A Clustering Based Linear Ordering Algorithm for K-Way Spectral Partitioning

Shiuann-Shiuh Lin Wen-Hsin Chen Wen-Wei Lin† TingTing Hwang
Department of Computer Science
†Department of Mathematics
National Tsing Hua University, HsinChu, Taiwan 30043

Abstract

The spectral method can lead to a high quality of multi-way partition due to its ability to capture global netlist information. For spectral partition, n netlist modules are mapped to n points in d-dimensional space, and then a linear ordering of these n modules is constructed to be used as a basis for partitioning. In this paper, we propose two clustering based linear ordering algorithms taking into consideration the objective function presented by [1].

1 Introduction

A general k-way partitioning problem can be defined as follows: Given n modules $V = \{v_1, v_2, ..., v_n\}$, construct a k-way partition P^k in which V is divided into k disjoint clusters $S_1, S_2, ..., S_k$ with $(S_1 \cup S_2 \cup ... \cup S_k) = V$ and $S_i \cap S_j = \emptyset$ for $1 \leq i, j \leq k$, and a user-defined object function $F(P^k)$ is minimized.

In this paper, we consider the partition problem that divides the netlist modules into clusters subject to size balance constraint while minimizing cut nets among clusters. Therefore, we will adopt the *scaled cost* [5] as our objective function:

$$F_{scaled}(P^k) = \frac{1}{n(k-1)} \sum_{l=1}^{k} \frac{E_l}{|S_l|}.$$

Here E_l is the number of signal nets crossing the boundary of S_l, and $|S_l|$ is the number of modules in cluster S_l.

A circuit netlist of n nodes can be represented as a hypergraph of n vertices. This hypergraph can be transformed into a graph using a clique model by adding weight $w = \frac{4}{p(p-1)} \frac{2^p-2}{2^p}$ to each edge in a p-pin net [4]. The transformed weighted graph $G(V, E)$ consists of a vertex set $V = \{v_1, v_2, ..., v_n\}$ and an $n \times n$ symmetric adjacency matrix $A = (a_{ij})$. Here $a_{ij} = 0$ if there is no edge between v_i and v_j and $a_{ij} > 0$ is the weight of (v_i, v_j) in E. An $n \times n$ degree matrix $D = (d_{ij})$ is also defined, where $d_{ii} = \sum_{j=1}^{n} a_{ij}$ and $d_{ij} = 0$ if $i \neq j$. Then, the Laplacian matrix of the graph G is defined as $Q = D - A$ (ie. $q_{ii} = 0$ for $1 \leq i \leq n$, $q_{ij} = -a_{ij}$ for $1 \leq i, j \leq n$ and $i \neq j$).

The spectral method uses the Laplacian matrix to construct the geometry representation of a graph. Many researches have proposed heuristics to solve the spectral partitioning problem [2, 4, 6, 7, 9]. Among them, Alpert [1] opened a new door to spectral partitioning. He proved that the min-cut graph partition problem $P^k = \{S_1, S_2, ..., S_k\}$ is exactly reduced to the *max-sum vector partition* problem.

Given n vectors $\{\vec{y_1}, \vec{y_2}, ..., \vec{y_n}\}$, a *k-way vector partition* $G^k = \{C_1, C_2, ..., C_k\}$ is a partition of vectors that each $\vec{y_i}$ belongs to exactly only one C_i. Furthermore, a *max-sum vector partition* problem is: Given subset cardinality bounds L_i and U_i, find a vector partition G^k such that $L_i \leq |C_i| \leq U_i$ for each $1 \leq i \leq k$ and maximize

$$\sum_{i=1}^{k} \left\| \vec{Y}_i \right\|^2 \quad , where \quad \vec{Y}_i = \sum_{\vec{y} \in C_i} \vec{y}. \tag{1}$$

To establish the relations between max-sum vector partitioning and min-cut graph partitioning problem, Alpert indicated that the "proper" spectral embedding should scale each eigenvector by a function of eigenvalues, and $\vec{\mu_i}$ is more significant than any other eigenvectors $\vec{\mu_j}$ when $i \leq j$, where λ_n represents the n^{th} eigenvalue of Laplacian matrix Q. The $n \times d$ scaled eigenvector matrix V_d, where n is the number of netlist vertices, and d is the number of eigenvectors used, is defined as follows with $H \geq \lambda_d$ being some constant and $\vec{\mu_i}$ a column vector:

$$V_d = \left[\vec{\mu}_1 \sqrt{H - \lambda_1}, \vec{\mu}_2 \sqrt{H - \lambda_2}, ..., \vec{\mu}_d \sqrt{H - \lambda_d} \right]. \tag{2}$$

We can then map n netlist modules $\{v_1, v_2, ..., v_n\}$ to n points $\{\vec{y_1}, \vec{y_2}, ..., \vec{y_n}\}$ in d-dimensional space where $\vec{y_i}$ denote the i^{th} row of V_d (we discard the first column of V_d, since $\vec{\mu}_1 = \left[\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, ..., \frac{1}{\sqrt{n}} \right]^t$ has no effect to partition).

The mathematical relation between min-cut graph partition and max-sum vector partition can then be established as follows:

$$\sum_{l=1}^{k} \frac{\left\| \vec{Y}_l \right\|^2}{|C_l|} = kH - \sum_{l=1}^{k} \frac{E_l}{|S_l|} \tag{3}$$

From Equation (3), it is obvious that if we maximize $\sum_{l=1}^{k} \frac{\left\| \vec{Y}_l \right\|^2}{|C_l|}$, then $\sum_{l=1}^{k} \frac{E_l}{|S_l|}$ would be minimized, i.e., $F_{scaled}(P^k)$ is minimized.

The spectral partitioning problem is usually not solved directly. Instead, a two-phase approach was taken [1] [2]. Given a geometric embedding which is obtained from the Laplacian matrix Q representing a

0-7803-5012-X /99/$10.00 ©1999 IEEE.

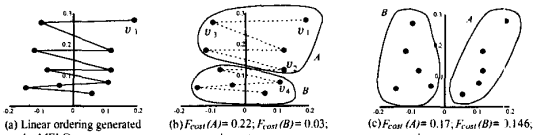

Figure 1: The Weakness of MELO

given netlist. In first phase, a sequential list of the netlist modules (linear ordering) is derived in which stronger connected netlist modules will be put close to each other.

Then in second phase, in order to consider area balance, the generated linear ordering list is applied by DP-RP (Dynamic Programming-Restricted Partition) to derive the optimum k-way problem solution in terms of the scaled cost. Since the DP-RP can optimally cut the linear list, to improve the partitioning solution, most researches [1, 2, 5] focus on improving the result of linear ordering.

2 Motivation

Based on the concept of Equation (3), Alpert proposed a greedy linear ordering algorithm "MELO" [1] to maximize

$$F_{cost}(P_v^k) = \sum_{l=1}^{k} \frac{\left\| \vec{Y}_l \right\|^2}{|C_l|}.$$

and is explained briefly as follows. Initially, n netlist modules $\{v_1, v_2, ..., v_n\}$ are mapped to n vectors $\{\vec{y_1}, \vec{y_2}, ..., \vec{y_n}\}$ in d-dimensional space by spectral method and two vector sets S_0, S_1 are created. "MELO" initializes S_1 to be empty and S_0 to be a vector set of n vectors $\{\vec{y_1}, \vec{y_2}, ..., \vec{y_n}\}$. Iteratively, MELO selects a vector $\vec{y_i} \in S_0$ that maximizes $\left\| \vec{y_i} + \sum_{\vec{y_j} \in S_1} \vec{y_j} \right\|$, adds it to S_1, and then deletes it from S_0. If $\vec{y_i}$ is the j^{th} vector added to C_1, then v_i is the j^{th} module in the linear ordering.

Intuitively, MELO derives a good solution to maximize $F_{cost}(P_v^k)$. However, MELO does not perform well in many cases. Let's take an example to explain the weakness of MELO [5]. In Figure 1, we have ten vectors plotted in 2-d space (each vector represents one netlist module) and we would like to partition them into two clusters. The linear ordering derived by MELO is shown in Figure 1(a), in which v_1 is the first vector selected. If DP-RP is applied to this ordered list, the bi-partition solution is shown in Figure 1(b) in which v_2, v_4 are put away from each other and v_1, v_3 are put next to each other.

However, if we initially divide the ten vectors into two clusters as shown in Figure 1(c) according to the density distribution of the points, and then order the five points in each cluster separately, no matter what order principle for these points is adopted, the points belong to the same coordinate will be put continuously in the linear ordering list. In such a case, after applying DP-RP, we can derive the optimal bi-partition solution.

Consequently, our proposal is to cluster the closed separated points in d-dimensional space together first, and then order the points within each individual cluster. In such a way, we will have a better chance to derive an optimal solution for k-way partition.

3 Clustering Based Linear Ordering Generation

Our linear ordering algorithm consists of three steps: cluster formation by algorithms KC or MPC, cluster order, and module order. In the cluster formation step, clusters are found such that modules in the same cluster are closely separated points in d-dimensional space. In the cluster order step, the formed clusters are linear ordered. Finally, in the module order step, modules are ordered within each individual cluster to obtain the final linear ordering list. The linear ordering list generated can then be applied by DP-RP to derive the optimal k-way partition solution.

3.1 Cluster Formation

In this section, we will present two algorithms to perform cluster formation. In section 3.1.1, the distance among modules (vertices) is considered as a cost function to find clusters. In section 3.1.2, the technique used in image partition is utilized.

3.1.1 KC Algorithm

Our goal of cluster formation is to find the closest points in geometry. Intuitively, the distance of two points can be used as a guidance. *KC (K-Center) algorithm* is a cluster formation algorithm which divides a set of points into clusters so as to minimize the maximum intercluster distance[4][8]. Therefore, our first attempt is to use K-Center (KC) algorithm to perform the "cluster" process. The objective function of KC algorithm is:

$$Minimize\{ \max_{v_1, v_2 \in C_i} \{distance(v_1, v_2)\}\}$$

$$distance(v_1, v_2) = \|\vec{y_1} - \vec{y_2}\| \ \ where \ v_1 = \vec{y_1}, v_2 = \vec{y_2}.$$

KC algorithm is explained as follows. KC algorithm consists of an initialization phase and $k-1$ "expanding" phases. In the initialization phase, all points $\{\vec{y_1}, \vec{y_2}, ..., \vec{y_n}\}$ are assigned to cluster C_1 and the point with the largest magnitude of vector is labeled as $seed_1$ for C_1. In the j^{th} "expanding" iteration, $seed_j$ is chosen from points in $C_1, C_2, ..., C_{j-1}$. $Seed_j$ is chosen as follows: First, point v_i which has the largest distance from its seed, $seed_i$, of cluster C_i is computed for each cluster. Then $seed_j$ is chosen as the largest value from v_i, for $1 \le i \le j-1$. After $seed_j$ is found, if the value of $distance(v, seed_i)$ is greater than the value of $distance(v, seed_j)$, move v to the newly formed cluster C_j for each cluster C_i, $1 \le i \le j-1$, and for each point, $v \in C_i$.

At the end of KC algorithm, for any point v, the distance between v in cluster C_l and the seed, $seed_l$, of the cluster it belongs to is less than the distance between v and the seed of any other clusters. (ie, $distance(v, seed_l) \le distance(v, seed_i)$ for $1 \le i \le k$ and $i \ne v$). Hence, the maximum intercluster distance is minimized. Figure 2 shows an example in which we divide the mapped points in 2-d space into 3 clusters ($k = 3$) by KC algorithm.

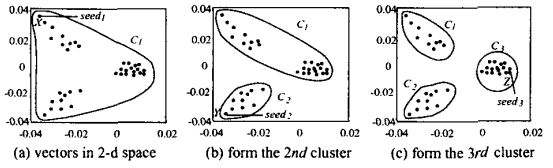

Figure 2: Process for 3-way KC Partition

3.1.2 MPC Algorithm

The *Moment-Preserving Cluster Algorithm* (MPC) [3] is initially applied in the field of image partition to find the clusters of the given patterns. Given n patterns in multi-dimensional space, MPC can classify the n patterns into approximate clusters, so that the patterns in the same cluster are densely populated and the region between two clusters is sparsely populated [3]. Since netlist modules can be mapped to points in multi-dimensional space by spectral method, we can view these points as patterns in the image. Therefore our second attempt is to implement this algorithm as another way of "cluster formation".

MPC algorithm can be explained as follows. For these n points in d-dimensional space, we treat each point as a column vector $\vec{y}_i$ for $1 \leq i \leq n$, and $Y = \{\vec{y}_1, \vec{y}_2, ..., \vec{y}_n\}$. First, we perform the Karhunen-Loeve expansion to find the covariance matrix $\text{Cov}(Y) = \sum_{i=1}^{n}\{(\vec{y}_i - \vec{\mu})(\vec{y}_i - \vec{\mu})^t\}$, where $\vec{\mu}$ is the mean of these n column vectors. Then we compute the eigenvectors $\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_d$ and the corresponding eigenvalues $\lambda_1, \lambda_2, ..., \lambda_d$ (with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$) of the covariance matrix.

Second, by using $\vec{\mu}$ as the original point and the computed eigenvector $\vec{\mu}_j (1 \leq j \leq d)$ as axis, we can choose an appropriate length related to λ_j at each axis and obtain a hypercube in which it contains most of the n points. This hypercube is defined as Central Region (CR) of the n points. Statistically, 70% to 80% of the n points are populated in CR.

Third, the points in CR are divided into m subregions by projecting the points onto the eigenvector $\vec{\mu}_1$ (the one whose corresponding eigenvalue is largest) first. Among these m subregions, if a sparsity subregion (i.e., the number of the projected points in this subregion is too few) is detected, all points are divided into subclasses according to the sparsity of subregions. Then, the newly formed clusters are viewed as new independent vector sets; $\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_d$ and CR corresponding to the new covariance matrix are calculated for each new class separately. Each new class can use the above procedure (Projection, Detection for the sparsity subregion) to find partition recursively. On the other hand, if no sparsity subregion is found, we repeat on projecting the points in CR to the next eigenvector whose corresponding eigenvalue is the second largest until we find at least one sparsity subregion or all d eigenvectors are used. MPC algorithm ends when no more new partition can be generated. Figure 3 shows an example in which the vertices are mapped to 2-d space.

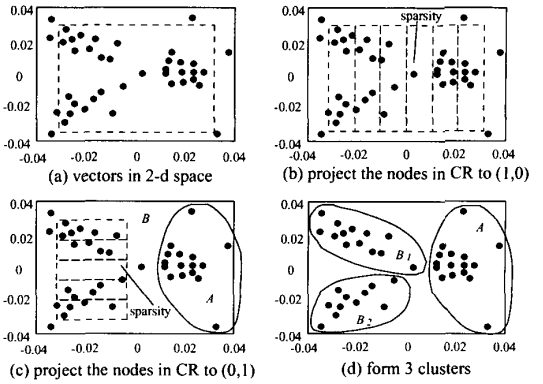

Figure 3: Process of 3-way MPC Partition

3.2 Cluster Order

Now we have m clusters, $\{C_1, C_2, ..., C_m\}$, produced in the "cluster formation" step. In the cluster-order phase, we want to find a sequential ordered cluster list $\{C_{\pi_1}, C_{\pi_2}, ..., C_{\pi_m}\}$ of the m clusters such that our cost function: $\sum_{l=1}^{k} \frac{\|\vec{Y}_{\pi_l}\|^2}{|C_{\pi_l}|}$ is maximized. In this step, we use the greedy "cluster order" algorithm proposed in CBLO [5].

3.3 Module Order

After performing "cluster formation" and "cluster order" steps, we have m ordered clusters $\{C_{\pi_1}, C_{\pi_2}, ..., C_{\pi_m}\}$ where each cluster contains closed separated modules in d-dimensional space. The final step is to determine the module order within each cluster. In this case, each cluster is viewed as an independent vector set. Algorithm MELO [1] is used to order modules within each cluster.

4 Experimental Results

We have implemented the two clustering algorithms in C programming language, called KC and MPC respectively, and tested our algorithms on a set of ACM/SIGDA benchmarks. We compared our results to those produced by MELO and CBLO [5] which is a clustering based partition algorithm for multi-way partitioning with the objective of minimizing the scaled cost function.

In our experiment, for each benchmark, first we used d eigenvectors ($2 \leq d \leq 11$, $\vec{\mu}_1$ is discarded) of the Laplacian matrix to construct the mapped points in geometry. Then our proposed clustering based linear ordering algorithm is used to derive the linear ordering of the modules. Finally, the obtained linear ordering is applied by DP-RP to derive the optimum partition solution. For comparison reason, we also chose the same H as used in MELO [1].

In Table 1, we select the best k-way results for each linear ordering ($2 \leq d \leq 11$) after running DP-RP for MELO, KC, and MPC. The column under "Number of cluster" is the results for k-way partitioning, for k= 2,3,...,10. As expected, the scaled cost shows that as the size of k increases, our proposed heuristic performs better. On an average, algorithms KC,

Table 1: Scaled Cost ($\times 10^{-5}$) Comparisons for KC,MPC,MELO,CBLO

Test Case	Algorithms	Number of Clusters, k									sum
		2	3	4	5	6	7	8	9	10	
19ks	MELO	4.8	5.0	5.3	6.1	6.8	7.9	8.3	9.1	9.9	63.2
	CBLO	4.8	5.1	5.5	5.9	6.1	6.3	6.9	7.9	8.9	57.4
	KC	4.6	4.8	5.0	5.5	6.0	6.6	7.3	7.7	8.3	55.8
	MPC	4.6	4.7	5.0	5.4	6.0	6.7	7.0	7.6	8.1	55.1
prim1	MELO	13.4	17.1	22.5	29.4	34.0	37.0	39.7	41.9	44.6	279.6
	CBLO	13.4	14.0	16.2	20.7	23.9	26.1	28.4	30.2	32.8	205.7
	KC	13.3	13.9	16.1	20.6	23.6	26.1	29.4	32.1	35.7	210.8
	MPC	13.5	14.6	16.6	21.4	24.8	28.3	32.0	34.8	37.1	223.1
prim2	MELO	4.7	6.8	8.0	9.2	10.1	11.2	12.0	12.7	13.7	88.4
	CBLO	4.6	6.4	7.7	8.4	9.1	9.7	10.0	10.9	11.5	78.3
	KC	4.6	6.5	7.4	7.9	8.4	8.9	9.7	10.3	11.0	74.7
	MPC	5.2	6.7	8.0	8.6	9.0	9.6	9.9	10.5	11.0	78.5
test02	MELO	8.1	10.7	12.4	13.9	15.4	17.0	18.5	19.9	21.1	137.0
	CBLO	8.1	10.5	11.7	13.0	14.5	16.1	17.6	18.8	20.0	130.3
	KC	8.0	10.4	11.6	12.4	14.0	15.4	17.0	18.4	19.6	126.8
	MPC	8.3	11.1	12.0	13.4	14.3	15.7	16.8	17.9	18.7	128.2
test03	MELO	9.3	11.6	12.5	13.7	14.6	15.3	16.7	17.6	19.0	130.3
	CBLO	9.2	10.3	11.3	12.7	14.0	14.9	15.8	16.5	17.4	122.1
	KC	9.2	11.1	11.8	13.1	14.0	14.7	15.4	17.0	17.8	124.1
	MPC	9.1	10.8	12.2	13.3	14.1	14.8	15.6	16.4	17.4	123.7
test04	MELO	5.8	6.8	8.2	9.3	10.0	10.8	11.5	12.3	13.2	87.9
	CBLO	5.7	6.5	7.4	8.7	9.5	10.2	11.2	12.5	13.6	85.3
	KC	5.7	6.6	8.4	9.5	10.2	11.1	11.7	12.5	13.5	89.2
	MPC	5.8	7.4	8.8	9.5	10.5	11.1	11.5	12.7	13.1	90.4
test05	MELO	3.1	4.4	4.9	5.5	5.8	6.1	6.5	7.0	7.4	50.7
	CBLO	3.1	4.0	4.4	4.7	5.2	5.7	6.0	6.3	6.6	46.0
	KC	2.9	3.8	4.1	4.7	5.4	5.8	6.1	6.4	6.8	46.0
	MPC	3.0	3.8	4.1	4.7	5.0	5.8	6.0	6.4	6.5	45.3
test06	MELO	8.8	9.5	11.3	13.5	14.7	16.7	18.5	20.2	21.3	134.5
	CBLO	8.2	9.9	11.6	12.4	13.5	15.1	16.8	18.3	19.4	125.2
	KC	7.9	9.9	11.6	12.5	13.5	15.3	16.8	17.8	18.8	124.1
	MPC	8.2	10.4	11.9	13.0	13.8	14.5	15.5	16.6	17.6	121.5
balu	MELO	17.6	24.4	32.3	36.7	40.0	43.2	46.5	50.1	54.0	344.8
	CBLO	17.7	25.6	28.1	34.3	37.7	40.4	42.3	44.4	49.7	320.2
	KC	16.8	22.6	24.5	29.9	33.3	37.6	40.2	42.9	45.3	293.1
	MPC	15.0	21.3	22.8	26.9	33.1	38.2	40.3	41.0	42.8	281.4
struct	MELO	4.3	5.5	6.5	7.6	8.5	9.8	10.9	12.0	12.9	78.0
	CBLO	4.7	5.9	7.4	8.0	8.6	9.3	10.0	11.0	11.8	76.7
	KC	4.2	6.0	7.5	8.2	8.8	9.4	10.2	10.9	11.5	76.7
	MPC	4.1	6.0	7.5	8.2	8.6	9.5	10.4	11.0	11.7	77.0
biomed	MELO	0.6	0.9	1.1	1.2	1.3	1.5	1.6	1.7	1.9	11.8
	CBLO	0.6	0.9	0.9	1.1	1.2	1.3	1.4	1.5	1.6	10.5
	KC	0.6	0.8	0.9	1.1	1.2	1.3	1.4	1.5	1.6	10.4
	MPC	0.6	0.8	0.9	1.1	1.2	1.3	1.4	1.5	1.6	10.4
sum	MELO	80.5	102.7	126.0	146.1	161.2	176.5	190.7	204.5	219.0	1406.2
	CBLO	80.1	99.1	112.2	129.9	143.3	155.1	166.4	178.3	193.3	1257.7
	KC	77.8	96.4	108.9	125.4	138.4	152.2	165.2	177.5	189.9	1231.7
	MPC	77.4	97.6	109.8	125.5	140.4	155.5	166.4	176.4	185.6	1234.6

MPC are 12.5% and 12.3% better than MELO, and 2.1% and 2.0% better than CBLO for k-way partitioning.

The results of KC and MPC in Table 1 shows that both methods achieve similar improvement for k-way partition. KC is more efficient than MPC in terms of CPU time. In our experiment, we observed that the points in d-dimensional of each benchmark are too close to be partitioned, which makes it difficult to apply MPC. We believe that if the points are well-separated (i.e., points are not too close to be divided properly) in d-dimensional space, MPC will produce good results. On the other hand, if the points in d-dimensional are indeed close together, the simple KC algorithm is sufficient to be used in the "cluster formation" step.

5 Conclusions

In this paper, we have presented two clustering based linear ordering algorithm for k-way spectral partitioning taking into consideration the new objective function presented in [1]. The first clustering algorithm uses the geometric distance among vertices as cost function to find clusters and the second clustering algorithm utilizes the moment-preserving clustering algorithm initially applied in the field of image partition. The experimental results show that our methods are indeed effective.

References

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

[2] C. J. Alpert and A. B. Kahng, "Multi-Way Partitioning Via Spacefilling Curves and Dynamic Programming," *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp.652-657.

[3] Song-Tyang Liu and Wen-Hsiang Tsai, "Moment-Preserving Clustering," *Proc. Pattern Recognition,* vol.22, No.4, 1989, pp.433-447.

[4] 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.

[5] Kwang-Su Seong and Chong-Min Kyung, "CBLO: A Clustering Based Linear Ordering for Netlist Partitioning," *Proc. ASP-DAC,* 1997.

[6] 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.

[7] J. Frankle and R. M. Karp, "Circuit Placements and Cost Bounds by Eigenvector Decomposition," *IEEE Conf. Computer Aided Design,* 1986, pp.414-417.

[8] T. F. Gonzalez, "Clustering to Minimizing the Maximum Intercluster Distance," *in Theoretical Computer Science* 38, 1985, pp.293-306.

[9] L. Hagen and A. B. Kahng., "New Spectral Methods for Ratio-Cut Partitioning and Clustering," *IEEE Trans. on CAD* 11(9), Sept. 1992, pp.1074-1085.