

An Efficient Eigenvector Approach for Finding Netlist Partitions

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

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 , such that the sum of the cut edges in G tightly underestimates the number of cut nets in any netlist partition. An eigenvector technique of Barnes [2] is used to partition the graph G into k blocks of fixed module size. Another feature of this graph underestimation model of the netlist is that it allows us to obtain *lower bounds* on the actual number of cut nets. A multiblock node interchange heuristic of Sanchis [20] 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 300 to 12 000 modules and nets). Test results on the larger netlists show that in most cases this eigenvector–node interchange approach yields netlist partitions with comparable or fewer cut nets than the best netlist partitions obtained by using node interchange heuristics alone on *many random initial* netlist partitions. Moreover, the running time of this method is a small fraction of the previous node interchange methods.

I. INTRODUCTION

THE partitioning of the modules of a netlist arises in many areas, including the laying out of circuits on computer chips and printed circuit boards [4], [11], [19], computer program segmentation [5], [8], [10], and the laying out of machines in advanced manufacturing systems [22], [23]. In each case, we assume that the netlist can be represented by a *hypergraph*, which is a generalization of a graph where a hypergraph generalized edge (net) can connect more than *two* modules. Throughout our discussion, we assume that the netlist or *hypergraph* H , contains n nodes (modules) $V = \{1, 2, \dots, n\}$ and $|E|$ generalized edges (nets). We study the problem of partitioning the n nodes into k disjoint blocks, $V_1, V_2, \dots,$

V_k , of node sizes m_1, m_2, \dots, m_k , respectively, such that the number of hypergraph edges connecting the k blocks is minimized.

Since this problem is known to be NP-complete (even in the simpler case where the hypergraph H is a graph G ([12]), heuristics have been developed to obtain suitable partitions [10], [16], [20]. These heuristics are known as interchange methods and are based on iteratively improving a series of partitions. In general, the quality of the final solution depends on the quality of the initial (starting) partition. In the heuristics described above the initial partition is randomly generated.

In this paper, we present a fast method for obtaining an initial partition to which any interchange method can be applied. The method is based on approximating the netlist or hypergraph by a weighted graph, G , that *tightly underestimates* the number of cut nets in any netlist partition. An eigenvector technique of Barnes [2] is used to partition the resulting graph, G , into k blocks of fixed module size. A novel feature of this graph underestimation model of the netlist is that it allows us to obtain *lower bounds* on the actual number of cut nets. An efficient implementation of the Sanchis node interchange heuristic is developed to further reduce the number of nets connecting k blocks [20].

This node interchange heuristics is tested on the *one* resulting netlist partitioned obtained by this new eigenvector approach on a variety of small to large sized benchmark netlist partitioning problems (between 300 to 12 000 modules and nets). Test results on the larger netlists indicate that in most cases this eigenvector–node interchange approach yields netlist partitions with comparable or fewer cut nets than the best netlist partitions obtained by using node interchange heuristics alone on *many random initial* netlist partitions. Also, the running time of this method is a small fraction of the previous node interchange methods.

The advantage of developing an eigenvector approach to solve the partitioning problem is that the generalized initial partitions tend to have many nodes placed in the “right blocks.” This is due to the observation that eigenvector methods are more *global* for solving large-scale optimization problems. For example, eigenvector approaches have been used to solve many VLSI *placement* problems [3], [14].

On the other hand, node interchange methods are greedy or *local* in nature and get easily trapped in local optima.

Manuscript received January 7, 1991; revised June 16, 1991. This work was supported by an operating grant (OGP 0044456) and a Microelectronics Research Fund grant from the Natural Sciences and Engineering Research Council of Canada. This paper was recommended by Associate Editor M. Marek-Sadowska.

S. W. Hadley was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ont., Canada. He is now with the Department of Mathematics and Systems Engineering, Shell Research B.V., Amsterdam, The Netherlands.

B. L. Mark was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ont., Canada. He is now with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08540.

A. Vannelli is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ont., Canada N2L 3G1.

IEEE Log Number 9107118.

More important, it has been shown that interchange methods may not converge to "optimal" or "near optimal" partitions unless they initially begin from "good" partitions [13], [17]. We use an eigenvector approach to attempt to place most of the nodes in the correct blocks or node partitions for large-scale partitioning problems. An efficient node interchange approach is developed to move nodes that reduce the number of generalized hypergraph edges that connect modules in at least two blocks.

This paper is divided into five sections. Section II discusses how a netlist H can be estimated by a weighted graph, G . The edge weights of the graph G are determined by characterizing the solution of a related ℓ_p ($p \geq 1$) minimization problem. In Section III, we discuss how the graph G can be used to obtain initial node partitions using the eigenvector based approach of Barnes [2]. Lower bounds on the number of cut nets are obtained using the tightly underestimated graph approximation, G , of H and the simple eigenanalysis results from Donath and Hoffman [9]. The testing of this new eigenvector-based model and the efficient multiblock interchange method of Sanchez [20] is conducted on a variety of well-known benchmark test problems from VLSI circuit layout design and is presented in Section IV. Conclusions and directions for future research are given in Section V.

II. APPROXIMATING A NETLIST BY A GRAPH

The *netlist partitioning* problem can easily be described as a *hypergraph partitioning* problem. In general, hypergraph partitioning problems are NP-hard, even in the simple case where the hypergraph is a graph [12]. However, there are cases where good heuristics exist for partitioning problems on graphs but not for the corresponding hypergraph partitioning problem. The problem of partitioning the modules of a netlist into blocks of specified size such that the number of nets with modules in more than one block is minimized falls into this category. In this case a good heuristic based on the eigenvalues and eigenvectors of the adjacency matrix of the graph has been given by Barnes [2]. As such, we are interested in finding ways to approximate a hypergraph by a graph. A method for approximating a netlist H by a graph, G , with weighted edges is given in [25], where we consider partitioning the netlist into *two blocks* only with *no block size* constraint. The node set of G is the same as the node set of H . The edge set of G is obtained by replacing each net of H by the edge set of a clique containing the modules of the net. Edge weights are assigned so that G provides an *underestimation* of H . We say that G underestimates H if the weight of the edges of G that are *cut* by any module partition is not greater than the number of nets cut by the same partition. An underestimation can prove to be useful when either a lower bound or the optimal solution for the graph partitioning problem can be found. The reason the underestimation is useful is that any lower bound for the graph partitioning problem will also provide a lower bound for the netlist partitioning problem. If a general

estimation is used, bounding results from graph partitioning cannot be exploited.

In this section, we extend the technique described in [25] to obtain weights for the edges of G when we consider partitioning the netlist into k blocks of specified sizes. In subsection A, we find a characterization of the weights on the approximating graph of the netlist. We provide an illustrative example using a netlist with five modules and three nets at the end of the section.

A. Determining 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.

By focusing on a particular generalized edge (net), we wish to assign values such that the weight of any cut in the clique *underestimates* any cut in the generalized edge. We show that all edge weights in the corresponding clique will have the same weight when we consider a single generalized edge (see Theorem 1).

Assume for simplicity that each generalized edge (net) 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. In order to find a graph fit that underestimates a partition of the netlist, we must have $|\varphi(\ell)| a \leq 1$, where $|\varphi(\ell)|$ is the number of cut edges in the ℓ th distinguishable partition of the net into k blocks. There are only a finite number of distinguishable partitions of the modules of a net. In order to minimize the error in the underestimation of a cut of generalized edge, we must choose the maximum value for a . The maximum possible value for a is

$$\frac{1}{\max_{\ell} (|\Sigma(\ell)|)} \quad (1)$$

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 arises if we assign two modules in each block. This implies that there are four interconnections between the two blocks. Thus $a = 1/4$. The best four-node (modules) approximation of the netlist where the four modules are connected to one net is shown in Fig. 1.

B. Theoretical Derivation of Optimal Edge Weights

In this subsection, we expand on the above observations by giving a proof that the intuitive edge weighting (1) is in fact optimal in the underestimation case. In general, to edge (i, j) in G we assign weight a_{ij} . Given a partition of the nodes, we want the edge weights to satisfy the property that, if the generalized edge is cut by a par-

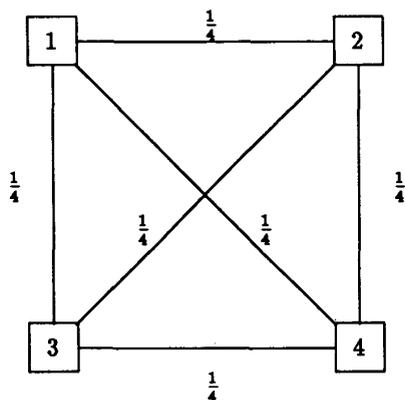


Fig. 1. Best four-module graph approximation.

tion, then the sum of the edge weights of the cut edges in G' is less than or equal to 1 (i.e., the value underestimates the number of cut edges in the hypergraph).

We employ the following notations and definitions when finding the best underestimation edge weighting for these graphs.

Definition 1: A k -partition of the nodes V of the complete graph K_n is a partition of $V = \{V_1, V_2, \dots, V_k\}$ such that $V_i \cap V_j = \emptyset \forall i \neq j$, and $\cup_{i=1}^k V_i = V$. Let P be the set of all k -partitions of the nodes of a graph.

Definition 2: An *equal partition* of the node set V is a k -partition such that $||V_i| - |V_j|| \leq 1 \forall i, j$. We let \mathcal{E} be the set of all equal partitions of the node set of a graph.

We can now give a system of equations describing the *underestimation* property described above:

$$\sum_{(i,j) \in \varphi(\ell)} a_{ij} + s_\ell = 1 \tag{2}$$

$$s_\ell \geq 0 \tag{3}$$

$$|\varphi(\ell)| = \binom{n}{k} - \binom{\lfloor \frac{n}{k} \rfloor}{2} \left(k(1 + \lfloor \frac{n}{k} \rfloor) - n \right) \binom{\lceil \frac{n}{k} \rceil}{2} \left(n - k \lfloor \frac{n}{k} \rfloor \right).$$

for all $\ell \in P$. We can now formulate the problem of finding a *best* (underestimation) edge weighting a mathematical programming problem. There are various measures for determining a *best* edge weighting. Likely candidates are the l_p norms for $p = 1, 2, \infty$. We let S_p denote the mathematical programming problem

$$S_p = \min \{l_p(a, s) \mid (a, s) \text{ satisfies (2), (3)}\},$$

where $l_p(a, s)$ is the l_p norm of the elements of s .

Lemma 1: If $\ell = \{V_1, V_2, \dots, V_k\}$ is a k -partition of the nodes of the complete graph K_n that maximizes the number of edges in $\varphi(\ell)$ over all k -partitions, then ℓ is an equal partition (i.e., $\ell \in \mathcal{E}$).

Proof: We show this result by contradiction. It is

easy to verify that the number of edges cut by $\ell \in P$ is

$$|\varphi(\ell)| = \binom{n}{2} - \sum_{i=1}^k \binom{|V_i|}{2}.$$

Without loss of generality, assume $|V_i| = |V_2| + t$ for some $t \geq 2$, (i.e., $\ell \notin \mathcal{E}$). Consider the k -partition $\hat{\ell} = \{\hat{V}_1, \hat{V}_2, \dots, \hat{V}_k\}$, where \hat{V}_1 and \hat{V}_2 are obtained by taking one node from V_1 and placing it in V_2 , giving \hat{V}_1 and \hat{V}_2 , respectively, with $\hat{V}_i = V_i$ for $i \geq 3$. By expansion and comparison it follows that

$$\begin{aligned} |\varphi(\hat{\ell})| &= |\varphi(\ell)| - (1 - t) \\ &> |\varphi(\ell)|. \end{aligned}$$

This is a contradiction. \square

Lemma 2: Assume there exists an optimal solution to S_p , $p \geq 1$ with $a_{ij} = a$ for all i, j ; then there exists an optimal solution where

$$a_{ij}^{\text{opt}} = a^{\text{opt}} = |\varphi(\ell)|^{-1} \tag{4}$$

where $\ell \in \mathcal{E}$.

Proof: Given any feasible solution where $a_{ij} = a \forall i, j$, we begin by noticing that for all $\ell \in P$, s_ℓ decreases as a increases. Therefore, we are interested in finding the largest value of a (implying smallest error for each partition) that yields a feasible solution. Since there exists a feasible solution with $a = 0$, it follows that $a^{\text{opt}} \geq 0$.

Since

$$s_\ell \geq 0, a^{\text{opt}} \geq 0, \text{ and } |\varphi(\ell)| \geq 0,$$

it follows that

$$a^{\text{opt}} \leq |\varphi(\ell)|^{-1}, \forall \ell \in P.$$

So, the largest value of a occurs when $|\varphi(\ell)|^{-1}$ is minimized over all $\ell \in P$, i.e., $|\varphi(\ell)|$ is maximized. From Lemma 1, we know that this occurs when ℓ is an *equal* k -partition. Thus, $a^{\text{opt}} = |\varphi(\ell)|^{-1}$ for $\ell \in \mathcal{E}$. \square

Lemma 3: For $\ell \in \mathcal{E}$,

Proof: Since $\ell \in \mathcal{E}$, there are $(n - k \lfloor n/k \rfloor)$ subsets of size $\lceil n/k \rceil$ and the remaining $k(1 + \lfloor n/k \rfloor) - n$ subsets are of size $\lfloor n/k \rfloor$. The number of edges in a subset of size t is $\binom{t}{2}$. The result follows. \square

Theorem 1: There exists an optimal solution to S_p , $p \geq 1$ with

$$a_{ij}^{\text{opt}} = |\varphi(\ell)|^{-1} \forall i, j \tag{5}$$

for any $\ell \in \mathcal{E}$.

Proof: First we define equivalence classes of all feasible partitions. Let F_d denote the equivalence class of partitions that yields the same partition after permuting the graph nodes. Now, let $a^* = (a_{ij}^*)$ denote an optimal edge weighting, let $s^* = (s_\ell^*)$ be the corresponding errors, and let err_d denote the sum of the errors for all partitions

TABLE I
CALCULATION OF a FOR DIFFERENT NUMBERS OF
BLOCKS (k)

n	$k = 2$	$k = 3$	$k = 4$
2	1	1	1
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

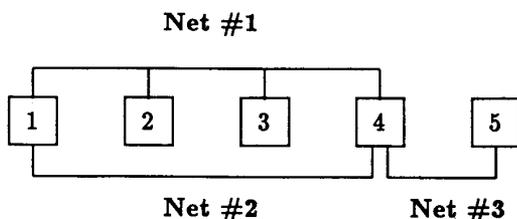


Fig. 2. Five-module, three-net example.

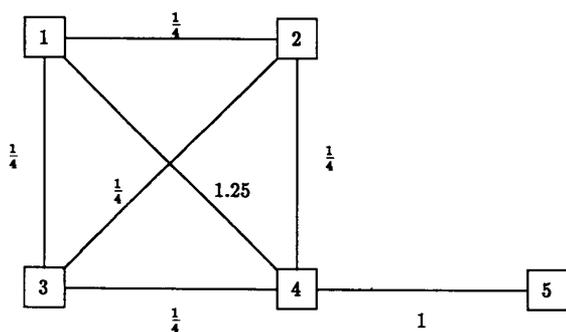


Fig. 3. Best four-module graph approximation.

in equivalence class d ; i.e.,

$$\text{err}_d = \sum_{t \in F_d} s_t^*. \quad (6)$$

Since the ℓ_p norm ($p \geq 1$ and finite) of the errors is convex in the feasible region (i.e., underestimations), it is easy to see that the minimum of the errors subject to (6) for all d occurs when all errors s_t^* are equal for each equivalence class d . By symmetry of the complete graph it follows that this equality holds when all edge weights are equal. It follows that there exists an optimal solution with all edge weights equal. Applying Lemma 2 the theorem follows for finite p . For the ℓ_∞ case a similar argument based on symmetry can be used. \square

Table I shows the calculation of the values a_{ij}^{opt} (using Theorem 1 and Lemma 3) for different modules and required blocks.

We now illustrate this technique by means of a small five-module, three-net example and obtain the best graph underestimation. The hypergraph corresponding to the netlist is shown in Fig. 2, and the best graph underestimation is shown in Fig. 3.

Each edge in the clique on modules $\{1, 2, 3, 4\}$ will

get weights of $1/4$, the edges of cliques $\{1, 4\}$ and $\{4, 5\}$ will get weights of 1. Notice that modules 1 and 4 occur in two nets together: Net #1 with four modules and Net #2 with two modules. Therefore the weight of edge $(1, 4)$ in G has weight $1.25 = 1 + 1/4$.

III. GENERATING INITIAL PARTITIONS AND BOUNDS

Given the graph approximation, G , of the hypergraph H , we find an initial partition of graph G by using the eigenvector-based approach of Barnes [2]. We will use the resulting partition of the weighted graph G as a partition for the hypergraph (netlist) 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 , under consideration has n nodes that are to be partitioned into k disjoint blocks of sizes $m_1 \geq m_2 \geq \dots \geq m_k$. A partition can be completely specified by a set of k node assignment vectors, x_1, x_2, \dots, x_k , one corresponding to each block, which have the form

$$x_j = (x_{1j}, x_{2j}, \dots, x_{nj})^T, \quad 1 \leq j \leq k$$

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 [2] shows that the solution of the following linear transportation problem gives an approximate solution to the graph partitioning problem:

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

$$\text{subject to } \sum_{i=1}^n x_{ij} = m_j, \quad j = 1, \dots, k$$

$$\sum_{j=1}^k x_{ij} = 1, \quad i = 1, \dots, n$$

$$x_{ij} \geq 0, \quad i = 1, \dots, n; \quad j = 1, \dots, k. \quad (7)$$

Technically, there are 2^k transportation problems (7) that should be solved where k is small. Barnes [2] describes a selection procedure that allows one transportation problem to be solved. A powerful interior point algorithm proposed by Adler *et al.* is used to solve the linear transportation problem (7) [1].

In the two-block case, the transportation problem (7) 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

$$\sum_{i=1}^n \left\{ \frac{v_{i1}}{\sqrt{m_1}} - \frac{v_{i2}}{\sqrt{m_2}} \right\} x_i + \sum_{i=1}^n \frac{v_{i2}}{\sqrt{m_2}}. \quad (8)$$

The transportation problem (7) reduces to the following $\{0, 1\}$ -knapsack problem:

$$\begin{aligned} & \text{Maximize } \sum_{i=1}^n \left\{ \frac{v_{i1}}{\sqrt{m_1}} - \frac{v_{i2}}{\sqrt{m_2}} \right\} x_i = \sum_{i=1}^n \delta_i x_i \\ & \text{subject to } \sum_{i=1}^n x_i = m_1, \quad x_i \in \{0, 1\}, \quad i = 1, \dots, n. \end{aligned} \quad (9)$$

It is well known that the solution to (9) is obtained by sorting the objective coefficients in nonincreasing 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 . Donath and Hoffman [9] introduced an approach that finds lower bounds on the weight of any cut of G . Thus we can find a lower bound on the number of cut generalized edges of H .

Consider the matrix $A = [a_{ij}]$, where a_{ij} is the weight of the edge joining nodes i and j (i.e., A is the adjacency matrix of G). The matrix A is symmetric and of order n with zeros along the main diagonal. Consider any diagonal matrix $U = [u_{ij}]$, where

$$\sum_{i=1}^n u_{ii} = -\sum_i \sum_j a_{ij}. \quad (10)$$

Then the main result of Donath and Hoffman [9] is

$$E_c \geq -\frac{1}{2} \sum_{i=1}^k m_i \lambda_i (A + U), \quad (11)$$

where E_c is the sum of the edges cut by the *optimal* partition and λ_i is the i th largest eigenvalue of $A + U$.

Now, let E_H be defined as the sum of the nets cut by the optimal partition. Since the number of cut edges for any partition of the generated graph G underestimates the number of cut nets for the *same* partition in the netlist, it follows that the sum of the optimal cut edges, E_c , is less than the sum of the optimal cut nets, E_H ; that is,

$$E_H \geq E_c \geq -\frac{1}{2} \sum_{i=1}^k m_i \lambda_i (A + U). \quad (12)$$

The bound provided by inequality (12) shows the importance of *tightly underestimating* the number of cut nets by the cut edges of the graph G given in subsection II-B.

We now illustrate the concepts that have been introduced by finding the best weighted graph approximation, partition, and lower bound on the smallest number of nets that are cut on the simple five-node, three-net example given in Fig. 2. The best underestimation graph fit for this netlist is given in Fig. 3.

Assume that we wish to partition the netlist into *two* blocks, one block containing three nodes and the other block two. The largest two eigenvalues of the adjacency matrix A (a_{ij} containing the weighted values connecting nodes i and j) of this graph and the corresponding eigen-

vectors are

$$\begin{aligned} \lambda_1 &= 1.7368, \quad v_1 = [0.548, 0.206, 0.206, 0.679, 0.391], \\ \lambda_2 &= 0.277, \quad v_2 = [0.223, 0.535, 0.535, -0.164, -0.592]. \end{aligned}$$

Substituting into (8), we find that the coefficients of the objective function are

$$\delta = [0.158, -0.259, -0.259, 0.508, 0.645].$$

The partition obtained by sorting the components in δ is

$$\begin{aligned} S_1 &= \{1, 4, 5\} \\ S_2 &= \{2, 3\}. \end{aligned}$$

Donath and Hoffman [9] obtain bounds on the number of cut edges in a graph by choosing the diagonal elements of $A + U$ by setting

$$\begin{aligned} u_{ii} &= -\sum_j a_{ij} \\ u_{ij} &= 0 \quad \text{for } i \neq j. \end{aligned}$$

In other words, the diagonal elements, $u_{ii} + A + U$ form the *negative sum* of all the weighted edges connected to node i .

We find that

$$E_H \geq E_c \geq 0.719$$

This implies that at least *one* (i.e., $\lceil 0.719 \rceil$) net is cut. Since the partition generated by S_1 and S_2 cuts exactly one net, we have provably obtained the *optimal partition*.

IV. TEST RESULTS

An available FORTRAN/C code, NETPART, has been developed on a UNIX environment to incorporate the eigenvector-based model partitioning method described in the preceding two sections. We present experimental results from applying the eigenvector method and the node interchange method to several networks. All computational work was done on a MIPS/2000 RISC computer at the University of Waterloo.

The FORTRAN subroutine LASO [21] was used to obtain the k largest eigenvalues and their corresponding eigenvectors. The LASO code is based on the block Lanczos method for finding a few largest or smallest eigenvalues and corresponding eigenvectors of a *sparse* matrix [7].

The calculation of the k largest eigenvalues and corresponding eigenvectors for larger netlists by LASO is modest [21]. This is accomplished by deleting the nets connected to a large number of modules (>20 modules). This has the effect of not introducing large-sized cliques in the graph with weighted edges; that is, the approximating graph of the netlist is kept sparse. These nets were put back into the netlist when the Sanchis multiblock interchange heuristic was used and the lower bounds were determined by inequality (12).

In the two-block case, a simple heap sort is used to solve the transportation problem (9). A general transportation problem solver using an interior point method was developed to handle the multiple-block case [1]. Our results show that the time required to solve the transportation problem is not significant in comparison with the time required to obtain the eigenvalues and eigenvectors using the block Lanczos code LASO.

A C program implementation of the efficient multiblock node interchange approach of Sanchis [20] which further reduces the number of cut nets of any initial partition was developed. This node interchange heuristic is capable of handling partitions involving an arbitrary number of blocks, and includes the feature of level gains to distinguish between node moves. Our implementation allows for nonuniform net weights without significantly changing the time complexity of the Sanchis method.

These techniques and the resulting computer codes were tested on five netlist partitioning problems, listed in Table II. Chip1 is taken from the work of Fiduccia and Mattheyses [10] and Primary1, Primary2, Bio, and Industry2 are taken from the MCNC gate-array and standard cell test suite benchmarks. These netlists vary in size from 300 to 12 000 nodes and 300 to 13 000 nets.

The preceding eigenvector-based procedure was tested on the netlists listed in Table II. In all cases, we attempted to partition the netlists into two, four, or six disconnected blocks of modules. Each block was allowed to have up to 10% more or less than the *equipartitioned* number of modules.

Table III shows the number of cut nets and running time to obtain these results by applying the eigenvector approach of Sections II and III to these netlists. The CPU time for the partition cuts include the time for forming the graph adjacency matrix, solving for the eigenvalues and eigenvectors, and finally solving the transportation problem.

Table IV shows the results obtained by supplying the multiblock node interchange heuristic of Sanchis [20] to improve the number of cut nets found using the eigenvector approach for the two, four, and six block cases given in Table III.

Tables V, VI, and VII give the results obtained from using node interchange on 30 *random* starting partitions. The *level parameter setting* was set to "1" in all test cases [20, pp. 64-66]. It was found that level parameter settings up to 4 led to negligible partition improvement on these test cases. One should note that the average net size does not exceed 4 for any of the test problems. However, the running times were substantially larger using level parameter settings larger than 1.

Each of these three tables contains seven columns. The second column gives the average number (mean) of cut nets obtained for each netlist example. Column 3 gives the standard deviation; columns 4 and 5 give the "best" and "worst" partitions obtained for each example. Finally, the last two columns yield the running time (second) and the number of runs that were conducted.

TABLE II
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
Primary1	904	833	3.50	1.29	3.22	2.59
Primary2	3029	3014	3.72	1.55	3.70	3.82
Bio	5711	6417	3.26	1.03	3.66	20.92
Industry2	12949	12142	3.89	1.79	3.64	11.15

TABLE III
EIGENVECTOR PARTITIONS

Name	2 Blocks		4 Blocks		6 Blocks	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	42	5.02	80	10.19	91	14.58
Primary1	181	13.58	298	74.16	329	32.10
Primary2	694	48.61	925	235.16	1097	238.69
Bio	364	134.74	873	209.85	906	341.68
Industry2	1383	201.78	4472	292.20	4102	614.14

TABLE IV
AFTER NODE INTERCHANGE

Name	2 Blocks		4 Blocks		6 Blocks	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	15	2.4	51	2.36	69	1.66
Primary1	63	3.33	120	5.36	149	7.08
Primary2	257	9.97	440	15.42	581	16.95
Bio	175	18.67	262	17.87	369	36.04
Industry2	530	39.32	1743	217.73	1689	190.10

TABLE V
TWO-BLOCK PARTITIONS: NODE INTERCHANGE

Name	\bar{x}	σ	Best	Worst	Time	Runs
Chip1	24.23	4.62	20	38	18.34	30
Primary1	71.17	11.09	53	101	73.57	30
Primary2	255.67	39.44	172	327	355.91	30
Bio	165.23	30.71	93	225	646.06	30
Industry2	774.37	206.99	393	1176	1585.03	30

TABLE VI
FOUR-BLOCK PARTITIONS: NODE INTERCHANGE

Name	\bar{x}	σ	Best	Worst	Time	Runs
Chip1	63.53	7.20	53	79	37.66	30
Primary1	180.93	15.35	140	204	137.55	30
Primary2	784.63	43.39	625	845	887.41	30
Bio	645.80	46.96	578	704	719.21	30
Industry2	2474.40	107.73	2266	2549	1365.18	30

TABLE VII
SIX-BLOCK PARTITION: NODE INTERCHANGE

Name	\bar{x}	σ	Best	Worst	Time	Runs
Chip1	83.60	7.31	69	97	52.20	30
Primary1	215.94	13.42	187	240	199.65	30
Primary2	913.23	34.39	806	961	2026.82	30
Bio	863.10	42.62	757	952	2951.34	30
Industry2	2881.90	90.80	2513	3025	18114.67	30

TABLE VIII
EIGENVECTOR BOUNDS

Name	2 Blocks		4 Blocks		6 Blocks	
	Cuts	Time	Cuts	Time	Cuts	Time
Chip1	7	12.27	19	22.56	28	36.45
Primary1	21	28.71	49	40.71	56	71.26
Primary2	93	80.16	191	194.39	224	386.74
Bio	51	209.17	83	299.38	119	443.66
Industry2	187	297.63	515	421.74	614	803.07

Some general observations can be made from the results presented in Tables II–VII. The final partitions that are obtained using the node interchange approach from the initial partition generated by the eigenvector technique 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 on this *one generated partition* is much less than the time required to perform the heuristic from 30 random starting partitions. For the four larger examples, the combined time of eigenvector and iterative improvement is from four to more than 20 times faster than using up to 30 random starting partitions.

In all cases, the results from the cascaded procedure of applying the eigenvector method and then iterative improvement are close to the best partitions that were obtained using iterative improvement alone from the random starting partitions. For most of the *larger sized* problems—Primary1, Primary2, Bio, and Industry2—this new approach yields cuts that are 50%–100% better than the cuts found for the partitions generated from random starting partitions only. Also, the quality of the resulting partitions is much better as the *size* of the problem increases and the number of *blocks* increases. The results are superior for the four and six block cases.

More important, the large test problems Bio and Industry2 show the importance of getting the best partition possible from using a node interchange approach *only once*. The running time of the interchange method from one starting partition is becoming expensive. Thus, the method becomes more attractive for solving these practical large partitioning problems.

Table VIII shows lower bounds obtained on the number of cut nets for the five tested problems using the Donath-Hoffman bounds given by inequality (12). The bounds generated in Table VIII can be tightened considerably by looking at other diagonal values, u_{ii} , satisfying (10). Cullum *et al.* [6] describe a nondifferentiable mathematical programming approach for selecting u_{ii} values that improve considerably the bounds of the inequality (12).

Rendl and Wolkowicz [18] recently introduced another eigenvector-based scheme for finding bounds on the number of cut edges in a graph with weighted edges. An important feature of their approach is that their bounds are *no worse* than the Donath-Hoffman bounds given by inequality (12). Preliminary results on small test examples

(i.e., up to 100 nodes) reveal that their approach generates bounds that are 25–200% better than the Donath-Hoffman bounds. This approach has the added advantage that it avoids the nondifferentiable mathematical programming analysis of Cullum *et al.* [6] required to compute these bounds. This latter approach is being investigated by the authors.

V. CONCLUSIONS

In this paper, we have introduced a new method to obtain initial node partitions for use in interchange heuristics where the aim is to minimize the number of nets (wires or signals) cut by the partition. This heuristic relies on a method to approximate a hypergraph or netlist by a graph with weighted edges in order to apply known partitioning results of Barnes [2] based on eigenvectors. The heuristic is very efficient in that the graph approximation of the hypergraph is easily obtained and the most expensive part of the procedure is finding and sorting the eigenvalues and eigenvectors corresponding to the adjacency matrix of the estimating graph.

An efficient implementation of Sanchis's interchange algorithm [20] is used to further reduce the number of cut nets for the k -partitions that are initially generated by the new eigenvector model. The outlined approach is tested on several small (300 modules/nets) to large size (12 000 modules/nets) netlist partitioning examples. This new approach is compared with the use of at most 30 random starting partitions and the interchange algorithm alone on these test examples.

Test results show that the new approach is comparable to the interchange approach on small examples but generates positions that are 50–100% better on the larger test cases. The largest test problems, Bio and Industry2, show the importance of getting the best partition possible from using a node interchange approach *only once*. The running time of the interchange method from one starting partition is becoming expensive. Thus, the method becomes more attractive for solving these practical large cases.

The calculation of the k largest eigenvalues and corresponding eigenvectors for larger netlists by the block Lanczos code is modest [21]. This is accomplished by deleting the nets connected to a large number of modules (>20 modules). This has the effect of not introducing large-sized cliques in the graph with weighted edges; that is, the approximating graph of the netlist is kept sparse. Future practical codes should attempt to balance *denser* graph approximations and the running time to find initial partitions. This work is currently in progress.

REFERENCES

- [1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming*, vol. 44, pp. 297–335, 1989.
- [2] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM J. Algebraic and Discrete Methods*, vol. 3, no. 4, pp. 541–550, 1982.

- [3] J. P. Blanks, "Near-optimal placement using a quadratic objective function," in *Proc. Design Automat. Conf.*, 1985.
- [4] R. K. Brayton, G. Hachtel, K. Mullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Boston: Kluwer Academic Publishers, 1984.
- [5] A. M. Comeau, "A study of user program optimization in a paging system," in *Proc. ACM Symp. Operating System Principles* (Gatlingburg, TN), Oct. 1967.
- [6] J. Cullum, W. E. Donath, and P. Wolfe, "The minimization of certain nondifferentiable sums of eigenvalues of symmetric matrices," *Mathematical Programming Study*, vol. 3, pp. 35-55, 1975.
- [7] J. Cullum and R. H. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalues Computation*, vols. I and II. Boston: Birkhauser, 1985.
- [8] P. J. Denning, "Virtual memory," *Comput. Surveys*, vol. 2, pp. 153-189, 1970.
- [9] W. E. Donath and A. J. Hoffman, "Lower bounds for the partitioning of graphs," *IBM J. Res. Develop.*, vol. 17, no. 5, pp. 420-425, 1973.
- [10] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. 19th Design Automat. Conf.*, 1982, pp. 175-181.
- [11] A. D. Friedman and P. R. Menon, *Theory and Design of Switching Circuits*. Rockville, MD: Bell Telephone Laboratories and Computer Sciences Press, 1975.
- [12] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA: Freeman, 1979.
- [13] J. R. Gilbert and E. Zmijewski, "A parallel graph partitioning algorithm for a message passing multiprocessor," *Int. J. Parallel Programming*, vol. 16, pp. 427-449, 1987.
- [14] K. Hall, "An r -dimensional quadratic placement algorithm," *Management Sci.*, vol. 17, pp. 219-229, 1970.
- [15] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373-395, 1984.
- [16] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291-307, 1970.
- [17] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Analysis and Appl.*, vol. 11, pp. 430-452, 1990.
- [18] F. Rendl and H. Wolkowicz, "A projection technique for partitioning the nodes of a graph," Research Report CORR 90-20, Dept. Combinatorics and Optimization, University of Waterloo, 1990 (submitted to *Mathematical Programming*).
- [19] R. L. Russo, P. H. Oden, and P. K. Wolff, Sr., "A heuristic procedure for the partitioning and mapping of computer logic blocks to modules," *IEEE Trans. Comput.*, vol. C-20, pp. 1455-1462, 1972.
- [20] L. A. Sanchis, "Multiple-way network partitioning," *IEEE Trans. Comput.*, vol. 38, no. 1, pp. 62-81, 1989.
- [21] D. S. Scott, "LASO2 documentation," Computer Science Department, University of Texas at Austin, 1980.
- [22] A. J. Vakharia, "Methods of cell formation in group technology: A framework for evaluation," *Int. J. Operations Management*, vol. 6, no. 3, pp. 257-271, 1986.
- [23] A. Vannelli and K. R. Kumar, "A method for finding minimal bottleneck cells for grouping part-machine families," *Int. J. Prod. Res.*, vol. 24, pp. 387-400, 1986.
- [24] A. Kusiak, A. Vannelli, and K. R. Kumar, "Clustering analysis: Models and algorithms," *Control and Cybernetics*, vol. 15, no. 2, pp. 139-154, 1986.
- [25] A. Vannelli and S. W. Hadley, "A Gomory-Hu cut tree approach for partitioning netlist," *IEEE Trans. Circuits Syst.*, vol. 37, no. 9, pp. 1133-1139, 1990.

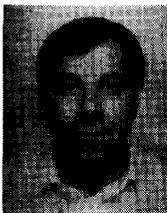


Scott W. Hadley received the B.Math, M.Math and Ph.D. (combinatorics and optimization) degrees from the University of Waterloo, Waterloo, Ontario, Canada, in 1981, 1987, and 1990.

He is presently a Research Mathematician in the Department of Mathematics and Systems Engineering at Koninklijke/Shell-Laboratorium, Amsterdam, Netherlands. His main research interests lie in the area of practical applications of large-scale combinatorial optimization problems.



Brian L. Mark received the B.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Waterloo, Ontario, Canada, in 1991. He is currently undertaking his graduate studies (M.A.Sc. and Ph.D.) in electrical engineering at Princeton University, Princeton, NJ. His main research interests lie in communication theory.



Anthony Vannelli received the Ph.D. degree in electrical engineering from the University of Waterloo, Waterloo, Ontario, Canada, in 1983. In 1983 and 1984 he was an IBM postdoctoral fellow in the Mathematical Sciences Department at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

He joined the Department of Industrial Engineering at the University of Toronto in 1984. In 1987, he joined the Department of Electrical and Computer Engineering at the University of Waterloo.

He is currently an Associate Professor and has held a Natural Sciences and Engineering Council of Canada University Research Fellowship since 1984. His research focuses mainly on the development of efficient linear, nonlinear, and combinatorial optimization techniques for solving VLSI circuit layout and design problems.