SIMPLE EIGENVECTOR-BASED CIRCUIT CLUSTERING CAN BE EFFECTIVE

Charles J. Alpert *Andrew B. Kahng*
UCLA Computer Science Department, Los Angeles, CA 90095-1596 USA
cheese@cs.ucla.edu and abk@cs.ucla.edu

ABSTRACT

Clustering has proven effective in improving the quality of VLSI netlist partitioning and placement algorithms. A wide variety of clustering schemes have been proposed, including random walks [13], iterative matching [7], and fairly complicated spectral techniques [1] [8]. Like [1] and [8], we use eigenvectors to compute a clustering, but do so in the simplest, most obvious manner. Our algorithm first computes a d-digit code for each module v_i according to the signs of the i^{th} entries in a set of d eigenvectors. Then, modules with the same code are assigned to the same cluster. Despite its simplicity, this new clustering algorithm is strongly motivated by theoretical results for both spectral bipartitioning [6] and multi-dimensional vector partitioning [4]. The algorithm also has linear time complexity (not including the eigenvector computation) and is at least as effective as previous clustering algorithms in terms of two-phase Fiduccia-Mattheyses bipartitioning.

1. INTRODUCTION

Clustering of netlist hypergraphs can effectively reduce the complexity of VLSI CAD problem instances, particularly for system partitioning and layout. For netlist bipartitioning, the two-phase Fiduccia-Mattheyses (FM) methodology [7] [10] has led to several leading results in recent years. In this approach, the netlist is first decomposed into disjoint *clusters*, i.e., subsets of modules, which induce a *contracted netlist*.

Formally, let $H(V, E)$ be a netlist hypergraph over the n modules $V = \{v_1, v_2, \ldots, v_n\}$; each net $e \in E$ is a subset of two or more modules. A *clustering* of H is a set of clusters $\{C_1, C_2, \ldots, C_k\}$ such that each $v_i \in V$ is contained in a unique cluster $C_h, 1 \leq h \leq k$. The contracted netlist $H'(V', E')$ has vertex set $V' = \{C_1, C_2, \ldots, C_k\}$. The set of nets E' potentially has one net $e' \in E'$ for each $e \in E$: we assign $C_h \in e'$ if and only if there exists some $v_i \in V$ such that $v_i \in e \cap C_h$, but if $|e'| < 2$, we do not actually add e' to E'. The first phase of two-phase FM applies the FM algorithm to H'; this yields a bipartitioning of V' into subsets P_1' and P_2'. A bipartitioning $\{P_1, P_2\}$ of V is then derived by assigning v_i to P_j if $v_i \in C_h$ and $C_h \in P_j$. The second phase applies FM to the original netlist $H(V, E)$ with $\{P_1, P_2\}$ as the initial solution.

Two-phase FM improves the number of nets cut versus standard FM for virtually any clustering algorithm. Two-phase FM also requires shorter runtimes than standard FM since the first phase has small instance size, and since a good initial starting solution causes the second phase of FM to converge much faster than with a random initial solution. Similar "two-phase" strategies can also be applied to cell placement, e.g., Sun and Sechen [19] use a two-level hierarchical clustering strategy (actually a "three-phase" approach) within the simulated-annealing based Timberwolf placement package. Applying clustering to FM is not inherently limited to two phases – Hauck and Borriello [15] have experimented with a *multilevel* technique which continues to run clustering phases as long as user-specified cluster size constraints are satisfied. Since clustering serves to reduce instance complexity, it has nearly universal application in VLSI CAD areas ranging from timing-driven layout to high-level synthesis.

The clustering literature contains many strategies with varying complexities (see [3] for a survey). Simple approaches include iterative matching [7] and agglomerative connectivity-based merging [15]. These algorithms merge connected modules together in a greedy, bottom-up fashion. More complicated approaches are based on random walks [13], clique compression [9], graph traversals [2], simulated annealing [19], $k-l$ connectivity [11], and top-down ratio cut [21]. Finally, two spectral approaches [1] [8] each form a d-dimensional geometric embedding of modules, using the coordinates defined by (entries of) the first d eigenvectors. The embedding forms a d-dimensional representation of the original netlist in which each module maps to a point in d-space. As discussed in Section 3, the embedding possesses many properties which make it a well-suited geometric representation of the netlist. Given this embedding, Alpert and Kahng [1] perform bottom-up clustering based on a min-diameter criterion, while Chan et al. [8] construct a d-way clustering by first selecting a set of d orthogonal "prototype" vectors and then clustering each module to its closest "prototype" according to a directional cosine metric.

Our new approach also uses the standard spectral embed-

ding, but in the simplest possible manner: if two modules are in the same orthant of the embedding, they are assigned to the same cluster. In contrast to complicated geometric clustering techniques, our algorithm is completely obvious and, in addition, has strong theoretical motivation. We show that this algorithm is a natural extension of spectral bipartitioning [6] and also follows the recent result of [4] which establishes the equivalence of min-cut graph partitioning and an eigenvector-based vector partitioning formulation. We have tested the quality of these clusterings via two-phase FM; our experiments show that simple eigenvector-based clustering produces bipartitionings with cuts that are at least as good as previous methods [1] [2] [19]. Thus, eigenvector-based clustering can be both simple and effective.

2. THE ALGORITHM

For spectral methods to be applied, an $n \times n$ adjacency matrix $A = (a_{ij})$ must first be derived from H, where a_{ij} gives the weight of the connections between modules v_i and v_j. We choose to construct A using the *clique* net model which adds weight $f(|e|)$ to a_{ij} if v_i and v_j are both in net e, where f is a function of the size of e. We adopt the weight function $f(|e|) = \frac{6}{|e|(|e|+1)}$ since it is the appropriate weighting scheme for linear placement[20].[1] Let $D = (d_{ij})$ be a diagonal matrix with $d_{ii} = \sum_{j=1}^{n} a_{ij}$. The $n \times n$ *Laplacian matrix* of A is given by $Q = D - A$. The eigenvectors $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_n$ of Q form an orthonormal basis in n-space and have corresponding eigenvalues $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$. The i^{th} entry of $\vec{\mu}_j$ is given by[2] μ_{ij}.

Our algorithm works as follows. First, the d eigenvectors $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_d$ are computed, where d is a user-defined parameter. Next, the d-digit binary code $code[i]$ is computed for each $v_i \in V$, where bit j of $code[i]$ is one if $\mu_{ij} \geq 0$ and zero otherwise. Finally, modules with the same binary code are assigned to the same cluster, i.e., each cluster corresponds to one of the 2^d orthants of the d-dimensional eigenvector embedding. The complete algorithm is described in Figure 1. The time complexity of steps 3 and 4 is only $O(nd)$, so the eigenvector computation (which has expected complexity $O(n^{1.4})$ for sparse graphs if we use the Lanczos iteration) dominates the time complexity.[3]

[1] Note that any eigenvector is a one-dimensional (i.e., linear) placement of the modules. Empirically, we observe a strong correlation between the cutsize of the graph given by A and number of nets cut by H.

[2] Using the eigenvectors of Q, rather than of A, offers several advantages [16]. For example, the number of eigenvalues of Q equal to zero is also the number of connected components of H. In addition, the sum of the entries of each eigenvector with a non-zero eigenvalue equals zero, which guarantees that the center of mass of the geometric embedding lies at the origin. The theoretical equivalence (or unequivalence) of Q and A has not yet been established.

[3] Our experiments show that computing eigenvectors with LASO code [18] is relatively efficient. For example, Sparc-1000 (single processor) runtimes were respectively 133 seconds and 42 minutes to compute 10 eigenvectors for the primary2 and avq.large circuits. In addition, much research is currently devoted to speeding up spectral computations via parallel and multilevel methods. For example, Barnard and Simon [5] use a multilevel contraction method to approximate the second eigenvector for a finite element graph with 262,620 nodes and 764,268 edges (in 152 seconds); the subsequent partitioning of this eigenvector led to a lower cut than pure spectral bisection.

Clustering Algorithm

1. Compute an adjacency matrix A from $H(V, E)$, e.g., via a clique net model.
2. Compute the eigenvectors $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_d$ from the Laplacian Q of A.
3. For all $v_i \in V$, compute the d-digit code $code[i]$ where digit j is 1 if $\mu_{ij} \geq 0$ and 0 if $\mu_{ij} < 0$.
4. For all $v_i \in V$, assign v_i to cluster $C_{code[i]}$. Return the set of non-empty clusters.

Figure 1. SIMPLE Eigenvector clustering algorithm.

3. THEORETICAL MOTIVATION

Theoretical properties of eigenvectors and spectral partitioning justify our simple eigenvector-based clustering strategy.

3.1. Relation to Spectral Bipartitioning

A bipartitioning $\{P_1, P_2\}$ can be represented as an n-dimensional *indicator vector* $\vec{x} = (x_i)$ with $x_i = 0$ if $v_i \in P_1$ and $x_i = 1$ if $v_i \in P_2$. The well-known *spectral bipartitioning* algorithm [6] computes the second eigenvector $\vec{\mu}_2$ of Q and sets x_i to 1 if $\mu_{i2} \geq 0$ and x_i to 0 otherwise (assuming no cluster size constraints). This choice of $\vec{x}$ maximizes $\vec{x} \cdot \vec{\mu}_2$, i.e., $\vec{x}$ is the indicator vector that *maximally projects* onto $\vec{\mu}_2$. Hall [14] showed that the optimal solution according to the objective

$$f(\vec{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(x_i - x_j)^2, \qquad (1)$$

such that $||\vec{x}||^2 = 1$ and $\sum_{i=1}^{n} x_i = 0$, is given by $\vec{x} = \vec{\mu}_2$. (The trivial solution $\vec{x} = \vec{\mu}_1 = [1, 1, \ldots, 1]$ cannot be scaled to satisfy both constraints.) If $\vec{x}$ is the indicator vector for the partitioning $\{P_1, P_2\}$ of the graph represented by A, then $f(\vec{x})$ is exactly the total weight of edges cut by the partitioning solution. Thus, $\vec{\mu}_2$ can be viewed as the optimal, although illegal, partitioning solution; the spectral bipartitioning algorithm finds the legal indicator vector $\vec{x}$ closest to $\vec{\mu}_2$.

Chan et al. [8] have shown that the d best non-discrete solutions to Equation (1) are given by $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_d$ under the constraint that all solutions are mutually orthogonal. Spectral bipartitioning can be run on these eigenvectors to generate d distinct indicator vectors, and since the entries in these indicator vectors form the binary code in our algorithm, our method can be viewed as assigning modules

to the same cluster if they are in the same spectral bipartitioning solutions from $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_d$. Let $\vec{x}^h = (x_i^h)$ be the indicator vector for cluster C_h returned by our algorithm (entry i is 1 if $v_i \in C_h$ and 0 otherwise). Then the projection of $\vec{x}^h$ onto eigenvector $\vec{\mu}_j$ (i.e., $\vec{x}^h \cdot \vec{\mu}_j$), $1 \leq j \leq d$, is either the sum of all nonpositive or all nonnegative terms. In other words, all projection components $x_i^h \cdot \mu_{ij}$, $1 \leq i \leq n$, have the same sign. Thus, the sum total of projection magnitudes for the k indicator vectors onto the first d eigenvectors is given by $\sum_{h=1}^{k} \sum_{j=1}^{d} |\vec{x}^h \cdot \vec{\mu}_j|$ and is maximum. Thus, our clustering is optimal with respect to projection magnitudes, and is a natural extension of spectral bipartitioning in that this clustering is as "close" as possible to the first d eigenvectors.

3.2. Relation to Vector Partitioning

Recently, Alpert and Yao [4] showed that min-cut k-way graph partitioning exactly reduces to the following *vector partitioning* problem: given a set of vectors $Y = \{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$ in d-dimensional space, partition the vectors into subsets $\{S_1, S_2, \ldots, S_k\}$ such that

$$|| \sum_{\vec{y}_i \in S_1} \vec{y}_i||^2 + || \sum_{\vec{y}_i \in S_2} \vec{y}_i||^2 + \ldots + || \sum_{\vec{y}_i \in S_k} \vec{y}_i||^2$$

is maximized. In other words, the vectors in each subset S_h are summed to form a *subset vector* $\vec{Y}_h = \sum_{\vec{y}_i \in S_h} \vec{y}_i$, and the objective is to maximize the sum of the squared magnitudes of the subset vectors.

If $\vec{y}_i$ is given by the n-vector $[(H - \lambda_1)\mu_{i1}, (H - \lambda_2)\mu_{i2}, \ldots, (H - \lambda_n)\mu_{in}]$ (where H is some constant larger than λ_n), then the optimal solution to this vector partitioning problem exactly corresponds to the optimal solution to the associated graph partitioning problem. The graph partitioning solution $\{C_1, C_2, \ldots, C_k\}$ is constructed by assigning v_i to C_h if and only if $\vec{y}_i \in S_h$. Furthermore, the number of nets cut by a given cluster C_h is directly expressible in terms of the subset vectors $\vec{Y}_h$, namely, the cut of cluster C_h is exactly given by $H|C_h| - ||\vec{Y}_h||^2$. Thus, maximizing the magnitude of $\vec{Y}_h$ serves to minimize the cut of cluster C_h.

Observe that each $\vec{y}_i$ vector belongs to one of the 2^n orthants of n-space. If all the vectors in one orthant are added together, it is likely that they sum to a vector of large magnitude since they point roughly in the same direction. In fact, no two vectors in the same orthant can be more than 90 degrees apart. It follows that assigning vectors in the same orthant to the same subset S_h should result in a subset vector $\vec{Y}_h$ with large magnitude, and hence a cluster C_h with small cut. Our clusterings are derived in exactly this manner, except that we only use the first $d \leq n$ eigenvectors, resulting in 2^d possible orthants. These eigenvectors comprise the best d-dimensional approximation of the n-dimensional vector partitioning instance.

4. EXPERIMENTAL RESULTS

We tested our clustering algorithm, which we call SIMPLE, on 14 circuit test cases that are commonly cited in the partitioning literature. These benchmarks are available from the site http://ballade.cs.ucla.edu/~cheese. Our clusterings were constructed using $d = 11$ (i.e., 10 nontrivial eigenvectors)[4], and we compare to the clustering algorithms WINDOW [2], ANNEAL [19], and AGG [1]. WINDOW first traverses the graph to induce a linear ordering of the modules, then splits the ordering into a clustering via dynamic programming. ANNEAL is the simulated annealing based algorithm used by Timberwolf to perform a "three-phase" cell placement. Finally, AGG also uses a d-dimensional spectral embedding, but clusters according to the Euclidean distance metric.

For each benchmark and each clustering, we ran two-phase FM 100 times with unit areas and exact bisection size constraints on the partitions. Our FM code was obtained from the authors of [12] and uses a last-in-first-out (LIFO) tie-breaking scheme in the gain buckets. This scheme has recently been shown to significantly outperform traditional random or last-in-first-out (FIFO) tie-breaking schemes [12], and consequently the two-phase FM results reported for WINDOW, ANNEAL, and AGG are superior to those reported in [2]. The lowest number of cut nets observed is reported in Table 1, with the average cut sizes given in parentheses. We also ran FM without clustering, and these results are given in the third column of the Table. Note that clusterings for AGG were only available for the first 9 benchmarks, and that since AGG constructs a clustering for each of the first 10 spectral embeddings, we ran two-phase FM 10 times on each clustering and combined the results.

The cuts given by SIMPLE are certainly competitive with the outputs of the other algorithms. SIMPLE yields smallest cut overall for seven of the benchmarks, and the best average cut for five of them. Surprisingly, the improvement of two-phase FM results over flat FM is not that impressive, especially for the three largest benchmarks. One possible explanation is that LIFO tie-breaking significantly improves "standard" FM, leaving less room for improvement by clustering; this observation was confirmed in [12].

Although SIMPLE two-phase results are not definitively superior, we find the simplicity (even naiveté) of the clustering algorithm to be its most attractive quality. The algorithm performs spectral clustering in the most obvious manner (e.g., as opposed to AGG), yet its clusterings are just as effective as the best previous methods. In addition, the algorithm is strongly motivated by theoretical results for spectral partitioning and vector partitioning. Our future

[4]For avq.small and avq.large, the clique net model constructs too many edges to make eigenvector computations feasible. Hence, we sparsified the matrix A by constructing two random tours over each net with six or more pins, and assigned each weight 1/6 to each edge.

Benchmark	#Mods	#Nets	#Pins	FM	WINDOW	AGG	ANNEAL	SIMPLE
primary1	833	902	2908	59(83)	**54**(**73**)	56(75)	**54**(76)	55(80)
primary2	3014	3029	11219	238(296)	183(263)	184(253)	156(242)	**147**(**231**)
test02	1663	1720	6134	122(177)	**94**(**113**)	97(140)	105(145)	99(124)
test03	1607	1618	5807	76(126)	63(**91**)	74(105)	70(108)	**61**(99)
test04	1514	1658	5975	86(140)	56(86)	64(99)	82(111)	**53**(**84**)
test05	2595	2750	10076	112(194)	**75**(**117**)	99(141)	107(150)	90(129)
test06	1752	1541	6638	69(96)	66(83)	**65**(**78**)	73(89)	69(88)
bm1	882	903	2910	58(83)	58(71)	54(**70**)	58(77)	**52**(71)
19ks	2844	3282	10547	137(184)	124(**153**)	**115**(177)	132(194)	123(169)
biomed	6514	5742	21040	92(176)	143(201)		104(205)	**86**(**130**)
industry2	12637	13419	48404	674(1151)	264(503)		466(695)	**255**(**389**)
industry3	15393	21919	65818	318(711)	**298**(793)		357(813)	329(**620**)
avqsmall	21918	22124	76231	343(650)	**239**(**498**)		368(724)	385(662)
avqlarge	25178	25384	82751	324(796)	415(**645**)		450(835)	**320**(657)
WINS				0(0)	4.5(7)	2(2)	0.5(0)	7(5)

Table 1. Results for 100 two-phase FM cuts for exact bisection with unit areas. The smallest cut and average cut (in parentheses) are reported for each algorithm. The lowest cuts for each benchmark are in boldface, and the last row gives the number of "wins" for each algorithm, i.e., the number of times the algorithm yielded the lowest cut.

work seeks to combine eigenvector clustering and iterative bipartitioning into a multilevel clustering framework.

ACKNOWLEDGMENTS

This work was supported by NSF Young Investigator Award MIP-9257982 and by a UCLA Dissertation Year Fellowship.

REFERENCES

[1] C. J. Alpert and A. B. Kahng, "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning," *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 743-748.

[2] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering," *IEEE Intl. Conf. on Computer-Aided Design*, 1994, pp. 63-67 (extended version to appear in *Trans. on VLSI Systems*).

[3] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey," *Integration: the VLSI Journal*, 19(1-2), 1995, pp. 1-81.

[4] C. J. Alpert and S.-Z. Yao, "Spectral Partitioning: The More Eigenvectors, the Better," *Proc. ACM/IEEE Design Automation Conf.*, 1995, pp. 195-200.

[5] S. T. Barnard and H. D. Simon, "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems," *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, 1993, pp. 627-632.

[6] E. R. Barnes, "An Algorithm for Partitioning the Nodes of a Graph," *Siam J. Algorithms and Discrete Methods* (3)4, 1992, pp. 541-549.

[7] T. Bui, C. Heigham, C. Jones, and T. Leighton, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms," *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 775-778.

[8] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD* 13(9), 1994, pp. 1088-1096.

[9] J. Cong and M. Smith "A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design" *Proc. ACM/IEEE Design Automation Conf.* 1993, pp. 755-760.

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

[11] J. Garbers, H. J. Promel and A. Steger, "Finding Clusters in VLSI Circuits" *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 520-523.

[12] L. Hagen, J.-H. Huang, and A. B. Kahng, "On Implementation Choices for Iterative Improvement Partitioning Algorithms", *Proc. European Design Automation Conf.*, September, 1995.

[13] L. Hagen and A. B. Kahng, "A New Approach to Effective Circuit Clustering", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1992, pp. 422-427.

[14] K.M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Manag. Sci.*, 17, pp. 219-229, 1970.

[15] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techniques", *Proc. Chapel Hill Conf. on Adv. Research in VLSI*, 1995.

[16] B. Mohar, "The Laplacian Spectrum of Graphs", in Y. Alavi and et al., editors, *Graph Theory, Combinatorics, and Applications*, 1991, pp. 871-898.

[17] B. M. Riess, K. Doll, and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 646-651.

[18] D. S. Scott, "LASO2 Documentation", *technical report*, CS Dept., University of Texas at Austin, 1980.

[19] W. Sun and C. Sechen, "Efficient and Effective Placements for Very Large Circuits" *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1993, pp. 170-177.

[20] R.-S. Tsay and E. S. Kuh, "A Unified Approach to Partitioning and Placement", *IEEE Trans. Circuits and Systems*, 38(5), 1991, pp. 521-533.

[21] Y.-C. A. Wei and C.-K. Cheng, "An Improved Two-Way Partitioning Algorithm with Stable Performance, *IEEE Trans. Computer-Aided Design*, 10(12), 1991, pp. 1502-1511.