Two-way Partitioning Based on Direction Vector

K.S.Seong and C.M.Kyung
Dept. of EE, KAIST
373-1 Kusong-dong, Yusong-gu, Taejon 305-701, Korea.
kyung@dalnara.kaist.ac.kr, kssung@dalnara.kaist.ac.kr

Abstract

In spectral method, the vertices in a graph can be mapped into the vectors in d-dimensional space, thus the vectors are partitioned instead of vertices to obtain graph partitioning. In this paper, we show a method to obtain optimal two-way vector partitioning based on *optimal direction vector*. As the problem to find the *optimal direction vector* is NP-problem, we propose an efficient heuristic to obtain high quality *direction vector*. As we approximate a given netlist into the graph and only use ten eigenvectors in practice, there is a chance to improve the solution quality by local optimization. Fiduccia-Mattheyses algorithm is employed as a post processing. Compared with FM and MELO, the proposed algorithm PDV reduces cutsize on the average 40% and 20.5%, respectively.

1 Introduction

With the ever-growing complexity of the electronic systems being integrated into a single chip or multiple chips, how the whole circuit is partitioned into multiple blocks or into multiple chips drastically affects the whole system performance.

Many approaches have been proposed to solve the partitioning problem with size constraints. The move-based approach starts with some initial solution and iteratively improves the solution by moving to the best neighboring solution [1][2][3]. Fiduccia-Mattheyses[2] proposed a linear time two-way partitioning algorithm (called FM) which exploits the sparsity of a netlist. The FM algorithm is widely used as it is easy to implement and fast. The FM algorithm, however, can be trapped in local minima, *i.e.*, final result is heavily affected by the initial solution.

Spectral method uses the eigenvectors and the eigenvalues of the Laplacian of a graph G. As the vertices in the graph can be mapped into the vectors in d-dimensional space using spectral method, the vectors are partitioned instead of vertices. Alpert et al. showed the relations between graph partitioning and vector partitioning[4]. However the relations were not fully utilized to obtain two-way vector partitioning.

In this paper, we propose a method to obtain two-way vector partitioning based on the so-called *direction vector*. If we find the *optimal direction vector*, we can obtain the optimal two-way vector partitioning, *i.e.*, the vector partitioning problem is converted to finding the *optimal direction vector*. Because the problem to find the *optimal direction vector* is NP-problem, we propose an efficient heuristic to obtain high quality *direction vector* with iterative approach. Each netlist is approximately transformed into the graph using clique model, and then the eigenvectors and eigenvalues of the Laplacian of the graph are calculated. Practically, only small number of eigenvectors are used in vector partitioning. Thus, there is a chance to improve the solution quality by local optimization. Fiduccia-Mattheyses algorithm is employed as a post processing.

This paper is organized as follows. Some preliminaries are described in section 2. In section 3, we convert the two-way vector partitioning problem into finding the *optimal direction vector*. In section 4, a heuristic to obtain high quality direction vector with iterative approach is proposed. Experimental results are given in section 5.

2 Preliminary

A circuit netlist which is well represented as a hypergraph can be transformed to a graph using clique net model [4][5]. The transformed graph $G(V, E)$ consists of vertex set, $V = \{v_1, v_2, \ldots, v_n\}$, and edge set $E = \{e_1, e_2, \ldots e_m\}$. The edge costs between the vertices are represented as a symmetric $n \times n$ *adjacency matrix*, $A = (a_{ij})$, where a_{ij} is the edge cost between v_i and v_j. If there exists edge between v_i and v_j, a_{ij} is larger than zero. Otherwise, a_{ij} is zero. Here, we can define $n \times n$ *degree matrix* $D = (d_{ij})$, where d_{ii} is degree of vertex v_i, *i.e.*, $d_{ii} = \sum_{j=1}^{n} a_{ij}$, and $d_{ij} = 0$ if $i \neq j$. The $n \times n$ *Laplacian matrix* of the graph G is defined as $Q = D - A$.

Given n vertices represented as each element in the

ED&TC '97 on CD-ROM

vertex set, $V = \{v_1, v_2, \ldots, v_n\}$, a k-way graph partitioning, P_g^k, is to divide n vertices into a set of k disjoint clusters, $\{C_1, C_2, \ldots, C_k\}$. Size-constrained k-way min-cut graph partitioning is to find a partitioning, P_g^k, which minimizes the following cost function under the size constraints.

$$F(P_g^k) = \sum_{l=1}^{k} E_l, \quad \text{where} \quad E_l = \sum_{v_i \in C_l, v_j \notin C_l} a_{ij}$$

For the given *Laplacian* matrix of a graph G, the set of eigenvectors are denoted by $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_n}$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n (\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n)$. We can define $n \times d$ scaled eigenvector matrix, V_d, for some constant $H \geq \lambda_d$, as follows;

$$V_d = [\vec{\mu_1}\sqrt{H-\lambda_1}, \vec{\mu_2}\sqrt{H-\lambda_2}, \ldots, \vec{\mu_d}\sqrt{H-\lambda_d}]$$

where $d \leq n$. Each vertex, v_i, is mapped to a vector $\vec{y_i^d}$ in d-dimensional space, which denotes the i-th row of V_d. The smallest eigenvalue $\lambda_1 = 0$ has the corresponding eigenvector $\vec{\mu_1} = [\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}]^T$. It does not contribute to partitioning, thus we can disregard the first column of the scaled eigenvector V_d.[4]

Consider the problem of partitioning n vectors in d-dimensional space represented as vector set $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$, where each vertex v_i corresponds to the vector $\vec{y_i^d}$. The definition of k-way vector partitioning, P_v^k, is to divide n vectors in Y into a k disjoint set of vectors, $\{S_1, S_2, \ldots, S_k\}$. If $d = n$ and P_v^k corresponds to P_g^k, the following property holds.[4]

Property 2.1
If $F(P_v^k) = \sum_{l=1}^{k} \|\vec{Y_l^n}\|^2$ is maximum for a given vector partitioning, where $\vec{Y_l^n} = \sum_{\vec{y_i^n} \in S_l} \vec{y_i^n}$, then the corresponding graph partitioning P_g^k is optimum, i.e., $F(P_g^k) = \sum_{l=1}^{k} E_l$ is minimum.

There is useful property in case of discarding the first eigenvector, $\vec{\mu_1}$, we do not consider that in this paper.

Property 2.2
When the first eigenvector $\vec{\mu_1}$ is discarded in V_d, the result of summing vectors in $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$ becomes zero vector.

In case of two-way vector partitioning $P_v^2 = \{S_L, S_R\}$, $\vec{Y_L} + \vec{Y_R} = \vec{0}$. Thus, $F(P_v^2) = \|\vec{Y_L}\|^2 + \|\vec{Y_R}\|^2 = 2\|\vec{Y_R}\|^2$ For more detail about vector partitioning, refer to [4].

3 Optimal two-way vector partitioning

In this section, we will show a procedure to obtain optimal vector partitioning with the *optimal direction vector*. A vector partitioning, $\{S_L, S_R\}$, which satisfies size-constraints and maximizes $\|\vec{Y_R}\|^2 = \|\sum_{\vec{y_i} \in S_R} \vec{y_i}\|^2$ is called optimal two-way vector partitioning. The definition of optimal direction vector is defined as follows;

Definition 3.1 (Optimal direction vector)
For a given optimal two-way vector partition $\{S_L, S_R\}$, we can define optimal direction vector $\vec{x_o}$ whose direction is the same as $\vec{Y_R} = \sum_{\vec{y_i} \in S_R} \vec{y_i}$ and length is unity.

Definition 3.2 (I_R, I_L and I_Y)
The elements in I_R, I_L and I_Y are the projection of vectors on S_R, S_L and Y in the direction of the optimal direction vector, $\vec{x_o}$, respectively. Thus, they can be represented as $I_R = \{p_i | p_i = \vec{y_i} \cdot \vec{x_o}, \ \vec{y_i} \in S_R\}$, $I_L = \{p_i | p_i = \vec{y_i} \cdot \vec{x_o}, \ \vec{y_i} \in S_L\}$ and $I_Y = \{p_i | p_i = \vec{y_i} \cdot \vec{x_o}, \ \vec{y_i} \in Y\}$.

Theorem 3.1
If a two-way vector partition $\{S_L, S_R\}$ is optimal, the maximum value p_{Lmax} in $I_L = \{p_i | p_i = \vec{y_i} \cdot \vec{x_o}, \vec{y_i} \in S_L\}$ is not larger than the minimum value p_{Rmin} in $I_R = \{p_i | p_i = \vec{y_i} \cdot \vec{x_o}, \vec{y_i} \in S_R\}$.
proof:
As the vector partitioning $\{S_R, S_L\}$ is optimal, $\|\vec{Y_R}\|^2 = |\sum_{p_i \in I_R} p_i|^2$ is maximum. If the P_{Lmax} is larger than P_{Rmin}, we can obtain another set $I_{L'}, I_{R'}$ by interchanging p_{Lmax} and p_{Rmin}. Then

$$|\sum_{p_i \in I'_R} p_i|^2 = |\sum_{p_i \in I_R} p_i + p_{Lmax} - p_{Rmin}|^2 \geq |\sum_{p_i \in I_R} p_i|^2$$

This is a contradiction to the assumption that $|\sum_{p_i \in I_R} p_i|^2$ is maximum. **Q.E.D.**

Theorem 3.2
If we know the optimal direction vector, we can find the optimal two-way vector partitioning.
proof : *When p_i's in I_Y are partitioned into I_L and I_R, the maximum values in I_L is not larger than the minimum values in I_R if the partitioning is optimal. When the $p'_i s$ in I_Y are sorted as $(p_{\pi_1} \leq p_{\pi_2} \leq \ldots \leq p_{\pi_n})$, there are $(n-1)$ partitions satisfing the above condition, which can be obtained by splitting between p_{π_i} and $p_{\pi_{(i+1)}}$, where $i = 1, 2, \ldots, (n-1)$. Thus we can find the optimal partitioning among the $(n-1)$ partitions, which maximizes $|\sum_{p_i \in I_R} p_i|^2$ for*

given size constraints. The corresponding vector partitioning of $\{I_L,\ I_R\}$ is the size-constrained optimal two-way vector partitioning. **Q.E.D.**

In this section, we convert the two-way vector partitioning problem into finding the optimal direction vector. Now we will show the algorithm to obtain the optimal two-way vector partition on the basis of the theorem 3.2, which is shown in Fig. 1.

	Optimal two-way vector partitioning
	Input : optimal direction vector $\vec{x_o}$
	: size constraints *Lower*, *Upper*
	: $Y = \{\vec{y_1}, \vec{y_2}, \ldots, \vec{y_n}\}$
	Output : optimal two-way vector partitioning
1.	$I_Y = \{p_i \mid p_i = \vec{y_i} \cdot \vec{x_o},\ \ \vec{y_i} \in Y\}$;
2.	Sort p_i's;
3.	Sorting result is $\{p_{\pi_1}, p_{\pi_2}, \ldots, p_{\pi_n}\}$ and its corresponding vectors are $\{\vec{y_{\pi_1}}, \vec{y_{\pi_2}}, \ldots, \vec{y_{\pi_n}}\}$, where $\{p_{\pi_1} \le p_{\pi_2} \le \ldots \le p_{\pi_n}\}$;
4.	Find the optimum splitting point s maximizing $\mid \sum_{i=1}^{s} p_{\pi_i} \mid^2$ or $\parallel \sum_{i=1}^{s} \vec{y_{\pi_i}} \parallel^2$, and satisfying $Lower \le s \le Upper$;
5.	$S_L = \{\vec{y_{\pi_1}}, \ldots, \vec{y_{\pi_s}}\}$, $S_R = \{\vec{y_{\pi_{(s+1)}}}, \ldots, \vec{y_{\pi_n}}\}$;

Figure 1: Optimal two-way vector partitioning algorithm

In step 4 (Fig. 1), there are two target functions to determine splitting point, s; one is $\mid \sum_{i=1}^{s} p_{\pi_i} \mid^2$, and the other is $\parallel \sum_{i=1}^{2} \vec{y_{\pi_i}} \parallel^2$. When the *optimal direction vector* is given, the algorithm can produce the optimal partition by using either function. But for a given *direction vector* obtained with heuristic, the partitioning result of the algorithm is affected by the selection of the function. As the algorithm in 1 is derived from the property 2.1, we used the function, $\parallel \sum_{i=1}^{2} \vec{y_{\pi_i}} \parallel^2$.

4 Two-way partitioning based on direction vector

In this section, we propose a heuristic to obtain high quality *direction vector* with iterative approach. The notation of two-way vector partitioning obtained with the *direction vector* $\vec{x}$ using algorithm in Fig. 1 is represented as $\{S_{L(\vec{x})}, S_{R(\vec{x})}\}$. The projection of vectors in $S_{R(\vec{x})}$ onto the direction $\vec{x}$ is represented as $I_{R(\vec{x})}$. Because the optimal two-way vector partitioning obtained with the *optimal direction vector* $\vec{x_o}$ maximizes

$$\parallel \sum_{\vec{y} \in S_{R(\vec{x_o})}} \vec{y} \parallel^2 = \mid \sum_{\vec{y} \in S_{R(\vec{x_o})}} \vec{y} \cdot \vec{x_o} \mid^2 = \mid \sum_{p_i \in I_{R(\vec{x_o})}} p_i \mid^2,$$

we try to find a *direction vector* $\vec{x}$ which maximizes the following cost function.

$$F(\vec{x}) \quad = \quad \mid \sum_{\vec{y} \in S_{R(\vec{x})}} \vec{y} \cdot \vec{x} \mid^2 = \mid \sum_{p_i \in I_{R(\vec{x})}} p_i \mid^2 \qquad (1)$$

Definition 4.1 (Refined direction vector)
For given direction vector $\vec{x}$ and corresponding vector partition $\{S_{L(\vec{x})}, S_{R(\vec{x})}\}$, we can define refined direction vector $\vec{x_r}$ whose direction is the same as $\vec{Y_{R(\vec{x})}} = \sum_{\vec{y} \in S_{R(\vec{x})}} \vec{y}$ and the length equals to unity.

As $F(\vec{x})$ is less than or equal to $F(\vec{x_r})$ which is shown in theorem 4.1, we can iteratively improve the quality of the *direction vector* $\vec{x}$.

Theorem 4.1
For a given direction vector $\vec{x}$ and the refined direction vector $\vec{x_r}$, the following relation is preserved;

$$F(\vec{x}) \le F(\vec{x_r}) \qquad (2)$$

Proof :
As a proof, we will show the following inequality;

$$F(\vec{x}) \le \mid \sum_{\vec{y} \in S_{R(\vec{x})}} \vec{y} \cdot \vec{x_r} \mid^2 \le F(\vec{x_r}).$$

For the left-hand side inequality,

$$F(\vec{x}) = \mid \vec{Y_{R(\vec{x})}} \cdot \vec{x} \mid^2 \le \parallel \vec{Y_{R(\vec{x})}} \parallel^2 = \mid \vec{Y_{R(\vec{x})}} \cdot \vec{x_r} \mid^2.$$

Next, the right-hand side inequality is proven as follows. Fig. 2 (a) and (b) show two-way vector partition using direction vector $\vec{x}$ and refined direction vector $\vec{x_r}$, respectively. We assume that all vectors are in the hashed area. In Fig. 2 (b), we can define vector set A and B whose vectors belong to $S_{L(\vec{x})}$ and $S_{R(\vec{x_r})}$, and $S_{R(\vec{x})}$ and $S_{L(\vec{x_r})}$, respectively. Then

$$\vec{Y_B} \cdot \vec{x_r} \le 0, \quad \vec{Y_A} \cdot \vec{x_r} \ge 0.$$

Thus

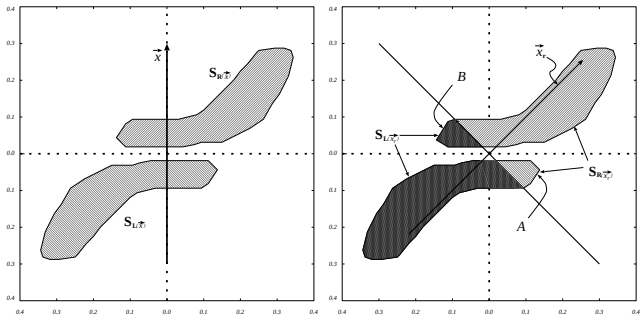
$$(\vec{Y_B} - \vec{Y_A}) \cdot \vec{x_r} \le 0.$$

By adding the same value in each side,

$$\sum_{\vec{y} \in S_{R(\vec{x_r})}} \vec{y} \cdot \vec{x_r} + (\vec{Y_B} - \vec{Y_A}) \cdot \vec{x_r} \le \sum_{\vec{y} \in S_{R(\vec{x_r})}} \vec{y} \cdot \vec{x_r}.$$

Thus,

$$\sum_{\vec{y} \in S_{R(\vec{x})}} \vec{y} \cdot \vec{x_r} \le \sum_{\vec{y} \in S_{R(\vec{x_r})}} \vec{y} \cdot \vec{x_r}.$$



(a) Two-way vector partition using *direction vector* $\vec{x}$

(b) Two-way vector partition using *refined direction vector* $\vec{x_r}$

Figure 2: Figure for proving the theorem 4.1.

As the left side is not negative, the following fact is preserved

$$|\sum_{\vec{y}\in S_{R(\vec{x})}} \vec{y}\cdot\vec{x_r}|^2 \le |\sum_{\vec{y}\in S_{R(\vec{x_r})}} \vec{y}\cdot\vec{x_r}|^2 = F(\vec{x_r}).$$

Q.E.D

	Find direction vector
	Input: vector set $\{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$.
	Output: a *direction vector* $\vec{x_b}$.
(1)	choose random *direction vector* $\vec{x}$;
(2)	best_length = 0;
(3)	**do** {
(3.1)	Partition the vectors using algorithm in Fig. 1 into $S_{L(\vec{x})}$ and $S_{R(\vec{x})}$ with *direction vector* $\vec{x}$;
(3.2)	length = $\|\sum_{\vec{y_i^d}\in S_{R(\vec{x})}} \vec{y_i^d}\cdot\vec{x}\|^2$;
(3.3)	**if** (length > best_length) new *direction vector* $\vec{x}$ whose direction is the same as $Y_{\vec{R(x)}}$; $\vec{x_b} = \vec{x}$; best_length = length; improved = yes;
(3.4)	**else** improved = no;
	} **while** (improved = yes)
(4)	**return** $\vec{x_b}$;

Figure 3: The proposed algorithm to find *direction vector*.

From the above theorem, we propose a procedure described in Fig. 3. Initial solution is randomly selected and improved through iteration. After obtained high quality direction vector, we produced two-way vector partitioning and its corresponding graph partitioning using algorithm in Fig. 1. As we approximate a given netlist as graph model to apply spectral method and only small number of eigenvectors are used in practice, there is a chance to improve the solution quality by local optimization. In this paper, we employed Fiduccia-Mattheyses(FM) algorithm as a post processing.

The time complexity of the proposed algorithm to find direction vector is $O(a(\cdot n\cdot logn + dn))$, where a is the number of iterations to find direction vector. Experimental result shows that the number of iterations is on the average 7.4, *i.e.*, the proposed algorithm is converged within 7.4 iterations in the average case.

5 Experimental result

The proposed algorithm, called PDV, was implemented in C and runs on Sparc-20. For the experiments, ACM/SIGDA benchmarks listed in Table 1 are used as test examples. Each netlist which can be represented as a hypergraph was first transformed into the graph G by replacing each p-pin net to a clique model and assigning weight of w to each edge in the clique, where $w = \frac{4}{p(p-1)}\frac{2^p-2}{2^p}$. And then, eigenvectors and eigenvalues were computed.

In this experiment, we used the smallest ten nontrivial eigenvectors, *i.e.*, $\vec{\mu_2}, \vec{\mu_3}, \ldots, \vec{\mu_{11}}$. When all the eigenvectors are not used, the choice of H can affect solution quality. While several schemes for selecting H were proposed in MELO[4], we adapted $H = \lambda_d + \lambda_2$ among them.

Table 1 shows the number of iteration of the proposed algorithm, PDV. The PDV obtains the refined direction vector with one iteration for a given initial direction vector, and uses the result as a new initial solution to obtain more refined direction vector. This procedure is repeated until the cost of the obtained direction vector is converged. The number of iterations is on the average about 7.4, *i.e.*, the proposed algorithm PDV is converged within 7.4 iterations in the average case. Table 1 also reports runtime required to produce two-way vector partitioning for a given randomly selected direction vector and to perform local optimization using FM algorithm, after the eigenvectors have been computed.

In table 2, we compared the performance of 30 runs of the proposed algorithm PDV with the previous partitioner based on spectral method. All the algorithms require each cluster to contain at least 45% of the total modules. Compared with PARABOLI[6] and MELO[4], PDV improved 7.4% and 20.5% respectively in terms of min-cut bipartitioning, and im-

Test Case	characteristics			PDV	
	#Modules	#Nets	#Pins	#iter.	runtime (second)
19ks	2844	3282	10547	8.0	1.26
prim1	833	902	2908	6.3	0.32
prim2	3014	3029	11219	6.6	1.40
test02	1663	1720	6134	7.0	0.72
test03	1607	1618	5807	7.3	0.65
test04	1515	1658	5975	6.6	0.62
test05	2595	2750	10076	6.8	1.23
test06	1752	1541	6638	6.4	0.80
balu	801	735	2697	6.2	0.28
struct	1952	1920	5471	16.3	1.08
biomed	6514	5742	21040	6.1	3.13
industry2	12637	13419	48404	5.5	7.53
SUM	37727	38316	136916	89.1	19.02

Table 1: The runtime and the number of iterations of the proposed algorithm. All the algorithms require each cluster to contain at least 45% of the total modules.

proved 10.5% and 9.7% respectively in terms of ratio-cut objective.

Test Case	PARABOLI		MELO		PDV(30)	
	cuts	RC	cuts	RC	cuts	RC
19ks			119	5.89	105	5.54
prim1	53	30.6	64	36.9	52	30.6
prim2	146	6.47	169	7.48	147	6.43
test02			106	15.4	91	14.0
test03			60	9.29	58	9.29
test04			61	10.6	50	9.93
test05			102	6.07	76	4.75
test06			90	11.7	67	9.38
balu	41	25.8	28	17.6	27	19.3
struct	40	4.20	38	4.00	36	3.78
biomed	135	1.28	115	1.08	90	0.85
industry2	193	0.49	319	0.81	211	0.63
SUM	608	68.84	1271	126.82	1010	114.48

Table 2: Comparison with previous partitioners based on spectral method in terms of min-cut bipartitioning and ratio-cut partitioning. The 'cuts' and 'RC' mean cutsize and ratio-cut, respectively. All the algorithms require each cluster to contain at least 45% of the total modules. The unavailable data are left as blanks.

Table 3 shows the comparison of the proposed PDV with previous two-way partitioners based on iterative approach. We ran FM[2] and Window[7] 100 times, and ran PDV 30 times. Compared with FM and Window, PDV yields 40% and 9.4% improvement in terms of min-cut bisection, respectively.

6 Conclusion

In this paper, we show a method to obtain optimal two-way vector partitioning using the *optimal direction vector*. From the theoretical background, we proposed a method to obtain high quality direction vector with iterative approach. In the experimental results, we showed the efficiency of the proposed algorithm.

Test Case	FM(100)		Window(100)		PDV(30)	
	Best	Avg.	Best	Avg.	Best	Avg.
19ks	142	195	124	153	112	131
prim1	57	82	54	73	53	70
prim2	236	309	183	263	146	193
test02	115	182	94	113	97	105
test03	72	127	63	91	60	66
test04	86	143	56	86	57	75
test05	97	190	75	117	80	98
test06	71	94	66	83	72	87
balu	36	70			31	43
struct	45	59			36	40
biomed	83	166	143	201	90	130
industry2	758	1244	264	503	250	265
SUM	1798	2861	1122	1683	1084	1303

Table 3: Comparison of the proposed PDV procedure with the previous move-based approaches in terms of min-cut bisection. There are two columns for each scheme, showing the best and average out of 100 solutions (PDV has only 30 solutions). The unavailable data are left as blanks.

Acknowledgment

We are grateful to C.J.Alpert for providing us with the MELO code, DP-RP code, benchmark examples and eigenvectors of the examples used in their work.

References

[1] B.W.Kernighan and S.Lin, "An efficient heuristic procedure for partitioning graphs," *Bell syst. Tech. Journal*, vol. 47, pp. 291–307, Feb. 1970.

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

[3] S.Dutt and W.Deng, "A probability-based approach to vlsi partitioning," in *ACM/IEEE Design Automation Conference*, pp. 100–105, 1996.

[4] C.J.Alpert and S.Z.Yao, "Spectral partitioning:the more eigenvectors, the better," in *ACM/IEEE Design Automation Conference*, pp. 195–200, 1995.

[5] C.J.Alpert and A.B.Kahng, "Recent directions in netlist partitioning:survey," *Integration, the VLSI journal*, vol. 19, no. 1-2, pp. 1–81, 1995.

[6] B.M.Riess, K.Doll and F.M.Johannes, "Partitioning very large circuits using analytical placement techniques," in *ACM/IEEE Design Automation Conference*, pp. 743–748, 1993.

[7] C.J.Alpert and A.B.Kahng, "A general framework for vertex ordering with application to netlist clustering," in *IEEE/ACM International Conference on Computer-Aided Design*, pp. 63–67, 1994.