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 objective 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 objective function:

\[
F_{\text{scaled}}(P_k) = \frac{1}{n(k-1)} \sum_{i=1}^{k} |S_i|.
\]

Here \( E_i \) is the number of signal nets crossing the boundary of \( S_i \), and \( |S_i| \) is the number of modules in cluster \( S_i \).

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 \( w = \frac{a_{ij}}{E(V,E)} \) to each edge in a 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 \( Q = D - A \) (i.e., \( q_{ij} = 0 \) 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 \( \{y_1, y_2, ..., y_n\} \), a k-way vector partition \( G^k = \{C_1, C_2, ..., C_k\} \) is a partition of vectors that each \( y_i \) 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

\[
\sum_{i=1}^{k} \|\hat{y}_i\|^2, \quad \text{where} \quad \hat{y}_i = \sum_{y \in C_i} y.
\]

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 \( \mu_i \) is more significant than any other eigenvectors \( \mu_j \) when \( i \leq j \), where \( \lambda_n \) 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 \( \mu_i \) a column vector:

\[
V_d = \begin{bmatrix} \mu_1 \sqrt{H - \lambda_1} & \mu_2 \sqrt{H - \lambda_2} & \cdots & \mu_d \sqrt{H - \lambda_d} \end{bmatrix}.
\]

We can then map \( n \) netlist modules \( \{v_1, v_2, ..., v_n\} \) to \( n \) points \( \{y_1, y_2, ..., y_n\} \) in \( d \)-dimensional space where \( \hat{y}_i \) denote the \( i^{th} \) row of \( V_d \) (we discard the first column of \( V_d \), since \( \mu_1 \) is some vector \( \frac{1}{\sqrt{\lambda_1}} \ldots \frac{1}{\sqrt{\lambda_d}} \) has no effect to partition).

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

\[
\sum_{i=1}^{k} \frac{\|\hat{y}_i\|^2}{|C_i|} = kH - \sum_{i=1}^{k} \frac{E_i}{|C_i|}
\]

From Equation (3), it is obvious that if we maximize \( \sum_{i=1}^{k} \frac{\|\hat{y}_i\|^2}{|C_i|} \), then \( \sum_{i=1}^{k} \frac{E_i}{|C_i|} \) would be minimised, i.e., \( F_{\text{scaled}}(P_k) \) is minimised.

The spectral partitioning problem is usually 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
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

$$F_{\text{cost}}(P^*_n) = \sum_{l=1}^{k} \| \tilde{Y}_l \|_{C_l}.$$ 

and is explained briefly as follows. Initially, n netlist modules \{v1, v2, ..., v\} are mapped to n vectors \{y1, y2, ..., y\} 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 \{y1, y2, ..., y\}. Iteratively, MELO selects a vector \(\tilde{y}_i \in S_0\) that maximizes \(\|\tilde{y}_i + \sum_{j \in E \in S_F} \tilde{y}_j\|\), adds it to \(S_1\), and then deletes it from \(S_0\). If \(\tilde{y}_i\) is the \(j^{th}\) vector added to \(S_1\), then \(v_i\) is the \(j^{th}\) module in the linear ordering.

Intuitively, MELO derives a good solution to maximize \(F_{\text{cost}}(P^*_n)\). 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:

$$\text{Minimize} \max \left\{ \text{distance}(v_1, v_2) \right\}$$

where

$$\text{distance}(v_1, v_2) = \| y_1 - y_2 \|$$

KC algorithm is explained as follows. KC algorithm consists of an initialization phase and \(k - 1\) "expanding" phases. In the initialization phase, all points \{y1, y2, ..., y\} are assigned to cluster \(C_1\) and the point with the largest magnitude of vector is labeled as seed1 for \(C_1\). In the \(j^{th}\) "expanding" iteration, seedj is chosen from points in \(C_1, C_2, ..., C_{j-1}\), where seedj is chosen as follows: First, point \(v_1\), which has the largest distance from its seed, seed1, of cluster \(C_1\) is computed for each cluster. Then seedj is chosen as the largest value from \(v_i\) for \(1 \leq i \leq j - 1\). After seedj is found, if the value of distance(\(v_i, \text{seed}_j\)) is greater than the value of distance(\(v, \text{seed}_j\)), move \(v\) to the newly formed cluster \(C_j\) for each cluster \(C_i\), \(1 \leq i \leq j - 1\), and for each point, \(v \in C_i\).

At the end of KC algorithm, for any point \(v\), the distance between \(v\) in cluster \(C_1\) and the seed, seed1, of the cluster it belongs to is less than the distance between \(v\) and the seed of any other clusters. (i.e., distance(\(v, \text{seed}_1\)) \leq distance(\(v, \text{seed}_j\)) for \(1 \leq i < k\) and \(i \neq j\)). 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.
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 $y_i$ for $1 \leq i \leq n$, and $Y = \{y_1, y_2, ..., y_n\}$. First, we perform the Karhunen-Loeve expansion to find the covariance matrix $\text{Cov}(Y) = \sum_{i=1}^{n} (y_i - \mu)(y_i - \mu)^T$, where $\mu$ is the mean of these n column vectors. Then we compute the eigenvectors $\mu_1, \mu_2, ..., \mu_d$ and the corresponding eigenvalues $\lambda_1, \lambda_2, ..., \lambda_d$.

Second, by using $\mu$ as the original point and the computed eigenvector $\mu_j (1 \leq j \leq d)$ as axis, we can choose an appropriate length related to $\lambda_j$ 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 $\mu_j$ (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; $\mu_1, \mu_2, ..., \mu_d$ 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.

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 $\{C_{e1}, C_{e2}, ..., C_{em}\}$ of the m clusters such that our cost function:

$$\sum_{i=1}^{k} \| \mathbf{y}_i - \mu_i \|^2 / |C_{ei}|$$

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 $\{C_{e1}, C_{e2}, ..., C_{em}\}$ 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 $2 \leq d \leq 11$, $\mu_1$ 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,
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 partitioning. KC is more efficient than MPC in terms of CPU time. In our experiment, we observed that the points in d-dimensional space 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 algorithms, KBLO and KBLO-P. 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