An Efficient Eigenvector-Node Interchange Approach for Finding Netlist Partitions

Dr. Anthony Vannelli, Dr. Scott W. Hadley and Brian L. Mark

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, CANADA N2L 3G1

ABSTRACT A fast eigenvector technique for obtaining good initial node partitions of netlists for use in interchange heuristics is described. The method is based on approximating the netlist or hypergraph by a weighted graph G and applying the eigenvector technique of Barnes [1] to partition the graph G into k blocks of fixed module size. An efficient generalization of the Fiduccia-Mattheyses node interchange heuristic is developed to further reduce the number of nets connecting k blocks [3]. This node interchange heuristic is tested on the *one* resulting netlist partition obtained by this new eigenvector approach on a variety of small to large sized benchmark netlist partitioning problems (between 200 to 12,000 modules and nets). The test results show that this novel eigenvector-node interchange approach yields netlist partitions that are competitive with the best netlist partitions obtained by using node interchange heuristics alone on many *random initial* netlist partitions. The running time of this method is a small fraction of previous node interchange methods.

1 Introduction

In this paper, we present a fast method for obtaining an initial netlist partition to which any module or node interchange method can be applied. This fundamental VLSI circuit layout problem is solved by a novel method that approximates the netlist by a graph G with *weighted edges*. We then apply the eigenvector-based approach of Barnes [1] to obtain a node partition of G.

The advantage of developing an eigenvector approach to solve the partitioning problem is that the generated initial partitions tend to have many nodes placed in the 'right blocks'. This is due to observation that eigenvector methods are more *global* approaches for solving large-scale VLSI layout problems. Eigenvector approaches have been used to solve many VLSI *placement* problems [2, 4]. On the other hand, node interchange methods are greedy or *local* in nature and get easily trapped in local optima. More important, it has been shown that interchange methods fail to converge to 'optimal' or 'near optimal' partitions unless they initially begin from 'good' partitions [5]. Ideally, one attempts to use an eigenvector approach to place most of the nodes in the correct blocks or node partitions for large-scale partitioning problems. This is the focus of this paper.

The main modeling approach generalizes a technique described in [6] to obtain weights for the edges of a graph G where we consider partitioning the netlist into k **blocks** of **specified** sizes. The edge weights are generated so that every cut of the resulting graph *tightly underestimates* every netlist cut in the original netlist. This translation from a netlist partitioning problem into a graph partitioning problem allows us to use the *powerful eigenvector techniques* for partitioning graphs [1].

2 Approximating a Netlist

It is easy to generate a *hypergraph* from a given netlist by having the node set of the hypergraph correspond to the modules in the netlist, and the generalized edges correspond to the nets. In the remainder of the paper, we will usually discuss the netlist in terms of its equivalent hypergraph.

In this section, we describe how one can approximate a hypergraph H by a graph G with weighted edges. The node set of G is the same as the node set of H, and the edge set of E is obtained by replacing each generalized edge of H by the edge set of a clique which connects the nodes of the generalized edge. In assigning weights to the edges of G we attempt to generate the best graph *underestimation* of the hypergraph H. We say G *underestimates* H if the weight of the edges in G, cut by any partition of the nodes is not greater than the number of generalized edges of H cut by the same partition. In Section 2.1 we discuss how the edge weights can be determined. An example of the graph approximation is given in Section 3.

2.1 Finding Edge Weights

We generate the edge weights by considering the clique obtained by each generalized edge in turn. After considering each generalized edge we obtain a graph containing multiple edges. (If node i and node j are contained in t generalized edges, there will be t (multiple) edges between i and j in the new graph.) We generate G by replacing each set of multiple edges by a single edge whose weight is the sum of the weights of the multiple edges.

If we focus on a particular generalized edge, we wish to assign values such that the weight of any cut in the clique *underestimates* any cut in the generalized edge. One would expect that when considering a single generalized edge all edge weights in the corresponding clique would have the same weight. In fact this is true, as is shown in [6]. Assume for simplicity that each generalized edge (net or wire) has a unit weight and that we wish to partition the nodes (modules) among k blocks. Let a be the value assigned to each edge of the clique representation of a generalized edge connected to c nodes. In order to find a graph fit which underestimates a partition of the netlist, we must have $0 \leq n_i a \leq 1$, where n_i is the number of cut edges in the i^{th} distinguishable partition of the net into k blocks. There are only a finite number of distinguishable partitions of a net connected to c modules. In order to minimize the error in the underestimation of a cut of generalized edge connected to c modules, we must choose the maximum value for a. The maximum possible value for a is

$$\frac{1}{max_i(n_i)}. \tag{1}$$

For example, in the case of a net connected to four modules where these modules are partitioned over two blocks, we see that the maximum number of interconnections arise if we assign two modules in each block. This implies that there are four interconnections between the two blocks. Thus $a = \frac{1}{4}$. The best four node (modules) approximation of the netlist where the four modules are connected to one net is shown in Figure 2. Note that equation (1) allows us to calculate a by finding the *inverse* of the largest number of edges connecting k blocks. Clearly, the largest number of edges connecting the nodes (modules) in these k blocks arises when we *equipartition* the number of nodes between the k blocks; that is, we assign approximately $\frac{n}{k}$ nodes to each block.

Table 1 shows the calculation of the values a for different modules and required blocks. Thus we generate a graph underestimation by considering the *best* underestimation of each generalized edge. The details of this graph approximation technique is described in [6]. We see how this graph can be used to find a partition in the Section 3.

3 Generating Initial Partitions

Given the graph approximation of the hypergraph H, we now find an initial partition of graph G by using the eigenvector-based approach due to Barnes [1]. We will then use this same partition for H. Barnes shows that the graph partitioning problem is equivalent to a matrix approximation problem. We summarize these results below.

Assume that the *approximating* graph, G, we are considering consists of n nodes which are to be partitioned into k disjoint blocks of sizes $m_1 \geq m_2, \geq \cdots \geq m_k$. A partition can be completely specified by a set of k *node assignment vectors* $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_k}$, one corresponding to each block, which have the form

$$\mathbf{x_j} = (x_{1j}, x_{2j}, \cdots, x_{Nj})^T, \; 1 \leq j \leq k,$$

where

$$x_{ij} = \begin{cases} 1 & \text{if node } i \text{ is in block } j \\ 0 & \text{otherwise.} \end{cases}$$

Let v_{ij} be the i^{th} component of the eigenvector corresponding to the j^{th} largest eigenvalue of the adjacency matrix of G. Barnes [1] shows that the solution of the following linear transportation problem gives an approximate solution to the *graph* partitioning problem.

$$Maximize \sum_{i=1}^{n}\sum_{j=1}^{k} \frac{v_{ij}}{\sqrt{m_j}} x_{ij}$$

$$\begin{array}{lll} subject\ to & \sum_{i=1}^{n} x_{ij} = m_j, & j = 1, \cdots, k, \\ & \sum_{j=1}^{k} x_{ij} = 1, & i = 1, \cdots, n, \\ & x_{ij} \geq 0, & i = 1, \cdots, n;\ j = 1, \cdots, k. \end{array} \tag{2}$$

The partition given by the solution of the transportation problem (2) usually places most of the nodes in the correct blocks. This has been empirically verified on many graph partitioning problems [6].

In the *two block* case, the transportation problem (2) can be further simplified by replacing x_{i2} by $1 - x_{i1}$. Making this substitution and letting $x_i = x_{i1}$, the objective function becomes

$$\sum_{i=1}^{n} \left\{ \frac{v_{i1}}{\sqrt{m_1}} - \frac{v_{i2}}{\sqrt{m_2}} \right\} x_i + \sum_{i=1}^{n} \frac{v_{i2}}{\sqrt{m_2}}. \tag{3}$$

The transportation problem (2) reduces to the following $\{0,1\}$-*knapsack* problem

$$\begin{array}{ll} Maximize & \sum_{i=1}^{n} \left\{ \frac{v_{i1}}{\sqrt{m_1}} - \frac{v_{i2}}{\sqrt{m_2}} \right\} x_i \\ subject\ to & \sum_{i=1}^{n} x_i = m_1, \\ & 0 \leq x_i \leq 1, \quad i = 1, \cdots, n. \end{array} \tag{4}$$

It is well known that the solution to (4) is obtained by sorting the objective coefficients in non-increasing order and setting $x_i = 1$ for the first m_1 variables in the sorted list (all other variables are set to zero).

Recall, the weight of any cut in the generated graph G underestimates the number of generalized edges cut in H by the corresponding node partition. We now find the partition using the best graph approximation and the method of Barnes described above for the 5 module - 3 net example shown in Figure 3. Assume that we wish to partition the hypergraph into *two* blocks of nodes, one block containing 3 nodes and the other block containing 2 nodes. The largest two eigenvalues of the adjacency matrix $\mathbf{A}$ (a_{ij} contains the weight of the edge connecting nodes i and j) of this graph and the corresponding eigenvectors are

$$\lambda_1 = 1.7368, \quad \mathbf{v_1} = [.548, .206, .206, .679, .391],$$
$$\lambda_2 = .27755, \quad \mathbf{v_2} = [.223, .535, .535, -.164, -.592].$$

Substituting into equation (3), we find that the coefficients of the objective function are

$$\delta = [.158, -.259, -.259, .508, .645].$$

The partition by sorting the components in δ is

$$S_1 = \{1,4,5\}$$
$$S_2 = \{2,3\}.$$

By inspection we see that at least one generalized edge must be cut by any partition satisfying the size constraints. Since the partition generated by S_1 and S_2 cuts exactly 1 generalized edge we have obtained the *optimal solution*.

4 Test Results

A C code NETPART has been developed on a UNIX environment to incorporate the new partitioning method. We present initial and promising experimental results that are obtained using the eigenvector method and an extension of the node interchange heuristic technique of Fiduccia and Mattheyses [3] to many networks. All computational work was done on a MIPS/2000 computer at the University of Waterloo.

The technique and the resulting computer code was tested on seven netlist partitioning problems listed in Table 2. Chipl-Chip4 are taken from the work of Fiduccia and Mattheyses [3] and Primaryl, Primary2 and Industry2 are taken from the MCNC gate-array test suite benchmarks. These netlists vary in size from 200 to 12,000 nodes and 200 to 13,000 nets.

Table 3 shows the results of applying the iterative improvement (interchange) heuristic of Fiduccia and Mattheyses on the *bi-partitioning* of netlists (networks) [3]. The block sizes of the partitions obtained using the heuristic are almost equal in size. The second and third columns of this table give the results obtained when the partition from the eigenvector approach was used as the initial partition. The remaining columns give the results from using 30 random starting partitions. The fourth and fifth columns give the cutset sizes of the best and worst partitions obtained. Finally, the last column gives the total CPU time (in seconds) spent in executing the 30 runs.

Some general observations can be made from the results presented in Table 3. The final partitions that are generated using the iterative improvement approach of Fiduccia and Mattheyses [3] from the initial partition generated by this new eigenvector approach compare favorably with those obtained using only random starting partitions. The total of the execution times for obtaining a starting partition using the eigenvector approach and then applying iterative improvement is *much less* (**up to 30 times faster**) than the time required to perform the heuristic from 30 random starting points. For the largest netlist that was partitioned -*Industry2*, this new approach yields cuts that are 300% *better* than the cuts found for the partitions generated from random starting partitions only. Also, the quality of partitions gets better as the size of the netlists increases.

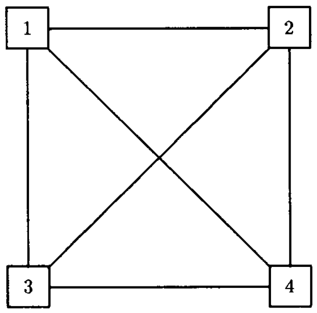

Figure 1: Complete Graph Connecting 4 Modules

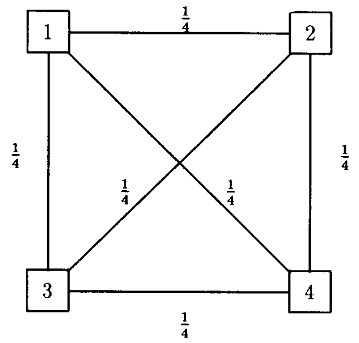

Figure 2: Best 4 Module Graph Approximation

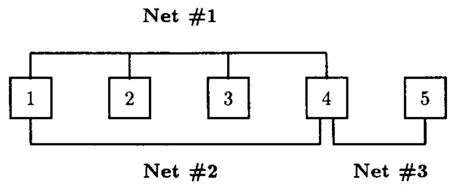

Figure 3: 5 module-3 net Example

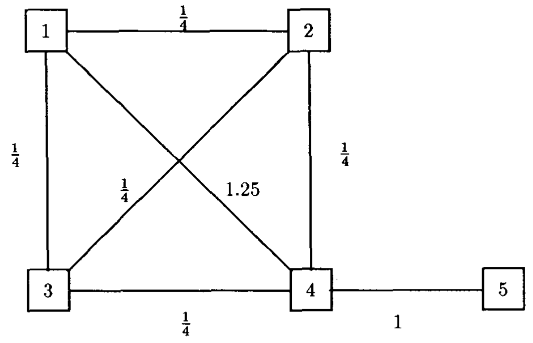

Figure 4: Best 4 Module Graph Approximation

Table 1

Calculation of a for Different Numbers of Blocks (k)

n	$k=2$	$k=3$	$k=4$
3	1/2	1/3	1/3
4	1/4	1/5	1/6
5	1/6	1/8	1/9
6	1/9	1/12	1/13
7	1/12	1/16	1/18

Table 2
Netlist Partitioning Test Cases

Name	Nets	Nodes	Node Degree		Net Size	
			$\bar{x}$	σ	$\bar{x}$	σ
Chip1	294	300	2.82	1.15	2.87	1.39
Chip2	239	274	2.45	1.14	2.81	1.12
Chip3	219	199	2.74	1.28	2.49	1.25
Chip4	221	244	2.34	1.14	2.59	1.01
Primary1	904	833	3.50	1.29	3.22	2.59
Primary2	3029	3014	3.72	1.55	3.70	3.82
Industry2	12949	12142	3.89	1.76	3.64	11.15

Table 3
Two-Block Partitions : After Iterative Improvement

Name	**New Method**		**Random Starts (30)**		
	Cuts	Time (sec.)	Best	Worst	Time (sec.)
Chip1	24	0.36	22	52	15.73
Chip2	17	0.40	15	39	15.64
Chip3	8	0.22	6	30	8.65
Chip4	8	0.37	8	30	10.41
Primary1	81	2.79	97	164	40.45
Primary2	261	8.55	468	623	236.41
Industry2	639	50.93	1844	2078	1350.00

References

[1] E.R. Barnes, "An algorithm for partitioning the nodes of a graph", **SIAM J. on Algebraic and Discrete Methods**, Vol. 3, No.4, pp. 541-550, 1982.

[2] J. P. Blanks, "Near-optimal placement using a quadratic objective function", **Proceedings of the Design Automation Conference**, 1985.

[3] C.M. Fiduccia and R.M. Mattheyses, "A linear-time heuristic for improving network partitions", **Proc. of 19th Design Automation Conference**, pp.175-181, 1982.

[4] K. Hall, "An r-dimensional quadratic placement algorithm", **Management Science**, Vol. 17, pp. 219-229, 1970.

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

[6] A. Vannelli and S.W. Hadley,"A Gomory-Hu cut tree approach for partitioning netlist," **IEEE Trans on Circuits and Systems**, Vol. 37, No. 9, pp. 1133-1139, 1990.