $i = 1$ to n = number of modules
  for each signal net s_j containing module v_i
    Add $1/|s_j|$ to w_i

{* *begin with all nets/modules assigned to partition* U *}
$z = \mathbf{0}$
for $j = 1$ to m = number of nets
  for each module v_i in net $s_{v'_j}$
    Add $1/|s_{v'_j}|$ to z_i
    if $z_i \geq (w_i/2)$ move module v_i from partition U to partition W
  Calculate ratio cut cost for (U, W) partition

{* *begin with all nets/modules assigned to partition* W *}
$z = \mathbf{0}$
for $j = m$ down to 1
  for each module v_i in net $s_{v'_j}$
    Add $1/|s_{v'_j}|$ to z_i
    if $z_i \geq (w_i/2)$ move module v_i from partition W to partition U
  Calculate ratio cut cost for (U, W) partition
Output best ratio cut partition found

Fig. 8. Conversion from sorted second eigenvector of intersection graph to module partition.

B. Computational Results: EIG1-IG

Table IV 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 [32]. The results show an *average* of over 23% improvement in ratio cut cost over the best results obtained using 10 runs 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 using the original clique-based hypergraph-to-graph transformation. For example, on the MCNC Test05 benchmark, the second eigenvector computation for the intersection graph requires 48 s on a Sun 4/60, versus 619 s for the EIG1 eigenvector computation. This speedup reflects the fact that for the Test05 netlist, the Q' matrix has 19 935 nonzeros, while Q has 219 811 nonzeros. At the same time, the EIG1-IG results are significantly better than those of EIG1. While it is possible to run both EIG1 and EIG1-IG and then choose the better result, we believe that a production tool might rely on the EIG1-IG algorithm alone.

VI. Future Work and Conclusions

There are several obvious speedups to the numerical computations used in EIG1 and EIG1-IG. A promising variant uses the *condensing* strategy proposed by Bui *et*

Algorithm EIG1-IG

Input $H = (V, E')$ = netlist hypergraph
Compute intersection graph $G' = (V', E_{G'})$ of H
Compute A' = adjacency matrix and D' = degree matrix of G'
Compute second smallest eigenvalue $\lambda(Q')$
Compute associated real eigenvector x'
Sort components of x' to yield vector v' of ordered net indices
Compute vector w of net-weights for modules in V
Put all nets and modules in U; shift nets one by one, moving module v_i to W when $\geq w_i/2$ of its net-weight has been shifted
Put all nets and modules in W; shift nets one by one, moving module v_i to U when $\geq w_i/2$ of its net-weight has been shifted
Output best ratio cut partition found

Fig. 9. Outline of Algorithm EIG1-IG.

TABLE IV
OUTPUT FROM THE EIG1-IG ALGORITHM. RESULTS ARE 24% BETTER ON AVERAGE THAN THOSE 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

al. [3] and cited above. Sparsifying heuristics can also be employed, such as simply thresholding small nonzero elements of Q to zero. A second type of speedup occurs through weakening the convergence criteria in the Lanczos implementation. For example, on the PrimGA2 benchmark our experiments indicate that we can speed up our current Lanczos computation by a factor of between 1.3 and 1.7 without any loss of solution quality. Implementations of the Lanczos algorithm on medium- and large-scale vector processors are also of interest, since the algorithm is amenable to parallel speedup.

A second research direction involves post-processing improvement of our current results. As noted in Section III above, we have deferred such post-processing since our initial partitions are already significantly better than previous results. However, in practice Fiduccia–Mattheyses style methods may be applied to initial partitions generated by EIG1 or EIG1-IG.

Finally, following the successes reported by Wei and Cheng [31], [32], 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. Our results above certainly suggest that for applications where the ratio cut has already proved successful, our spectral construction will afford further improvements. Extensions to handle multi-way partitioning are relatively straightforward, e.g., by using locally minimum partitions in the sorted eigenvector. Lastly, we note that devising a netlist transformation which accounts for arbitrary node weights remains an important open issue.

In conclusion, we have presented theoretical analysis showing that the second smallest eigenvalue of the Laplacian $Q = D - A$ yields a new lower bound on the cost of the optimum ratio cut partition. The derivation of the bound suggests that good heuristic partitions can be constructed directly 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, and which scale well with increasing problem size [14]. Our algorithm EIG1 performed an average of 17% better than the RCut1.0 program of [29] on cell-based designs. As expected, EIG1 was also effective for ratio cut partitioning of unweighted graphs, e.g., for test and simulation applications. Moreover, high-quality circuit *clustering* is "free" with the second eigenvector computation. Next, analysis of net cut probabilities as a function of net size led to algorithm EIG1-IG, which constructs a node partition from the second eigenvector of the netlist *intersection graph*. For the MCNC layout benchmark suite, EIG1-IG gave an average of 24% improvement over the previous methods of Wei and Cheng. The EIG1-IG algorithm is also faster than EIG1, due to the sparsity of the 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 does not require multiple runs as with other approaches. The spectral approach thus satisfies virtually all of the desirable traits listed in Section I.

No previous work applies numerical algorithms to ratio cut partitioning, mostly because the mathematical basis of ratio cuts has been developed only recently. 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 due to several reasons. First, progress in numerical methods and progress in VLSI CAD have followed more or less disjoint paths: only recently have the paths converged 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 graph bisection were not generally encouraging. By contrast, Theorem 1 shows 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 been only 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.

Acknowledgment

The authors are grateful to Prof. C. K. Cheng of UCSD and his students, Dr. Arthur Wei of Cadence Design Systems, and Dr. Chingwei Yeh of National Chung-Cheng University, for providing RCut1.0 code [31] and access to benchmarks. They also thank Dr. Horst Simon of NASA Ames Research Center for providing an early version of his Lanczos implementation [23].

References

[1] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM J. Alg. Disc. Meth.*, vol. 7, no. 2, pp. 541-550, Apr. 1982.

[2] R. B. Boppana, "Eigenvalues and Graph Bisection: An Average-Case Analysis," in *Proc. IEEE Symp. Found. Computer Sci.*, 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*, vol. 7, no. 2, pp. 171-191, Feb. 1987.

[4] H. R. Charney and D. L. Plato, "Efficient partitioning of components," in *Proc. 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," Tech. Rep. 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.* New York: North-Holland, 1988.

[7] W. E. Donath, "Logic partitioning," in *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti, Eds. Menlo Park, CA: 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.*, pp. 420-425, 1973.

[9] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. ACM/IEEE Design Automation Conf.*, 1982, pp. 175-181.

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

[11] J. Frankle and R. M. Karp, "Circuit placement and cost bounds by eigenvector decompostion," in *Proc. IEEE Int. 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 Int. 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.* San Francisco, CA: Freeman, 1979.

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

[15] K. M. Hall, "An r-dimensional quadratic placement algorithm," *Manag. Sci.*, vol. 17, pp. 219-229, 1970.

[16] A. B. Kahng, "Fast hypergraph partition," in *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 762-766.

[17] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning of electrical circuits," *Bell Syst. Tech. J.*, Feb. 1970.

[18] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, 1983.

[19] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks," *IEEE Trans. Computers*, vol. 33, pp. 438-446, May 1984.

[20] T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms," in *Proc. IEEE Ann. Symp. Found. Computer Sci.*, 1988, pp. 422-431.

[21] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout.* Chichester, U.K.: Wiley-Teubner, 1990.

[22] L. T. Pillage and R. A. Rohrer, "A quadratic metric with a simple solution scheme for initial placement," in *Proc. ACM/IEEE Design Automation Conf.*, 1988, pp. 324-329.

[23] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Anal. Appl.*, vol. 11, pp. 430-452, 1990.

[24] L. A. Sanchis, "Multiple-way network partitioning," *IEEE Trans. Computers*, vol. 38, pp. 62-81, 1989.

[25] D. G. Schweikert and B. W. Kernighan, "A proper model for the partitioning of electrical circuits," in *Proc. ACM/IEEE Design Automation Conf.*, 1972, pp. 57-62.

[26] C. Sechen, "Placement and global routing of integrated circuits using simulated annealing," Ph.D. dissertation, Univ. of Calfornia, Berkeley, 1986.

[27] R. S. Tsay and E. S. Kuh, "A unified approach to partitioning and placement," *Princeton Conf. Inf. and Comp.*, 1986.

[28] G. Vijayan, "Partitioning logic on graph structures to minimize routing cost," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1326-1334, Dec. 1990.

[29] Y. C. Wei and C. K. Cheng, "Towards efficient hierarchical designs by ratio cut partitioning," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1989, pp. 298-301.

[30] Y. C. Wei and C. K. Cheng, "A two-level two-way partitioning algorithm," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1990, pp. 516-519.

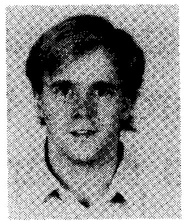
[31] Y. C. Wei, "Circuit partitioning and its applications to VLSI designs," Ph.D. dissertation, Univ. of California, San Diego, Sept. 1990.

[32] Y. C. Wei and C. K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 911–921, July 1991.

[33] C. W. Yeh, C. K. Cheng, and T. T. Lin, "A general purpose multiple way partitioning algorithm," in *Proc. ACM/IEEE Design Automation Conf.*, June 1991, pp. 421–426.

[34] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, Y. Alavi *et al.* Eds., New York: Wiley, 1988, pp. 871–898.

[35] L. Hagen and A. B. Kahng, "Fast spectral methods for ratio cut partitioning and clustering," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Santa Clara, CA, 1991, pp. 10–13.



Lars Hagen (S'91) received the B.S. degree in engineering with option in computer science from California State University, Long Beach, in 1987. He is currently working toward the Ph.D. degree in computer science at the University of California, Los Angeles.

His research interests include partitioning and clustering in VLSI circuit design, and mathematical programming.

Mr. Hagen is a member of ACM SIGDA.

Andrew B. Kahng (A'89), for a photograph and biography please see page 902 of the July 1992 issue.