Multi-Way Partitioning Via Spacefilling Curves and Dynamic Programming*

C. J. Alpert and A. B. Kahng

UCLA Computer Science Department, Los Angeles, CA 90024-1596

Abstract

Spectral geometric embeddings of a circuit netlist can lead to fast, high quality multi-way partitioning solutions. Furthermore, it has been shown that d-dimensional spectral embeddings ($d > 1$) are a more powerful tool than single-eigenvector embeddings ($d = 1$) for multi-way partitioning [2] [4]. However, previous methods cannot fully utilize information from the spectral embedding while optimizing netlist-dependent objectives. This work introduces a new multi-way circuit partitioning algorithm called DP-RP. We begin with a d-dimensional spectral embedding from which a 1-dimensional ordering of the modules is obtained using a *spacefilling curve*. The 1-dimensional ordering retains useful information from the multi-dimensional embedding while allowing application of efficient algorithms. We show that for a new *Restricted Partitioning* formulation, dynamic programming efficiently finds *optimal* solutions in terms of Scaled Cost [4] and can transparently handle user-specified cluster size constraints. For 2-way ratio cut partitioning, DP-RP yields an *average* of 45% improvement over KP [4] and EIG1 [6] and 48% improvement over KC [2].

1 Introduction

Systems with several million transistors entail problem complexities that are unmanageable for existing logic- and physical-level design tools. Thus, partitioning is used to divide the system into smaller, more manageable components. Previous work has focused on 2-way partitioning algorithms that in practice are recursively applied to generate k-way partitionings. Since the top-down approach can lead to unnatural partitioning solutions, our work seeks to reveal the natural circuit structure via a non-hierarchical decomposition into k subcircuits with minimum interconnectivity between the subcircuits:

General k-Way Partitioning: Given a circuit netlist hypergraph $H = (V, E_H)$ with n modules $V = \{v_1, v_2, \ldots, v_n\}$, and a value $2 \leq k \leq n$, construct a *k-way partitioning*, P^k, that divides V into k disjoint *clusters* $C_1, C_2, \ldots, C_k$ to optimize a given objective function $f(P^k)$.

Because cluster sizes are generally not known in advance, we require a multi-way partitioning measure which can account for both cut nets and size balance among the clusters. The *Scaled Cost* objective [4] captures these requirements and is a multi-way generalization of the *ratio cut* objective [14]:

Scaled Cost: Find $P^k = \{C_1, C_2, \ldots, C_k\}$ that minimizes

$$f(P^k) = \frac{1}{n(k-1)} \sum_{i=1}^{k} \frac{|E_i|}{|C_i|}$$

where $E_i \subseteq E_H$ is the set of signal nets crossing the boundary of cluster C_i. In what follows, we assume f is the Scaled Cost objective. Minimizing Scaled Cost is shown NP-complete by restriction to minimum ratio cut.

1.1 Previous Work

Previous work on 2-way partitioning has centered on the minimum *bisection* and the minimum *ratio cut* objectives, and various greedy or hill-climbing approaches having been proposed [9]. Approaches to multi-way partitioning have involved seeded epitaxial growth, extensions of the Fiduccia-Mattheyses [5] iterative bipartitioning algorithm [11], a primal-dual iteration motivated by a generalization of the ratio cut metric [15], and spectral approaches [6].

Other multi-way partitioning approaches extend the well-established spectral method. Hall [7] showed that the eigenvector corresponding to the second smallest eigenvalue of the netlist Laplacian[1] minimizes a squared-wirelength objective. Hall's work [7] discusses the construction of a d-dimensional *spectral em-*

*Partial support for this work was provided by a Department of Defense Graduate Fellowship, by NSF Young Investigator Award MIP-9257982, and by Cadence Design Systems and Zycad Corporation under the State of California MICRO program. ABK was also supported by NSF MIP-9117328 during a Spring 1993 sabbatical visit to UC Berkeley.

[1] Given an edge-weighted graph representation of the netlist, $G(V, E_G)$, we construct the $n \times n$ *adjacency matrix* $A = A(G)$ in which A_{ij} has weight $e(v_i, v_j) \in E_G$ (by convention $A_{ii} = 0$ for all $i = 1, \ldots, n$). If $deg(v_i)$ denotes the degree of node v_i (i.e., the sum of the weights of all edges incident to v_i), we obtain the $n \times n$ diagonal *degree matrix* D defined by $D_{ii} = deg(v_i)$. The *Laplacian* of the netlist graph is given by $Q = D - A$.

bedding of a graph, based on the d eigenvectors that correspond to the smallest eigenvalues of the Laplacian. The embedding is constructed by letting the i^{th} components of the d eigenvectors form the coordinates in $\Re^d$ of module $v_i \in V$. Hall suggested partitioning the spectral geometric embedding as a graph partitioning heuristic; extensions to netlist hypergraphs are accomplished using a clique net model. Intuitively, since all d eigenvectors are good solutions for the squared-wirelength objective, the d-dimensional embedding has strong "distance-preserving" properties. In other words, the embedding reflects "distance" and "separation" between netlist modules, such that two modules which are strongly (weakly) connected in the netlist should map to closely (widely) separated points in the geometry.

The spectral approach is computationally efficient: eigenvectors of the Laplacian may be computed using readily available Lanczos codes. For example, Hagen and Kahng [6] based their work on the code reported by Pothen et al. [10], while others use the more widely distributed LASO package of Parlett and Scott [12]. The Lanczos iteration solves the sparse symmetric eigenproblem with $O(n^{1.4})$ expected complexity, with runtimes generally competitive with those of iterative partitioning methods. [Other fields have also found eigenvector computations to be efficient, e.g., Simon [13] has used spectral methods to partition very large finite-element grids].

Alpert and Kahng [2] and Chan, et al. [4] have utilized d-dimensional spectral embeddings to construct high-quality multi-way netlist partitioning algorithms. Since a geometric representation of the netlist requires only $O(n)$ space to capture $\Theta(n^2)$ "distances", both methods have low algorithmic complexity and memory requirements. Given the netlist embedding, Alpert and Kahng apply KC, an efficient $O(n \log k)$ geometric partitioning heuristic. Chan et al. apply the more effective KP heuristic which has $O(nk^2 + nk \log n)$ complexity, assuming constant degree bounds. Alpert and Kahng also studied the criticality of differing clique net models with respect to performance; based on this result, we will adopt their "partitioning-specific" net model to generate geometric netlist embeddings[2].

While KC and KP demonstrate that spectral geometric embeddings preserve fundamental netlist properties, any partitioning algorithm that ignores netlist topology or the underlying objective will be handicapped. For instance, KC minimizes a geometric measure (maximum cluster diameter) which has only a heuristic correlation to Scaled Cost or other netlist-dependent objectives. Similarly, KP assigns modules to clusters via a directional cosine distance measure and does not use the netlist except when assigning the more "difficult" modules; the actual Scaled Cost objective is never used in the cluster assignment.

We believe that the geometric embedding should serve as a *guide*, not an absolute – we require a new partitioning approach that utilizes both geometric and netlist information together, and that can directly optimize topology-dependent objectives such as Scaled Cost.

[2] The "partitioning-specific" clique net model assigns cost $\frac{4}{p \cdot (p-1)}$ to each edge in the clique that represents a p-pin net.

1.2 Overview of Our Approach

To exploit both a d-dimensional geometric embedding and the underlying netlist topology within an efficient partitioning optimization, we propose a new method based on constructing a "tour" of the embedded points via a *spacefilling curve* (SFC). The SFC-based ordering can be viewed as a 1-dimensional representation of the d-dimensional pointset, which was in turn a representation of the original circuit netlist. This approach has two clear advantages:

- The 1-dimensional SFC-based ordering contains more information than the 1-dimensional ordering generated by a single eigenvector of the Laplacian and used by, e.g., [6].
- The SFC-based ordering is amenable to efficient optimizations that cannot be applied to higher-dimensional representations.

We present a new *Restricted Partitioning* (RP) formulation that requires clusters of the partitioning to be contiguous in the SFC-based ordering. We then show RP can be solved *optimally* by dynamic programming while transparently handling user-specified cluster size constraints. This dynamic programming technique holds theoretical interest since all of the unrestricted partitioning formulations of interest (ratio cut, min-bisection, Scaled Cost, etc.) are NP-complete. Figure 1 contrasts the original methodology of Hall (a) with previous spectral k-way partitioning methods (b), and our new multi-way partitioning algorithm (c).

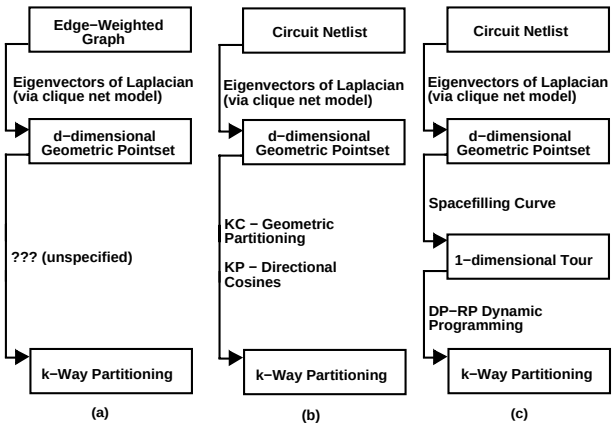


Figure 1: (a) Methodology proposed by Hall; (b) KC [2] and KP [4] algorithms; and (c) our new multi-way partitioning algorithm.

The remainder of our paper is organized as follows. Section 2 presents our new Restricted Partitioning formulation. Section 3 presents spacefilling curves for creating a one-dimensional orderings. Section 4 describes the DP-RP dynamic programming algorithm. Section 5 gives experimental results indicating that DP-RP is an excellent k-way partitioning heuristic in terms of Scaled Cost, for $2 \leq k \leq 5$.

2 Restricted k-Way Partitioning

A genesis of our new approach lies in the Traveling Salesman Problem (TSP) heuristic of Karp [8], which uses a partitioning of a planar pointset to construct a tour. Karp's heuristic visits the clusters of the partitioning one at a time, and visits every point in a particular cluster before moving to the next cluster. We ask whether an "inverse" methodology can succeed, i.e., whether we can use a tour of the geometric pointset to generate a partitioning. We require each cluster of the partitioning to be a contiguous "slice" of the tour, thereby obtaining the following general approach: (i) construct a "good tour" over the geometric points, and (ii) minimize a partitioning objective subject to clusters being contiguous slices of the tour.

We represent a tour by a circular *permutation*[3], i.e., a bijection $\Pi : V \rightarrow V$. If we write $\Pi(v_i) = v_{\pi_j}$; then Π can be expressed using the ordered-set notation $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$, i.e., the tour begins with point v_{π_1}, visits v_{π_2}, etc., until it visits v_{π_n} and then finally revisits v_{π_1}. A *slice* $[i, j]$ of Π is a contiguous subset of Π; we treat indices modulo n so that $[i, j] = \{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi_j}\}$ if $i \leq j$ and $[i, j] = \{v_{\pi_j}, v_{\pi_{j+1}}, \ldots, v_{\pi_n}\} \cup \{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_i}\}$ if $i > j$. We now define the "restricted" k-way partitioning problem:

Restricted k-Way Partitioning (RP): Given a permutation $\Pi : V \rightarrow V$, cluster size bounds L and U, a value $2 \leq k \leq |V|$, and an objective f, partition V into disjoint clusters $P^k = \{C_1, C_2, \ldots, C_k\}$ that optimizes $f(P^k)$ such that:

Condition 1: if $v_{\pi_i}, v_{\pi_j} \in C$ for some cluster C, then either

(a) $[i, j] \subseteq C$, or

(b) $[j, i] \subseteq C$.

Condition 2: $L \leq |C_j| \leq U, \quad 1 \leq j \leq k$.

Condition 1 captures the restriction that clusters must be slices of Π, and Condition 2 adds cluster size constraints. The RP formulation in effect seeks to partition a 1-*dimensional* representation of V, namely, Π. We will show that we can solve RP *optimally* for a large class of objectives (including Scaled Cost) in polynomial time.

Given the embedded netlist modules V, our multiway partitioning methodology first constructs a tour Π using a *spacefilling curve*, then applies an optimal dynamic programming algorithm to minimize Scaled Cost within the RP formulation. A variant of RP removes Condition 1(b), thus requiring clusters to be slices from a *linear ordering* rather than from a tour. This restriction narrows the solution space but allows an $O(n)$ factor speedup; our experiments use linear orderings to capitalize on this complexity savings.

[3] In the discussion below, we use $V = \{v_1, v_2, \ldots, v_n\}$ to denote either the modules of the netlist or the corresponding points in d-space, i.e., v_i can denote either a module or a point in $\Re^d$. Our discussion will often overload these meanings for notational convenience, e.g., we may discuss a partitioning of $V \subset \Re^d$ with respect to the netlist topology over V. The meaning will be clear from the context.

3 Spacefilling Curves in d Dimensions

Recall that two modules that are strongly connected in the netlist will tend to be near each other in the spectral geometric embedding. We now seek a good tour of the embedding such that points of V that are close to each other in the embedding remain near each other in Π. Furthermore, if two strongly connected modules are *not* actually adjacent in the tour, they should be separated by modules with which they may profitably share a cluster. From the above intuition, a good TSP solution over the embedded pointset should suffice since it is unlikely to wander out of, and then back into, a natural cluster. However, it is not obvious which TSP heuristic to choose. For example, even relatively effective methods such as the greedy nearest-neighbor approach can yield long edges, forcing distant points to be adjacent in the tour.

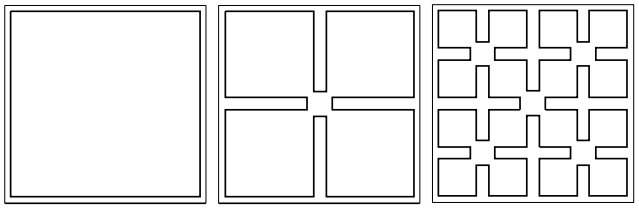
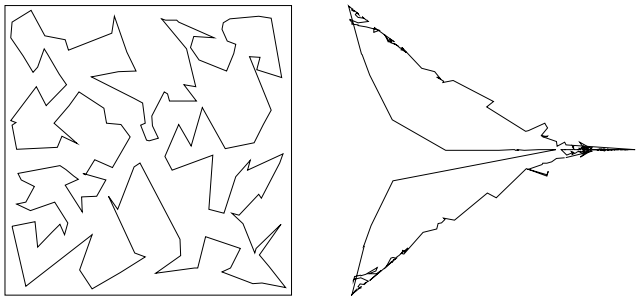


Figure 2: The planar Sierpinski spacefilling curve.

Bartholdi and Platzman have used spacefilling curves as the basis of a provably good TSP heuristic [3]. They use the recursive construction due to Sierpinski (1912), the 2-dimensional case of which is shown in Figure 2. In the Figure, the successive approximations are progressively refined until the curve "fills" up the unit square to the necessary precision, i.e., it passes over every point. The order in which points of a TSP instance are visited by the curve yields the heuristic TSP solution. Figure 3 shows the SFC tour for (a) a uniformly random set of 150 points in the plane, and (b) the 2-dimensional spectral embedding of the Primary1 layout synthesis benchmark.



(a) (b)

Figure 3: The Sierpinski tour over (a) 150 random points, and (b) the 2-dimensional spectral embedding of the Primary1 benchmark.

In practice, the SFC heuristic yields planar TSP tours within 25% of optimal for uniformly random instances and also has an $O(\log n)$ worst-case error bound [3]. For uniformly random pointsets, [3] em-

pirically found that the Sierpinski curve outperforms these and other spacefilling curves as the basis for a TSP heuristic. Other spacefilling curves might be better suited for nonuniform distributions; indeed, [3] has outlined a method for creating application-specific spacefilling curves.

The tour generated by the Sierpinski curve seems to suit our purposes for several reasons.

- Points that are close to each other in d-space will generally also be close along the tour. More critically, the tour avoids long edges, so that if two points are adjacent in the tour, they will be close to each other in the geometry.
- The Sierpinski construction can easily be extended into higher dimensions by recursively applying the 2-dimensional construction.
- The tour is calculated in $O(n \log n)$ time [3], [1].

4 Dynamic Programming (DP) for RP

While the Scaled Cost objective is of primary interest in our work, we cast the following discussion more generally in order to extend our result to a larger class of objectives f.

Let $w(C_i)$ be the cost of having cluster C_i in our partitioning P^k, i.e., the contribution of C_i to the value $f(P^k)$. For example, $w(C_i) = \frac{E_i}{|C_i|}$ in the Scaled Cost objective. The cluster corresponding to slice $[i,j]$ is denoted by $C_{[i,j]}$ and $P^k_{[i,j]}$ denotes a k-way RP solution over the slice $[i,j]$. We use $\hat{P}^k_{[i,j]}$ to denote the *optimal* k-way RP solution over $[i,j]$. Notice that $P^1_{[i,j]} = \hat{P}^1_{[i,j]} = \{C_{[i,j]}\}$. We will use optimal partitioning solutions $\hat{P}^k_{[i,j]}$ as "building blocks" of solutions $\hat{P}^{k'}_{[i',j']}$ where $[i,j] \subset [i',j']$ and $k < k'$.

Since each cluster $C_{[i,j]}$ is uniquely determined by its first and last points v_{π_i} and v_{π_j}, only $(U-L+1)n$ clusters can be part of any RP solution. Our DP-RP algorithm (Figure 4) first computes the cost $w(C_{[i,j]})$ for each of these $(U-L+1)n$ clusters using some procedure **Cluster_Costs** (cf. the discussion below). These clusters form the set of all optimal 1-way partitionings $\hat{P}^1_{[i,j]}$. We then build 2-way partitioning solutions $\hat{P}^2_{[i,j]}$ from the $\hat{P}^1_{[i,j]}$, etc. until all possible k-way partitioning solutions are obtained.

Whether the $\hat{P}^k$ values are optimal depends on the objective f. A sufficient condition for DP-RP to generate optimal solutions is for f to be *monotone nondecreasing* over w: for any $P^k = \{C_1, C_2, \ldots, C_k\}$ and $Q^k = \{C'_1, C'_2, \ldots, C'_k\}$ with $w(C_i) \leq w(C'_i)$ for $1 \leq i \leq k$, f is monotone nondecreasing if and only if $f(P^k) \leq f(Q^k)$. For such objectives f, the *principle of optimality* holds: all subsolutions of an optimal RP solution are themselves optimal RP subsolutions, e.g., if $P^3 = \{C_1, C_2, C_3\}$ is an optimal 3-way RP solution

DP-RP Algorithm

Input: Permutation $\Pi = \{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
$L, U \equiv$ Lower and upper cluster size bounds
$k \equiv$ Number of clusters
Output: $\hat{P}^k \equiv$ Optimal RP solution
Vars: $\hat{P}^{k'}_{[i,j]} \equiv$ Subsolutions
$k' \equiv$ Index denoting current partitioning size
$m \equiv$ Beginning index of possible new cluster
$f_{best} \equiv$ Objective value for best current $\hat{P}^{k'}_{[i,j]}$

1. $\forall i,j$ compute $f(\hat{P}^1_{[i,j]}) = w(C_{[i,j]})$ using **Cluster_Costs**
2. **for** $k' = 2$ **to** k **do**
3. **for each** i,j **do**
4. $f_{best} = \infty$
5. **for** $m = j - U$ **to** $j - L$ **do**
6. **if** $f_{best} < f(\hat{P}^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\})$ **then**
7. $f_{best} = f(\hat{P}^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\})$
$\hat{P}^{k'}_{i,j} = \hat{P}^{k'-1}_{[i,m]} \cup C_{[m+1,j]}$
8. **return** $\hat{P}^k = \hat{P}^k_{[i,i-1]}$ for the i that minimizes $f(\hat{P}^k_{[i,i-1]}), 1 \leq i \leq n$

Figure 4: The DP-RP algorithm. All index manipulations are performed modulo n.

then $\{C_1, C_2\}$ will be the optimal 2-way RP solution for the points in $\{C_1 \cup C_2\}$. Thus, given the set of all optimal $\hat{P}^{k'-1}_{[i,j]}$, we can build the set of optimal $\hat{P}^{k'}_{[i,j]}$ by virtue of $\hat{P}^{k'}_{[i,j]}$ being expressible as $\hat{P}^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\}$ for some m with $L \leq |C_{[m+1,j]}| \leq U$. Since DP-RP considers each possible value of m (Step 5) and records each new partitioning which reduces f (Step 7), every $\hat{P}^{k'}_{[i,j]}$ retained when the loop of Step 5 terminates must be optimal. DP-RP has complexity $O(k(U-L)n^2)$ since, as we will subsequently show, Cluster_Costs has only $O(nU)$ complexity. When there are no cluster size constraints, DP-RP has $O(kn^3)$ complexity.

When only a few relatively large values of k are of interest, DP-RP can be improved to $O(\log k(U-L)n^2)$ complexity by the technique of addition chains. We first construct optimal 4-way partitionings from 2-way optimal partitionings, then 8-way partitionings from 4-way partitionings, etc. Optimal k-way solutions may now be derived from the 2^i-way partitioning subsolutions corresponding to the $\log k$ values of 2^i that sum to k.

The class of partitioning objectives $f(P^k)$ which are monotone nondecreasing for some cluster weight function w includes Scaled Cost and the common geometric partitioning objectives, Min-Max-Diameter and Min-Sum-Diameters[4]. [When f corresponds to Min-Max-Diameter, DP-RP significantly outperforms standard Min-Max-Diameter partitioning algorithms

[4] $diam(C)$ is the maximum of the geometric distances between every pair of points in C. Setting $w(C) = diam(C)$, $f(P^k) = \min_{1 \leq i \leq k} w(C_i)$ for Min-Max-Diameter and $f(P^k) = \sum_{1 \leq i \leq k} w(C_i)$ for Min-Sum-Diameters; see [1] for more details.

for uniformly random pointsets, illustrating the flexibility of the DP-RP approach [1].]

Figure 5 describes Cluster_Costs for f corresponding to Scaled Cost; the procedure returns all cluster costs $w(C_{[i,j]})$ in $O(nU)$ time, assuming $O(n)$ signal nets. This is a valid assumption since both fanout and cell I/O are bounded for any given technology. Steps 1-9 calculate the outdegree for each cluster, and $w(C_{[i,j]})$ is computed in Step 10. Given the value of $w(C_{[i,j-1]})$, one can compute $w(C_{[i,j]})$ by adding v_{π_j} to cluster $C_{[i,j-1]}$ and checking whether any cut nets become completely contained in the cluster (Step 8), or whether any previously uncut nets become cut (Step 9). Appropriate data structures allow Steps 8 and 9 to be executed in constant time, hence Cluster_Costs has $O(nU)$ time complexity [1].

Cluster_Costs

Input: Permutation $\Pi = \{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
$L, U \equiv$ Lower and upper cluster size bounds

Output: $w(C_{[i,j]}) \equiv \frac{E_{[i,j]}}{|C_{[i,j]}|}$ for every possible cluster $C_{[i,j]}$

Vars: $\Delta \equiv$ One less than size of current clusters
$S_i \equiv$ Set of signal nets which contain module v_{π_i}

1. **for** $i = 1$ **to** n **do**
2. $w(C_{[i,i]}) = |S_i|$ where $S_i = \{s | \text{signal net } s \text{ contains module } v_{\pi_i}\}$
3. **for** $\Delta = 1$ **to** U **do**
4. $j = ((i + \Delta - 1)$ **mod** $n) + 1$
5. $S_j = \{s | \text{signal net } s \text{ contains module } v_{\pi_j}\}$
6. $w(C_{[i,j]}) = w(C_{[i,j-1]})$
7. **for every** $s \in S_j$ **do**
8. **if** (s is completely contained in $C_{[i,j]}$) **then** decrement $w(C_{[i,j]})$
9. **if** (s contains no modules of $C_{[i,j-1]}$) **then** increment $w(C_{[i,j]})$
10. **for every** $w(C_{[i,j]})$ calculated **do** $w(C_{[i,j]}) = \frac{w(C_{[i,j]})}{j-i+1}$

Figure 5: Cluster_Costs (Scaled Cost).

So far, we have considered the RP formulation where both conditions 1(a) and 1(b) apply. DP-RP may require up to $O(kn^3)$ time complexity, which is prohibitive for practical netlist sizes. However, eliminating rule 1(b) changes the tour into a linear ordering, which restricts the solution space but allows a factor of n speedup, i.e., $O(kn(U-L))$ time complexity and $O(kn^2)$ when there are no cluster size constraints. The speedup arises since we are guaranteed that some cluster begins with index π_1; thus, for each value of k' (see Step 2 of Figure 4) we maintain only $O(n)$ subsolutions of form $P^{k'}_{[1,j]}$ instead of $O(n^2)$ subsolutions of form $P^{k'}_{[i,j]}$.

5 Results and Future Work

We initially generated a linear ordering from the Sierpinski tour by removing the tour's largest edge. We also generated a subsequent linear ordering based on the split of the 2-way partitioning solution, i.e., if the original solution yielded $P^2 = \{C_{[1,m]}, C_{[m+1,n]}\}$, then the second linear ordering was $\{\pi_{m+1}, \pi_{m+2}, \ldots, \pi_n, \pi_1, \pi_2, \ldots \pi_m\}$. We repeated this procedure to generate a third linear ordering and then recorded the best partitioning solution obtained from the three linear orderings. The subsequent linear orderings did not afford significant improvements over the initial linear ordering.

For each benchmark in our experiments, we considered the d-dimensional embedding derived from d eigenvectors ($1 \leq d \leq 10$), and then computed the spacefilling curve for each embedding. Our experiments also set $L = 1$ and $U = n$ in order to consider the full range of solutions. We recorded the best solutions obtained by running DP-RP on 1- through 10-dimensional embeddings.

The best Scaled Cost results are recorded in Table 1. We compare DP-RP to KC [2], KP [4] and EIG1 [6], the results of which are quoted from [4]. Table 1 shows that DP-RP is significantly superior to EIG1, KP and KC, particularly for $k = 2$ through $k = 5$. For example, we see that DP-RP averages 45% improvement over both KP and EIG1 for $k = 2$. Generating the Sierpinski tour and executing DP-RP for a single linear ordering was also reasonably efficient: we obtained k-way partitioning solutions for all values $2 \leq k \leq 10$, with $d = 5$, in 63 seconds for Primary1 and in 633 seconds for Primary2 on a Sparc IPX.

The 1-dimensional SFC ordering captures useful information from the *multi*-dimensional spectral embedding that is not available from the 1-dimensional spectral embedding. To see this, we executed DP-RP on the linear ordering corresponding to the eigenvector of the smallest non-zero eigenvalue of the Laplacian. For k-way partitioning with $2 \leq k \leq 10$, DP-RP applied to the SFC linear ordering obtained anywhere from 26.4% (19ks) to 84.0% (Test05) *average* reduction in Scaled Cost versus DP-RP applied to the single-eigenvector linear ordering. Thus, our SFC-based ordering is valuable for its ability to capture information from d eigenvectors while still remaining tractable to efficient partitioning optimizations.

We note that DP-RP performs poorly as k continues to increase, and we suspect this is due to the fact that RP is too restrictive for larger values of k. The tour successfully captures global information, but when more local decisions have to be made, e.g, for $k \geq 6$, DP-RP's reliance on a "monolithic" spacefilling curve seems less effective. This leaves open the future improvement to DP-RP for larger values of k. One possibility would be to integrate some clustering method which will permanently fix some groups of modules into the same cluster. Then, the spacefilling curve will traverse complete clusters of modules instead of individual modules, with the clusters being subsequently re-expanded before dynamic programming is applied. We also note that although DP-RP can easily handle bounds on cluster size (i.e., module cardinalities), it handles area constraints only with added complexity on the order of the total module area. We leave open the question of whether area constraints can be integrated without increasing algorithmic complexity. On a similar note, we leave open the integration of individual cluster size or area constraints into DP-RP, e.g., $L_1 \leq |C_1| \leq U_1$, $L_2 \leq |C_2| \leq U_2$, etc.

Test Case	ALG	Number of Clusters - k (Best dimension)								
		10	9	8	7	6	5	4	3	2
19ks	DP-RP	17.6	16.8	15.6	14.3	12.7	11.7	8.37	7.74	5.44
	KC		15.0	15.8	15.6	15.1	14.4	13.1	12.5	17.6
bm1	DP-RP	24.8	22.8	20.7	18.1	14.4	11.5	8.89	6.61	5.53
	KC		27.6	30.6	28.6	19.8	17.9	11.1	7.0	5.8
Prim1	DP-RP	38.9	36.7	35.2	31.7	28.8	26.0	22.1	14.7	13.5
	KP	44.7	41.3	32.3	33.2	31.3	29.9	21.2	14.7	13.5
	KC		34.6	33.6	34.4	30.7	27.5	16.4	17.4	13.5
	EIG1	59.4	56.2	51.0	46.6	43.2	40.3	38.9	22.5	13.5
Prim2	DP-RP	13.7	13.3	12.8	12.1	11.0	9.43	7.95	6.86	5.05
	KP	15.0	15.2	13.5	11.0	10.5	10.1	9.24	7.25	4.64
	KC		11.7	12.0	11.8	11.5	10.4	9.0	7.5	5.9
	EIG1	11.1	10.6	10.4	9.56	8.44	8.47	7.56	6.56	4.78
Test02	DP-RP	25.5	24.1	22.8	20.9	18.5	16.1	13.4	10.9	8.07
	KP	24.2	23.4	21.5	19.0	16.4	13.9	14.1	12.7	9.26
	KC		21.5	21.2	21.1	21.2	23.1	23.6	19.1	30.1
	EIG1	25.5	23.8	21.7	20.3	18.9	14.5	13.0	11.4	8.73
Test03	DP-RP	22.6	21.1	19.2	17.1	16.2	15.2	14.3	13.0	10.2
	KP	20.6	20.1	19.8	17.6	18.0	17.5	20.2	15.0	31.2
	KC		21.0	22.4	23.2	22.4	22.2	19.3	21.4	16.7
	EIG1	19.8	17.9	17.3	17.0	16.9	16.9	17.1	20.7	31.2
Test04	DP-RP	22.2	19.9	17.8	17.6	16.5	15.1	12.4	8.19	5.85
	KP	17.4	17.6	20.0	15.8	15.2	14.3	14.8	18.9	66.1
	KC		22.1	23.8	24.4	24.3	27.2	27.4	36.0	66.1
	EIG1	21.9	21.6	23.3	24.4	24.9	28.2	32.3	37.4	66.1
Test05	DP-RP	9.88	8.66	8.06	7.84	7.32	6.56	5.49	4.90	3.15
	KP	9.32	7.21	8.91	8.66	7.64	7.98	8.07	8.65	11.3
	KC		11.0	10.6	10.7	11.1	10.3	8.8	10.2	10.6
	EIG1	9.28	8.43	6.58	6.42	6.22	6.28	6.30	6.37	8.94
Test06	DP-RP	27.1	25.1	23.7	20.2	18.4	16.6	14.2	11.3	9.2
	KP	21.3	21.6	20.7	18.5	17.2	15.0	18.0	12.1	28.6
	KC		31.0	32.4	33.6	26.4	28.8	25.9	19.3	28.6
	EIG1	21.7	21.7	22.7	14.1	16.6	15.1	16.6	18.6	28.6

Table 1: Scaled Cost values ($\times 10^5$) of best k-way partitions obtained using d-dimensional embeddings, $1 \leq d \leq 10$. Results for KP and EIG1 are quoted from [4], and results for KC are quoted from [2].

Acknowledgments

We have enjoyed research discussions with Pak Chan, Martine Schlag and Jason Zien. Lars Hagen and Jen-Hsin Huang developed the ideas behind the partitioning-specific net model. Part of this work was performed during a Spring 1993 visit to UC Berkeley; the hospitality of Professor Ernest S. Kuh and his research group is gratefully acknowledged.

References

[1] C. J. Alpert and A. B. Kahng, "Multi-Way Netlist Partitioning Using Spacefilling Curves," *UCLA technical report 930016*, 1993.

[2] 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.

[3] J. J. Bartholdi and L. K. Platzman, "Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space" *Management Sciences* Vol. 34, No. 3, March 1988, pp. 291-305.

[4] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio Cut Partitioning and Clustering", *Proc. Symp. on Integrated Systems*, Seattle, March 1993. (also see J. Zien, "Spectral K-Way Ratio Cut Graph Partitioning", M.S. Thesis, Computer Engineering Dept., UC Santa Cruz, March 1993, for experimental results).

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

[6] 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.

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

[8] R. M. Karp, "Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane", *Mathematics of Operations Research* 2(3), 1977, pp. 209-224.

[9] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, 1990.

[10] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Matrix Anal. Appl.,* vol. 11, pp. 430-452, 1990.

[11] L. A. Sanchis, "Multiple-way Network Partitioning", *IEEE Trans. on Computers*, 38, 1989, pp. 62-81.

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

[13] H. D. Simon, "Partitioning of Unstructured Problems for Parallel Processing", *technical report*, NAS Systems Division, NASA Ames Research Center, Feb. 1991.

[14] Y. C. Wei and C. K. Cheng, "Ratio Cut Partitioning for Hierarchical Designs", *IEEE Trans. on CAD* 10(7), July 1991, pp. 911-921.

[15] C. W. Yeh, C. K. Cheng and T. T. Lin, "A General Purpose Multiple Way Partitioning Algorithm", *Proc. ACM/IEEE Design Automation Conf.*, June 1991, pp. 421-426.