Test problem	Wei-Cheng RCut1.0	IG-Match	Gain (%)
	L:R:cutsize	L:R:cutsize	
bm1	9:873:1	21:861:1	57
19ks	1011:1833:109	650:2194:85	-1
Prim1	152:681:14	154:679:14	1
Prim2	1132:1882:123	740:2274:77	21
Test02	372:1291:95	211:1452:38	37
Test03	147:1460:31	803:804:58	38
Test04	401:1114:51	73:1442:6	50
Test05	1204:1391:110	105:2490:8	53
Test06	145:1607:18	141:1611:17	3

Table 1: Output from IG-Match algorithm, compared with results from the RCut1.0 program of Wei and Cheng.

5 Conclusions

We have presented a new approach to module partitioning, based on a combination of spectral techniques and the intersection graph representation G' for the circuit hypergraph. Following [12], we use a sparse Lanczos code to induce a linear ordering of *nets* via the sorted second eigenvector of $Q'(G')$. We then formulate the optimal *completion* of the module partition as a maximum independent set computation in a bipartite graph. Our IG-Match algorithm is guaranteed to complete a module partition without cutting more nets than the size of a maximum matching in the bipartite graph; this bound is tight. Furthermore, the computation is efficient in an amortized sense when we test *all* possible partitions ("splits") of the sorted eigenvector to see which leads to the best module partition: IG-Match tests all splits in $O(|V'| * (|V'| + |E|)$ time. Since the computational complexity of the Lanczos implementation scales well with increasing problem sizes [10], we believe that this overall methodology will continue to be useful even when problem sizes grow very large.

A number of interesting open issues remain. The eigenvector computation can be sped up further by additionally sparsifying the input through thresholding, or by relaxation of the numerical convergence criteria. A hybrid algorithm which uses clustering to condense the input before applying the partitioning algorithm (such an approach is discussed by Bui et al. [4] and by Lengauer [18]) is also promising. Parallel speedups of the Lanczos code are also possible. With any of these heuristics, the ratio cuts so obtained may optionally be improved by using standard iterative techniques. Finally, following the successes reported by Wei and Cheng [22] [23], the intersection graph based ratio cut partitioning should be applied to ratio cut partitioning for other CAD applications, particularly test and the mapping of logic for hardware simulation.

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] J. Blanks, "Partitioning by Probability Condensation", *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 758-761.

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

[4] 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.

[5] J. Cong, L. Hagen and A. B. Kahng, "Net Partitions Yield Better Module Partitions", UCLA CS Dept. TR-910075, November 1991.

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

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

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

[9] 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.

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

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

[12] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering", to appear in *IEEE Transactions on CAD*; also available as UCLA CSD TR-910073.

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

[14] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

[15] N. Hasan and C. L. Liu, "Minimum Fault Coverage in Reconfigurable Arrays", *Proc. 18th IEEE Intl. Symp. on Fault-Tolerant Computing Systems*, 1988, pp. 348-353.

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

[17] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", *IEEE Trans. on CAD* 10(3) (1991), pp. 356-365.

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

[19] C. L. Liu, *Introduction to Combinatorial Mathematics*, McGraw-Hill, 1968.

[20] L. T. Pillage and R. A. Rohrer, "A Quadratic Metric with a Simple Solution Scheme for Initial Placement", *Proc. ACM/IEEE Design Automation Conf.*, 1988, pp. 324-329.

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

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

[23] Y. C. Wei and C. K. Cheng, "Ratio Cut Partitioning for Hierarchical Designs", *IEEE Trans. on CAD*, July 1991, pp. 911-921.

[24] C. W. Yeh, C. K. Cheng and T. T. Lin, "A General Purpose Multiple Way Partitioning Algorithm", *Proc. ACM/IEEE Design Automation Conf.*, June 1991, pp. 421-426.

tices do not belong to any one of the sets $Even(L)$, $Even(R)$, $Odd(L)$, or $Odd(R)$. These remaining vertices and their induced bipartite subgraph are denoted as $B' = (L', R', E_{B'})$. Phase II of the IG-Match main loop will consider all *modules* which have not been assigned to a side of the partition via any of the winners determined in Phase I. Essentially, Phase II will put all of these unassigned modules first on one side, then the other, and determine which option yields the better ratio cut cost. Note that this will force all nets in L' to be winners and all nets in R' to be losers, or vice versa.

The IG-Match algorithm is optimal in that the number of nets cut by the module partition will never exceed the size of the maximum matching in B (see [5] for all theorem proofs).

Theorem 4: The set of loser nets output by Algorithm IG-Match is an MVC in B. □

In practice, the number of nets cut by the completed module partition can be less than the size of the MM. This is because a loser net v in $Odd(L)$ may in some instances only have modules in common with nets in $Even(R)$. When Phase II assigns all the modules of nets in $Even(R)$ to the R-side of the partition, the net v will end up with all its modules on the R-side and none of its modules on the L-side, i.e., net v will actually *not* be cut by the partition, even though it is a loser.

With each test of a split of the sorted eigenvector, we move one node from L to R. Since the change in the bipartite graph B is very small, we may retain information between the successive maximum matching computations, as well as between the successive MIS constructions. This allows an efficient implementation of the IG-Match algorithm which has small amortized complexity.

Theorem 6: Given the intersection graph $G' = (V', E_{G'})$ of the netlist hypergraph, The IG-Match algorithm requires $O(|V'| * (|V'| + |E_{G'}|))$ time to complete the module partition for each of the $|V'| - 1$ net partitions derived by splitting the sorted eigenvector v' of $Q'(G')$. □

4 Experimental Results

Table 1 shows computational results for the IG-Match algorithm on benchmark circuits from the MCNC layout test suite, as well as two additional industry circuits analyzed in [23]. The experiments were conducted using uniform module areas as in [12]; thus, the values L and R sum to the total number of modules in each benchmark circuit. For each benchmark, we compare our results with the best result over 10 runs of the RCut1.0 program, following the experimental procedure in [23].[5] Overall, our results are an average of 28.8% better than those of RCut1.0. IG-Match also uniformly dominates results from the precursor intersection graph based algorithm in [12], averaging almost 6% improvement.

The CPU times required by our numerical algorithms are very competitive with those cited in [23] for Fiduccia-Mattheyses optimization: for example, the eigenvector computation for PrimSC2 requires 83 seconds of CPU time on a Sun4/60, versus 204 seconds of CPU for 10 runs of RCut1.0.

[5]The results reported in [23] are already an average of 39% better than Fiduccia-Mattheyses output in terms of the ratio cut metric; in obtaining this assessment, the authors of [23] compared the best of 10 Rcut1.0 runs to the best of 20 F-M runs, all with random starting seeds.

```
IG-Match for $B = (L, R, E_B)$.

$L$, $R$ : sets of net-vertices in left, right partitions
$E_{G'}$ : set of edges in the intersection graph $G'$
$E_B$ : set of edges between $L$ and $R$
$M$ : set of edges in maximum matching between $L$ and $R$
$N$ : set of net-vertices to examine in breadth first search
$W_L$, $W_R$ : sets of winner net-vertices in $L$, $R$ resp.
$V_L$, $V_R$ : sets of modules contained by nets in $W_L$, $W_R$ resp.
$V_N$ : set of modules not contained by nets in $W_L$ or $W_R$

Main Loop Phase I: Selecting Winner Nets
  $v \in L$    /* $v$ is the next net in the sorted IG eigenvector */
  $L := L - \{v\}$
  for all edges $(v, y) \in E_B$ do
      $E_B := E_B - \{(v, y)\}$
  $R := R \cup \{v\}$
  for all edges $(x, v) \in E_{G'}$ do
      if $x \in L$ then
          $E_B := E_B \cup \{(x, v)\}$

  $M$ := a maximum matching of $B$

   /* Construct a maximum independent set */
  $W_L$ := set of unmatched net-vertices of $L$, i.e., $U_L$
  $N := W_L$
  while $N \neq \emptyset$ do
      let $x \in N$
      $N := N - \{x\}$
      for all edges $(x, y) \in E_B$ do
          if $(x', y) \in M$ and $x' \notin W_L$ then
              $W_L := W_L + \{x'\}$
              $N := N + \{x'\}$
  endwhile /* $W_L = Even(L)$ */
  $W_R$ := set of unmatched net-vertices of $R$, i.e., $U_R$
  $N := W_R$
  while $N \neq \emptyset$ do
      let $y \in N$
      $N := N - \{y\}$
      for all edges $(x, y) \in E_B$ do
          if $(x, y') \in M$ and $y' \notin W_R$ then
              $W_L := W_L + \{y'\}$
              $N := N + \{y'\}$
  endwhile /* $W_R = Even(R)$ */


Main Loop Phase II: Module Assignment
  $V_L := \emptyset$
  $V_R := \emptyset$
  for all nets $x \in W_L$ do
      $V_L := V_L \cup \{$ modules in net $x\}$
  for all nets $y \in W_R$ do
      $V_R := V_R \cup \{$ modules in net $y\}$
  $V_N := V - (V_L \cup V_R)$
  calculate ratio-cut with partition $(V_L \cup V_N \mid V_R)$
  calculate ratio-cut with partition $(V_L \mid V_R \cup V_N)$
```

Figure 4: Algorithm IG-Match.

one of the following can be true: (i) l_i has all of its modules on the L-side of the partition, or (ii) r_j has all of its modules on the R-side of the partition. This follows immediately from the fact that l_i and r_j have some module in common. In [16], nets left uncut by the final module partition were called *winners*, and those cut were called *losers*. Adopting this terminology, we see that the min-cut objective is to maximize the number of winner nets, or equivalently, to minimize the number of loser nets.

In formalizing this optimization, the following graph-theoretic terms are useful.

Definition: Given a graph $G = (V, E)$, an *independent set* in G is a subset $V' \subseteq V$ such that no two nodes of V' are connected by an edge. A *maximum independent set* (MIS) is an independent set with largest possible cardinality.

Definition: Given a graph $G = (V, E)$, a *vertex cover* (VC) of G is a subset $V' \subseteq V$ such that for every edge $\{v_i, v_j\} \in E$, either $v_i \in V'$ or $v_j \in V'$. A *minimum vertex cover* (MVC) is a vertex cover with smallest possible cardinality.

Definition: Given a graph $G = (V, E)$, a *matching* in G is a set of k edges in E, no two of which have a vertex in common; we say that k is the *size* of the matching. A *maximum matching* (MM) is a matching with largest possible size.

Using these terms, the problem of maximizing the number of winners (minimizing the number of cut nets) is equivalent to finding a maximum independent set in B. While the MIS problem is NP-complete in general, it is efficiently solved for bipartite graphs. The following two standard results (see, e.g., Theorems 10.1 and 10.2 in [14]) motivate our algorithmic approach.

Theorem 2: For a bipartite graph $B = (L, R, E_B)$ with $|L| + |R| = n$, the sizes of any minimum vertex cover and any maximum independent set sum to n. Moreover, the complement of any MIS will be a MVC, i.e., $(L \cup R) - MIS = MVC$. $\square$

Theorem 3: For a bipartite graph $B = (L, R, E_B)$, the size of a minimum vertex cover of B is equal to the size of a maximum matching in B. $\square$

Given a minimum vertex cover, by Theorem 2 we may simply take its complement to obtain a maximum independent set, and vice versa. (Our plan will be to derive an MIS and make all of the corresponding signal nets winners.) Theorem 3 provides a lower bound for the size of the vertex cover (i.e., the set of loser nets).

Our high-level strategy is as follows. Given the intersection graph $G' = (V', E_{G'})$, we will split the eigenvector-based net ordering v' at some index r. Placing v'_i, $i \leq r$, in L, and v'_j, $j > r$, in R, induces a bipartite subgraph $B = (L, R, E_B)$ of G'. The algorithm will try all splitting indices $r = 1, \ldots, n-1$, where $n = |V|$.

For each bipartite subgraph B, Phase I of the IG-Match main loop (the reader is referred to Figure 4) finds a maximum independent set in B. We first find a maximum matching in B using the standard augmenting-path technique [19] and breadth-first search. The size of the MM gives the size of the MVC, which is the number of cut nets that we hope to attain. From the maximum matching, we construct a maximum independent set in B (i.e., the set of winner nets), as follows. Any unmatched vertex of B is a winner (Figure 3 shows the unmatched vertices on each side of the partition as U_L and U_R). Starting from each vertex in U_L or U_R, we trace alternating paths; none of these paths will be an augmenting path since the matching was maximum. We mark the second, fourth, etc. vertices in each of these alternating paths as losers; the (first,) third, fifth, etc. vertices in each path are marked as winners. We do this because the vertex cover (i.e., the set of losers) must contain at least one vertex from every edge, and in particular, it must contain at least one node from every edge in the set of alternating paths. (If we make nodes of the opposite parity into losers, i.e., the first, third, etc., the resulting vertex cover would not necessarily be minimum.) In Figure 3, winners on paths starting at vertices of U_L are denoted as the set $Even(L)$ since they are at even distance from U_L, and losers on these paths are denoted as $Odd(L)$; $Even(R)$ and $Odd(R)$ are similarly denoted. Note that $U_L \subseteq Even(L)$ and $U_R \subseteq Even(R)$.[4]

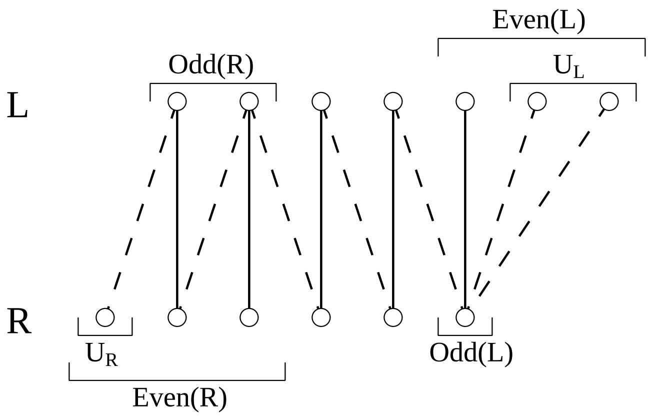

Figure 3: An example bipartite graph showing the set M of matching edges (solid lines), the edges not in the matching (dashed lines), and the sets U_L, U_R, $Even(L)$, $Even(R)$, $Odd(L)$, and $Odd(R)$.

It is possible that after Phase I has been executed, some edges in the matching M remain whose ver-

[4]It is worth noting that the set of loser nets computed during Phase I of the IG-Match main loop is the so-called *critical set* described by Hasan and Liu in [15]. The critical set $C = Odd(L) \cup Odd(R)$ is the unique subset of vertices in B such that every minimum vertex cover of B contains C. Note that even though we start with an arbitrary maximum matching M in B, a result of [15] shows that we will always end up with the same Odd and $Even$ sets, independent of which maximum matching we use.

peredge 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 Figure 1.

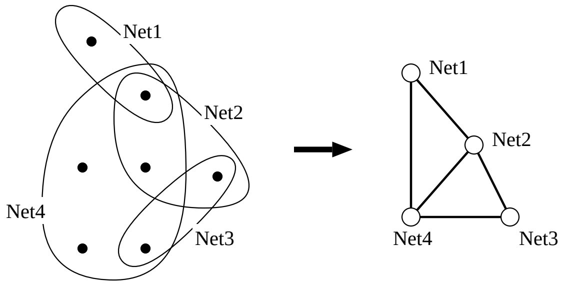

Figure 1: Left: the hypergraph for a netlist with four signal nets (each node represents a module). Right: the intersection graph of the hypergraph (each node represents a signal net).

Given this definition, the adjacency matrix A' of the intersection graph G' will have a nonzero element A'_{ab} when signal nets s_a and s_b share at least one module. As with the usual mapping of the netlist hypergraph to a graph via the weighted clique net model, there are a number of possible heuristic edge weighting methods for the intersection graph. We have tried several approaches, most of which lead to extremely similar, high-quality partitioning results; this indicates that the intersection graph is indeed a robust, natural representation. In the discussion below, we use the following weighting in the intersection graph 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

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

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.[3] The diagonal degree matrix D' is constructed analogously to the matrix D described in Section 1 above, with the D'_{jj} entry equal to the sum of the entries in the j^{th} row of A', i.e., $D'_{jj} = \sum_{i=1}^{m} A'_{ij}$.

Given A' and D', we then find the eigenvector x' corresponding to the second smallest eigenvalue λ' of $Q' = D' - A'$, using the same Lanczos code as in [11] [12]. As described above, the sorted eigenvector yields an ordering v' of the *net* indices, and we use this ordering to derive a heuristic *module* partition. Before presenting our new partitioning algorithm, we note that the intersection graph has had only limited previous application in the CAD literature. Pillage and Rohrer [20] applied the "nets-as-points metric" to module placement, the idea being that a heuristic 2-D placement of nets would establish preferred regions for each module – i.e., a module would wish to lie somewhere within the convex hull of the locations of nets to which it belonged. For partitioning, Kahng [16] used diameters of the intersection graph to yield an approximate hypergraph bisection heuristic; more recently, Yeh et al. [24] proposed to compute gains from a "net perspective" in an iterative multiway partitioning approach. Finally, Hagen and Kahng [12] have recently given a spectral approach which also involves the netlist intersection graph.

3 Module Partitions Based on Net Partitions

Recall that sorting the eigenvector of the second smallest eigenvalue of the netlist intersection graph yields a linear ordering v' for the signal nets of the original netlist. Nets at one end of the sorted eigenvector will usually have very weak connections to nets at the other end, so our basic strategy is to test all possible splitting points of the linear ordering, in order to see which splits might lead to a good module partition. Although such an approach seems computationally expensive, an efficient incremental method can be applied.

Consider what happens when we split the vertices of the intersection graph into two sets L and R. A net $l_i \in L$ might share one or more modules with a net $r_j \in R$. If we draw an edge $\{l_i, r_j\}$ between all pairs of signal nets (l_i, r_j) which are on opposite sides of the split and which have at least one module in common, we induce a bipartite graph $B(L, R, E_B)$ (see Figure 2). Note that according to such results as Theorem 1 [11], use of the eigenvector-based ordering suggests that $|E_B|$, i.e., the number of edges in $\{(l_i, r_j) | l_i \in L, r_j \in R\}$, will be small.

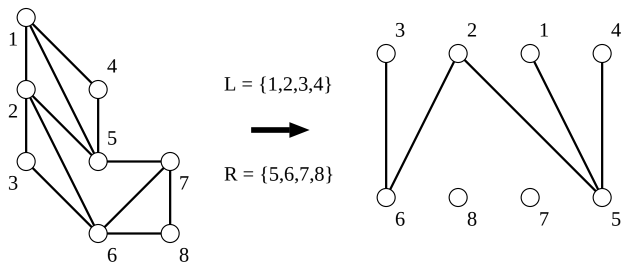

Figure 2: Inducing a bipartite graph from the intersection graph.

If a given edge $\{l_i, r_j\}$ is present in B, the key observation is that for any module partition, at most

[3]This net weighting scheme is designed so that overlaps between large nets are accorded somewhat lower significance than overlaps between small nets.

[2] and Boppana [3]). In general, these previous works formulate the partitioning problem as the assignment or placement of nodes into bounded-size clusters, i.e., chip locations. The problem is then transformed into a quadratic optimization, and Lagrangian relaxation is used to derive an eigenvector formulation [13]. In [11], Hagen and Kahng established a close relationship between the optimal ratio cut cost and the second smallest eigenvalue of the matrix $Q = D - A$, where D and A are as defined above:

Theorem 1 (Hagen-Kahng): 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 $c \geq \frac{\lambda}{n}$. $\square$

This result suggests that the eigenvector x corresponding to λ, i.e., the solution of the matrix equation $Qx = \lambda x$, be used to guide the partitioning. In [11], x was used to induce a linear ordering of the modules, and the best "split" in terms of ratio cut cost was returned. To be more specific, the n components x_i of the eigenvector were sorted, yielding an ordering $v = v_1, \ldots, v_n$ of the modules. The splitting index r, $1 \leq r \leq n-1$, was then found which gave the best ratio cut-cost when modules with index $> r$ were placed in U and modules with index $\leq r$ were placed in W.

The main contribution of the present work follows [12] in highlighting advantages to using the *dual* representation of the logic design, and confirming that *net* structure and interrelationships, rather than *module* adjacencies, should constitute the primary descriptors of a circuit. In particular, the dual *intersection graph* representation of the netlist hypergraph yields much more natural circuit partitioning formulations, since it inherently emphasizes relationships between signal nets. Moreover, the intersection graph yields a sparser circuit representation than traditional net models.[2]

When we use the intersection graph representation of the netlist, we may formulate the minimum-cost module partitioning as a two-stage process. In the first stage, we partition the *nets* of the design. Some modules will belong only to nets on one side of the partition; these modules can be unambiguously assigned to that side. However, other modules may belong to nets on both sides of the partition. Thus, the second stage of the module partitioning involves finding the best *completion* of the net partition, i.e., an assignment of each shared module to one side or the other such that the partition cost is minimized. We propose an efficient algorithm, called IG-Match, for completing the net partition; the algorithm is so named because it is based on a matching computation in a special bipartite graph. IG-Match affords a tight bound on the number of nets cut in completing any given partition; this bound is essentially best-possible. When we move from one net partition to another based on a shift in the splitting index of the sorted eigenvector, the corresponding change in the bipartite graph is very small. Therefore, an incremental strategy is possible, with the computational burden of examining all splits of the eigenvector effectively amortized. Empirical results are very encouraging. The IG-Match method yields significant improvements over the previous ratio-cut partitioning methods: results are an average of 28.8% better than those of Wei and Cheng [22] [23], and this also represents a 6% average improvement over the recent "voting" scheme of Hagen and Kahng [12].

The remainder of this paper is organized as follows. In Section 2, we discuss netlist representations and define the netlist intersection graph. In Section 3, we formulate the problem of completing the module partition from a given net partition as a maximum independent set instance in a bipartite graph, and then present the IG-Match algorithm along with analyses of its performance and complexity. Section 4 gives performance results on benchmarks from MCNC and industry sources; we also give comparisons with the RCut1.0 program of Wei and Cheng [23]. Finally, Section 5 gives conclusions and directions for future work.

2 The Netlist Intersection Graph

The transformation of the netlist hypergraph to a graph is accomplished via a *net model* which determines the A_{ij} values in the weighted graph representation of the design. The most common net model is that of a weighted clique, where a k-pin net will induce $C(k, 2)$ edges among its k modules. Recent work has widely adopted a "standard" weighted clique model [18], wherein a k-pin net contributes $\frac{1}{k-1}$ to each of $C(k, 2)$ A_{ij} values. Advantages of the clique model stem chiefly from its "fairness"; however, the model also has a number of disadvantages. For circuit netlists with large nets the clique model will induce dense adjacency matrices poorly suited for the numerical methods used to calculate the eigenvector.

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 of maximizing 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. A central observation [12] is that such an objective can be captured using the graph dual of the 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 $G' = (V', E_{G'})$ which has $|V'| = m$, i.e., G' has m vertices, one for each hy-

[2] For example, as reported in [12], the intersection graph of the MCNC Test05 benchmark has an adjacency matrix that is over *ten times* sparser than the adjacency matrix created using the standard clique model (19935 nonzeros versus 219811 nonzeros). This allows significant speedup of numerical computations: the eigenvector computation for Test05 using the intersection graph representation requires 48 CPU seconds on a Sun 4/60, versus 619 seconds with the standard clique model.

Net Partitions Yield Better Module Partitions

Jason Cong, Lars Hagen and Andrew Kahng

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

Abstract

In this paper, we demonstrate that the "dual" intersection graph of the netlist strongly captures circuit properties relevant to partitioning. We apply this transformation within an existing testbed that uses an eigenvector computation to derive a linear ordering of nets, rather than modules [12]. We then find a good module partition with respect to the ratio cut metric [23] via a sequence of incremental independent-set computations in bipartite graphs derived from the net ordering. An efficient matching-based algorithm called IG-Match was tested on MCNC benchmark circuits as well as additional industry examples. Results are very encouraging: the algorithm yields an average *of 28.8% improvement over the results of [23]. The intersection graph representation also yields speedups over, e.g., the method of [11], due to additional sparsity in the netlist representation.*[1]

1 Preliminaries

A standard model for 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 net.

Two basic partitioning formulations are:

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

Minimum-width bisection has been a popular objective, particularly within hierarchical approaches. However, the area bisection requirement is unnecessarily restrictive, and various ad hoc thresholds and penalty functions (e.g., the r-bipartition formulation of Fiduccia and Mattheyses [8]) have had varying degrees of success in relaxing this constraint. With this in mind, the recent *ratio cut* metric of Wei and Cheng [23] has proved to be a highly successful objective function for many applications, ranging from layout to hardware simulation. 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. Unfortunately, finding either a minimum-width bisection or a minimum ratio cut is NP-complete, so heuristic methods must be used. Previous approaches to min-width bisection, most of which can also be applied to the minimum ratio cut objective fall into several classes, as surveyed in [6] [12] [18].

In practice, the class of iterative methods are popular either as stand-alone strategies or as a postprocessing refinement to other methods. Iterative methods are based on local perturbation of a current solution and can be greedy or stochastic. Practical implementations will use a number of random starting configurations and return the best result [18] [23] in order to adequately search the solution space and give predictable performance, or "stability". For example, Wei and Cheng [23] use an adaptation of the shifting and group swapping methods in [8], and take the best result of 10 runs, in order to achieve the results cited in Section 4.

With respect to the present work, the most important class of partitioning algorithms consists of "spectral" methods which use eigenvalues and eigenvectors of matrices derived from the netlist graph. Recall that the circuit netlist may be represented by the 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} is equal to the weight of (v_i, v_j) if $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise. 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 degree matrix* D defined by $D_{ii} = d(v_i)$.

Early theoretical work connecting graph spectra and partitioning is due to Barnes, Donath and Hoffman [1] [6] [7]. Most recent eigenvector and eigenvalue methods have dealt with both module placement (Frankle and Karp [9], Kleinhans et al. [17] and Tsay and Kuh [21]) and graph min-cut bisection (Blanks

[1]Research supported in part by NSF grants MIP-9110511 and MIP-9110696.