Multi-Way Partitioning Via Geometric Embeddings, Orderings, and Dynamic Programming*

Charles J. Alpert and Andrew B. Kahng

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

Abstract

This paper presents effective algorithms for multi-way partitioning. Confirming ideas originally due to Hall [27], we demonstrate that *geometric embeddings* of the circuit netlist can lead to high-quality k-way partitionings. The netlist embeddings are derived via the computation of d eigenvectors of the Laplacian for a graph representation of the netlist. As [27] did not specify how to partition such geometric embeddings, we explore various geometric partitioning objectives and algorithms, and find that they are limited because they do not integrate topological information from the netlist. Thus, we also present a new partitioning algorithm that exploits both the geometric embedding and netlist information, as well as a *Restricted Partitioning* formulation that requires each cluster of the k-way partitioning to be contiguous in a given linear ordering. We begin with a d-dimensional spectral embedding and construct a heuristic 1-dimensional ordering of the modules (combining *spacefilling curve* with *3-Opt* approaches originally proposed for the traveling salesman problem). We then apply dynamic programming to efficiently compute the optimal k-way split of the ordering for a variety of objective functions, including Scaled Cost [12] and Absorption [44]. This approach can transparently integrate user-specified cluster size bounds. Experiments show that this technique yields multi-way partitionings with lower Scaled Cost than previous spectral approaches [2] [12] [25].

1 Introduction

Partitioning optimizations are critical to the system-level synthesis of complex VLSI designs. Systems with several million transistors are now common, and 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, with the traditional goal being to minimize the number of signals which pass between the components.

An increasingly typical design flow will synthesize an HDL program via logic synthesis tools, then place and route the resulting netlist using back-end tools. This flexible methodology improves design cycle time, allows rapid design changes, and avoids human error in layout. However, even with the increased maturity

*Partial support for this work was provided by a Department of Defense Graduate Fellowship, by NSF MIP-9110696 and 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. This work is an extension of preliminary results reported in *Proc. ACM/IEEE Design Automation Conf.*, 1993 and 1994.

of the high-level synthesis tools that are available for such a strategy, netlist-level partitioning remains central to the success of the design procedure. This is for reasons such as the following.

- First, HDL subroutines may not correspond to ideal physical block partitions. Thus, while previous building-block physical design methodology gave a small number of function blocks that could be optimally hand-partitioned, experience with HDL-based synthesis is that partitioning a flattened design representation often leads to improved solution quality [15]. Flattened partitioning inputs may also be required by such applications as technology migration or the design of precursor systems (i.e., finding the packaging tradeoffs that correspond to optimum cost-performance points at early stages in the product cycle).
- Second, system designers are now *software designers* who write HDL code and pass synthesized gate-level netlists to floorplanning and physical layout tools, without necessarily understanding the hardware resource implications of the HDL code. In particular, an *a priori* understanding of interconnect attributes (for example, the effect of the few lines of code that specify a register file or crossbar) is difficult to achieve. Because the function-to-layout transformation cannot be predicted (this is exacerbated by synthesis tools' well-known difficulties with control logic and exception handling), system partitioning must be performed at a lower level of the hierarchy.
- Third, as noted above, system complexity has reached such levels that partitioning is required simply to *enable* the application of current toolsets to smaller pieces of the design.

Beyond its layout-directed applications, the partitioning task also arises in other phases of system synthesis and verification, such as area/performance estimation, partitioning for testability, and hardware simulation and emulation.

Definition: A *k-way partitioning* P^k of a set V is a set of k disjoint, nonempty *clusters* (i.e., subsets of V) $\{C_1, C_2, \ldots, C_k\}$ such that $C_1 \cup C_2 \cup \ldots \cup C_k = V$.

Most of the previous work in partitioning has focused on 2-way partitioning algorithms that in practice are recursively applied to generate k-way partitionings. As noted in [12], this 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 connectivity between the subcircuits. Our general problem statement is as follows.

General k-Way Partitioning: Given a circuit netlist hypergraph $H = (V, E)$ with modules $V = \{v_1, v_2, \ldots, v_n\}$ and nets $E = \{e_1, e_2, \ldots, e_m\}$, and a value $2 \le k \le n$, find a *k-way partitioning* P^k that optimizes a given objective function $f(P^k)$.

1.1 Previous Work

Previous work on 2-way partitioning has centered on the minimum bisection and minimum ratio cut [46] objectives. Approaches to multi-way partitioning have involved seeded epitaxial growth, extensions of the Fiduccia-Mattheyses iterative bipartitioning algorithm [41], a primal-dual iteration motivated by a generalization of the minimum ratio cut metric [47], and spectral approaches [25] (see [3] for an extensive survey).

For multi-way partitioning, Yeh et al. [48] proposed a "shortest-path clustering" method, where each iteration partially erases edges of a shortest path between a random pair of modules. The iterations terminate when enough edges have been erased for there to remain k connected components, i.e., the clusters. The algorithm probabilistically captures the relationship between multicommodity flow and minimum ratio cut, and yields high-quality solutions when measured by *cluster ratio*, which Yeh et al. characterize as the "proper" k-way generalization of the ratio cut objective:

Cluster Ratio: Minimize:

$$f(P^k) = \frac{c(P^k)}{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} |C_i| \times |C_j|}$$

where $c(P^k) = \{e \in E \mid \exists u, v \in e \text{ such that } u \in C_i, v \in C_j, i \neq j\}$ is the number of nets that cross between two or more clusters in P^k.

Another multi-way partitioning approach extends well-established spectral methods. Hall [27] showed that the eigenvector corresponding to the second smallest eigenvalue of the netlist Laplacian[1] minimizes a squared-wirelength objective. Hagen and Kahng [25] based their work on the relationship between this second eigenvalue and the optimum ratio cut cost. In [12], Chan, Schlag and Zien generalize such spectral results to k-way ratio cut partitioning, using the first k eigenvectors of the netlist Laplacian to construct an orthogonal *projector* that maps an n-dimensional space into a k-dimensional space. Ideally, the n elementary unit vectors (the modules) in the n-space will be mapped to exactly k distinct points (the clusters) in the k-space by this projector. Since this is not the case in practice, [12] uses a heuristic (called KP) based on directional cosines to obtain a k-way partitioning of the modules embedded in k-space.

Chan et al. propose *Scaled Cost* as the natural multi-way generalization of the ratio cut objective:

Scaled Cost: Minimize:

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

where $E_i = \{e \in E \mid \exists u, v \in e \text{ such that } u \in C_i, v \notin C_i\}$ is the set of signal nets crossing the boundary of cluster C_i. Both [12] and [48] establish robust generalizations of the ratio cut concept. Each metric

[1] Given an edge-weighted graph representation of the netlist, $G(V, E_G)$, we may construct from G the $n \times n$ *adjacency matrix* $A = A(G)$ in which A_{ij} has weight equal to that of $e(v_i, v_j) \in E_G$ (by convention $A_{ii} = 0$ for all $i = 1, \ldots, n$). If we let $deg(v_i)$ denote the degree of 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, denoted as Q, is given by $Q = D - A$. The Laplacian provides the advantages of having all non-negative eigenvalues and the multiplicity of the zero eigenvalue equals the number of connected components in G.

automatically integrates both cut nets and size balance among the clusters, but each is also shown NP-complete by restriction to minimum ratio cut. For reasons that are clarified in Section 6.1, our work will address the Scaled Cost objective of [12].

While multi-way partitioning typically entails finding k clusters for small values of k (e.g., $k \leq 10$),large values of k (e.g., $k = \frac{n}{5}$) arise in the *two-phase* use of the Fiduccia-Mattheyses (FM) bipartitioning heuristic [19]. Here, the k-way partitioning reduces the problem size by inducing a "compacted" netlist in which each cluster becomes a module, inducing a problem instance of size k; the smaller solution space can then be searched more effectively [34]. After compaction, heuristic FM bipartitioning is performed, then the solution is re-expanded into a "flat" initial configuration for a second FM phase. A successful two-phase FM implementation depends on a high-quality clustering decomposition, i.e., a good k-way partitioning for the regime of $k = \Theta(n)$.

To achieve such a decomposition, Bui et al. [9] proposed the "matching-based compaction" (MBC) algorithm, where the edges of a maximal random matching in the netlist graph induce a compacted graph of $\frac{n}{2}$ vertices, and the compaction is iterated until the problem size becomes manageable. Ng et al. [37] proposed a clustering algorithm that attempts to minimize the Rent parameter of the compacted netlist; [16] also used Rent-based clustering to improve the performance of a placement algorithm. Hagen and Kahng [26] developed the probabilistic RW-ST method, which identifies clusters by recording multiple revisitations of modules by a random walk in the circuit netlist. [26] reports that the RW-ST clusters lead to significant improvements in two-phase FM performance as compared to the MBC strategy, but $\Theta(n^3)$ time is required to process a random walk of the recommended $\Theta(n^2)$ length. Cheng and Wei [14] have developed a "stable, two-way" algorithm that uses recursive FM-based ratio cut partitioning to construct the compacted netlist. In contrast to the other works, [14] reports two-phase FM results for bipartitioning with a 1:3 size ratio bound (as opposed to an exact bisection).

Recently, Sun and Sechen [44] have applied a similar two-phase clustering methodology in the TimberWolf7.0 placement algorithm: simulated annealing is used to derive a clustering that maximizes the following *Absorption* objective.

Absorption: Maximize:

$$f(P^k) = \sum_{i=1}^{k} \sum_{\{e \in E \ | \ e \cap C_i \neq \emptyset\}} \frac{|e \cap C_i| - 1}{|e| - 1}$$

We call this the "Absorption" objective because it counts the number of nets which are "absorbed" by the clusters: if a net e is incident to cluster C_i, then it adds absorption $= (p-1) \cdot \frac{1}{|e|-1}$ to the cluster, where p is the number of pins of e in the cluster. Put another way, if the i^{th} net e_i has its pins distributed among c_i clusters, then $f(P_k) = \sum_{i=1}^{m} \frac{|e_i| - c_i}{|e_i| - 1}$. Maximizing Absorption is a very reasonable k-way partitioning heuristic for $k = \Theta(n)$, but may not be as useful for small k since the objective does not consider cluster sizes.

1.2 Overview of Contributions

Definition: A *d-dimensional geometric embedding* of $H(V, E)$ is a set of $|V|$ points in d-space that are in one-to-one correspondence with the modules of V.[2] A *d-dimensional spectral embedding* is a d-dimensional geometric embedding derived using d eigenvectors of the discrete Laplacian (see Section 2).

Our main premise is that a multi-dimensional geometric embedding of the netlist can lead to effective multi-way partitioning solutions. The geometric representation can improve both algorithmic complexity and memory requirements since it captures implicitly the concept of "distance" or "separation" between netlist modules, i.e., the n points define $\Theta(n^2)$ distances.

Hall's pioneering work [27] discusses partitioning a spectral embedding of a graph in order to obtain a graph partitioning. However, Hall did not specify how to partition the geometric embedding. Thus, we first examine various geometric partitioning objectives and heuristics from the literature, as well as the partitioning solutions derived when these heuristics are applied to spectral embeddings of circuit netlists. Although a strictly geometric approach can yield useful partitioning solutions, it cannot directly utilize the netlist topology. We address this limitation via a new approach to partitioning that utilizes both information from the spectral embedding and the netlist topology. Our contributions are the following.

- In the geometric partitioning, both the partitioning objective and the heuristic used to optimize the objective dramatically affect solution quality. We find that diameter-related objectives and the K-Center (KC) and Complete-Linkage (CL) heuristics yield the highest quality solutions for k-way (and two-phase FM) partitioning applications. Our results support the observation of Chan et al. [12] that the "proper" embedding dimension which reveals a k-way partitioning will grow with k. However, we also observe that a strictly geometric approach has inherent limitations.

- We propose a new method based on a *Restricted Partitioning* (RP) formulation. Given a "tour" or "ordering" of the modules, the k-way RP formulation seeks a k-way partitioning such that each cluster is a contiguous subset of the tour. The advantage of the RP formulation is that it can be solved optimally using dynamic programming; this makes the ordering the "bottleneck" in our methodology, leading us to seek good ordering methods.

- For an RP solution to be of high-quality, the ordering must preserve connectivity attributes of the netlist. We hypothesize that a good solution to the traveling salesman problem (TSP) on the spectrally embedded instance will yield such an ordering. Such an ordering can be viewed as a 1-dimensional representation of the d-dimensional geometric embedding (which was in turn a representation of the original circuit netlist). Our TSP heuristic uses the *spacefilling curve* (SFC) construction of [5] and then applies the fast 3-Opt implementation described in [6]. Empirically, these orderings are more useful for multi-way partitioning information than the 1-dimensional single-eigenvector ordering used

[2] We use $V = \{v_1, v_2, \ldots, v_n\}$ to denote either the modules of the netlist or the points of the geometric embedding, i.e., v_i can denote either a netlist module or a point in $\Re^d$. We overload these meanings for notational convenience; the meaning should be clear from the context.

in, e.g., [23] [25].

- Finally, we describe the DP-RP algorithm which uses dynamic programming to *optimally* solve the RP formulation for various objective functions in the literature. These objectives include Scaled Cost [12], Absorption [44] and diameter-related objectives. DP-RP can transparently handle user-specified cluster size bounds. We show that the use of DP-RP together with our heuristic (SFC + 3-Opt) orderings achieves improvements over previous spectral methods.

Figure 1 contrasts the original methodology of Hall (a) with our specific implementations of Hall's methodology (b), and our new multi-way partitioning algorithm (c).

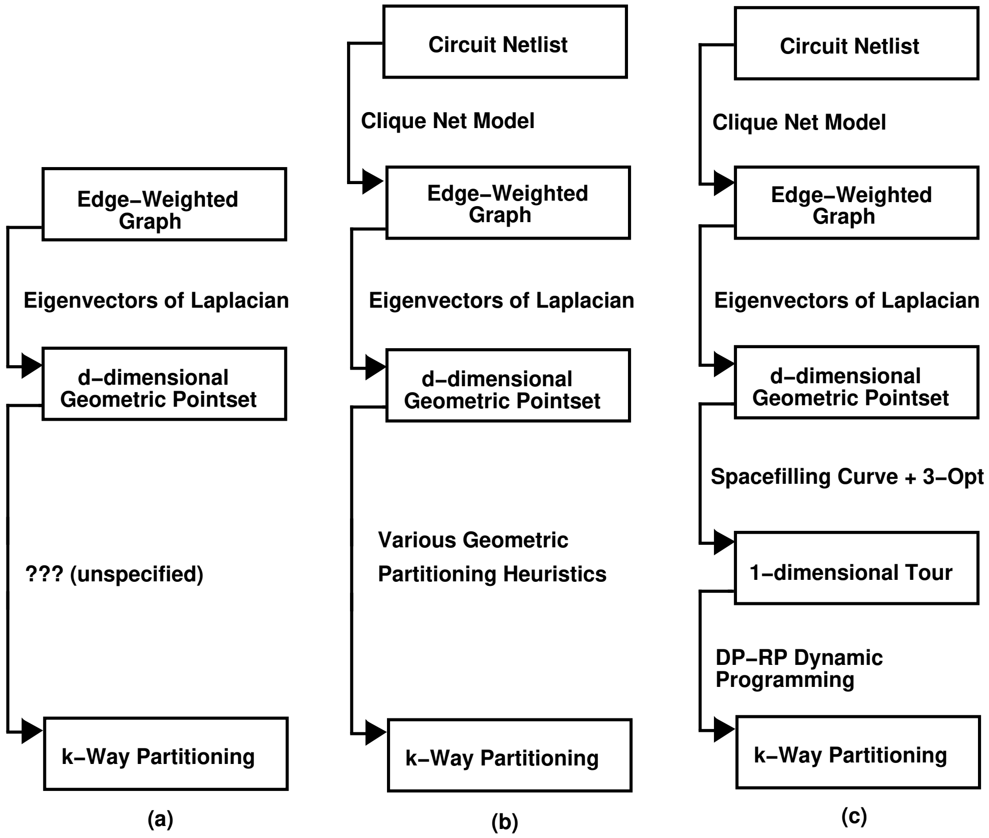


Figure 1: (a) Methodology proposed by Hall; (b) our study of specific implementations of Hall's methodology; and (c) our new multi-way partitioning algorithm.

The remainder of this paper is organized as follows. Section 2 describes the construction of the spectral embedding via a given clique net model and the eigenvectors of the Laplacian. Section 3 gives taxonomies of traditional geometric partitioning objectives and algorithms, and presents experimental results indicating both the promise and the limitations of the "basic Hall approach" (Figures 1(a) and (b)). Section 4 motivates and describes the Restricted Partitioning problem formulation. Section 5 describes the SFC + 3-Opt ordering heuristic, while Section 6 presents the DP-RP algorithm and shows how it may be applied to Scaled Cost and Absorption. Finally, Section 7 gives experimental results and directions for future research.

2 The Spectral Embedding

Three motivations for netlist partitioning based on a geometric embedding are notable as problem instances grow large. First, algorithm speedups often result when a graph input is known to be embedded in a geometric space [39]. Second, storage requirements for a geometric pointset are less than for the original (hyper)graph representation. Third, we obtain a natural measure of "distance" or "proximity" between two modules which can be evaluated in $O(1)$ time.

Intuitively, a successful geometric embedding is "distance-preserving": the distance between two points in the embedding should reflect the connectivity between the corresponding modules in the netlist. We adopt the two-step spectral approach (see, e.g., [12] and its survey of previous works).

- The netlist $H(V, E)$ is first transformed into an edge-weighted graph $G(V, E_G)$ using a clique net model (we give results for three such models including the "partitioning-specific" net model with edge weight $\frac{4}{p(p-1)}$ for a p-pin net[3]).
- The first d eigenvectors corresponding to the smallest nonzero eigenvalues of the Laplacian of $G(V, E_G)$ are then used to embed the netlist into d dimensions.

As with previous works of [12] [23] [25], we are motivated by the well-established relationships between eigenvectors of a graph's Laplacian and either min-cut partitionings or minimum squared-edgelength placements of the graph (see, e.g., [4] for a concise overview). Let $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_n}$ be the n eigenvectors of Q corresponding to eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Every eigenvector $\vec{\mu_i}$ gives a *distinct*, 1-dimensional spatial embedding of the circuit graph. Hall [27] showed that the total squared wirelength of the 1-dimensional placement given by $\vec{\mu_i}$ is λ_i. Hence, the eigenvectors with the lowest eigenvalues will correspond to embeddings that are the most "distance-preserving".

A useful property of the Laplacian of G is that $\lambda_1 = 0$ (hence, $\vec{\mu_1} = [\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}]$ is uninteresting) and $\lambda_2 > 0$ if G is connected. Since this is the case with all of our test circuits, we derive a d-dimensional embedding using the eigenvectors $\vec{\mu_2}, \ldots, \vec{\mu_{d+1}}$. The i^{th} component of $\vec{\mu_j}$ gives the $(j-1)^{st}$ coordinate of module v_i in $\Re^d$. The eigenvectors may be computed using readily available Lanczos codes which efficiently solve the sparse symmetric eigenproblem. For example, Hagen and Kahng [25] use the code of Simon reported in [38]; Hadley et al. [23] and Chan et al. [12] use the more widely distributed LASO package of

[3] This edge weight is motivated by a 2-way partitioning analysis such that the *expected* cost of cutting the net is one [1]. We choose this as an alternative to the "standard" $\frac{1}{p-1}$ net model [13] that has been used by e.g., [25]. While experimental results in Section 3 and 6 support this choice, many other reasonable net model choices are available as well. For example, Hanan et al. [28], Huang [30], Frankle and Karp [20], and Tsay and Kuh [45] have proposed using the clique net model with weights $\frac{2}{p}$, $\frac{6}{p(p+1)}$, $(\frac{2}{p})^{\frac{3}{2}}$, and $(\frac{2}{p})^3$ respectively. Our choice of $\frac{4}{p(p-1)}$ is motivated by 2-way partitioning, while most of the other net models are motivated by placement formulations. Hadley et al. [23] also proposed a partitioning-based net model, where the edge weight is a function of p and k. However, since our methodology constructs a single set of eigenvectors which is then used to find k-way partitionings for a range of k values, the Hadley et al. model is not appropriate for our study. (Appropriate net models for k-way partitioning remain an open area of research.) In our experiments, we compare the partitioning-specific net model to both the standard net model and the Frankle net model used by Chan et al. [12] in their KP algorithm. The experiments show that the standard net model is inferior, but more importantly show that the benefits of our approach are relatively independent of the net model used.

Parlett and Scott [43].[4]

3 A Study of Geometric Partitioning

We seek a k-way partitioning of the geometric embedding that will correspond to a k-way partitioning of the netlist. As noted earlier, this is the approach of Hall [27], but Hall specifies neither an objective nor an algorithm for the partitioning. In this section, we explore the taxonomy of geometric partitioning objectives and algorithms in the classification literature, and show the effects of these choices on solution quality. We make the following observations.

- Minimizing the maximum cluster diameter appears to be the best geometric partitioning objective in terms of the netlist partitioning solution.
- The bottom-up Complete-Linkage (CL) algorithm due to S. C. Johnson [31] yields excellent results when used within the two-phase FM approach.
- The $O(dkn)$ K-Center (KC) algorithm due to Gonzalez [21] is competitive with previous netlist partitioning methods despite relying on purely geometric considerations.
- Multi-dimensional spectral embeddings provide more information than a single-eigenvector embedding and seem to permit better k-way partitioning solutions.

3.1 A Taxonomy of Geometric Partitioning Objectives

A large body of work in the classification literature has established various objectives $f(P^k)$ to assess the quality of a given partitioning P^k. Intuitively, clusters should be compact and well-separated. Clusters with small *diameter*, defined as $diam(C) = \max_{u,v \in C} dist(u,v)$, where $dist(u,v)$ denotes the Euclidean distance between u and v, are compact. Clusters with large *split*, defined as $split(C_1, C_2) = \min_{u \in C_1, v \in C_2} dist(u,v)$ are well-separated. Some standard geometric partitioning objectives include the following.

1. **Max-Min-Split:** Maximize
$$f(P^k) = \min_{1 \le i < j \le k} \{split(C_i, C_j)\}$$
2. **Min-Max-Diameter:** Minimize
$$f(P^k) = \max_{1 \le i \le k} \{diam(C_i)\}$$

[4]For sparse matrices, the Lanczos iteration solves the symmetric eigenproblem with $O(n^{1.4})$ expected complexity; its runtimes are generally competitive with those of iterative partitioning methods. For the Parlett-Scott LASO code, the relative CPU cost of achieving a d-dimensional embedding (d eigenvectors) versus a 1-dimensional embedding is indicated in Table 6 below. The reader may note that the spectral computation does have a worst-case $O(n^2)$ space complexity. However, netlist representations are typically quite sparse; in practice they may be made even sparser by discarding large signal nets (e.g., with $|e| > 20$ pins as in the work of Hadley et al. [23]) or $|e| > 99$ pins as in the work of Chan et al. [12].

3. **Min-Sum-Diameters:** Minimize

$$f(P^k) = \sum_{i=1}^{k} diam(C_i)$$

Each of these objectives seems reasonable for our purposes. For the k-way partitioning of a pointset in Euclidean d-space, the following results are known.

- Objective 1 can be solved optimally in polynomial time for all k and all dimensions d [31].
- Objectives 2 and 3 can be solved optimally in polynomial time for $k = 2$ and all d [29].
- Objectives 2 and 3 are NP-complete for $k \geq 3$ and $d \geq 2$ [36].
- Objective 2 can be solved within a factor of two of optimal in polynomial time for all k and all d; however, achieving an error bound less than ≈ 1.97 is NP-complete for $k \geq 3$ and $d \geq 2$ [18].

Thus, although some of these formulations are NP-complete, the geometric embedding allows us optimal algorithms for $k = 2$ and constant-factor approximation algorithms for $k > 2$. We now give a taxonomy of algorithms used to address these partitioning objectives.

3.2 A Taxonomy of Geometric Partitioning Algorithms

The **Single-Linkage** algorithm [31] optimally solves Objective 1; it begins with each point in its own cluster, then iteratively merges the two closest clusters into a single cluster. The same solution can be obtained by removing the $k-1$ largest edges from a minimum spanning tree over V, and letting each of the remaining k connected components define a cluster.

For $k = 2$, algorithms that optimally solve Objectives 2 and 3 are given by Guénoche et al. [22] and Hershberger [29]. These algorithms can be extended to generate heuristic k-way partitionings for $k > 2$ by iteratively reapplying the algorithm to the largest remaining cluster. We call these extensions **Divisive Max-Diameter** and **Divisive Sum-Diameters** respectively.

Just as there are divisive hierarchical approaches for Objectives 2 and 3, it is also reasonable to use bottom-up hierarchical, or *agglomerative*, methods. We have observed that the **Complete-Linkage (CL)** algorithm of Johnson [31] (see Figure 2) is an excellent heuristic for addressing Objective 2, especially for large k [1]. CL starts with each point in its own cluster and then iteratively merges the pair of clusters which minimizes the increase in maximum cluster diameter; Benzecri [8] has given an $O(dn^2)$ implementation of the CL algorithm. The agglomerative technique can also address Objective 3: iteratively merging the pair of clusters that minimize the increase in the *sum* of cluster diameters yields the **Agglom Sum-Diameters** heuristic.

Last, we note that the **K-Center (KC)** algorithm of Gonzalez [21] (see Figure 3) achieves a 2-approximate solution for Objective 2 and can be implemented to run in $O(dkn)$ time. KC iteratively

Complete-Linkage (CL) Algorithm
Input: Points $V = \{v_1, v_2, \ldots, v_n\}$ in d-space
$k \equiv$ Number of clusters
Output: $P^k = \{C_1, C_2, \ldots, C_k\} \equiv k$-way partitioning
Vars: $P^n, P^{n-1}, \ldots, P^k \equiv$ Set of partitionings
1. **for** $i = 1$ **to** n **do**
2. $C_i = \{v_i\}$
3. $P^n = \{C_1, C_2, \ldots, C_n\}$
4. **for** $m = n$ **downto** $k+1$ **do**
5. Find clusters C_i, C_j s.t. $diam(C_i \cup C_j)$ is minimum
6. $P^{m-1} = (P^m \cup \{C_i \cup C_j\}) - C_i - C_j$.

Figure 2: Complete-Linkage (CL) Algorithm.

constructs a set of k "cluster centers" such that the next center chosen is as far as possible from the previously chosen centers. After k centers have been selected, each point is clustered with the nearest cluster center.

K-Center (KC) Algorithm
Input: Points $V = \{v_1, v_2, \ldots, v_n\}$ in d-space
$k \equiv$ Number of clusters
Output: $P^k = \{C_1, C_2, \ldots, C_k\} \equiv k$-way partitioning
Vars: $W \equiv$ Set of cluster centers
1. $W = \{v\}$ for some random $v \in V$
2. **while** $\lvert W \rvert \leq k$ **do**
3. Find $v \in V$ s.t. $\min\limits_{w \in W} dist(v, w)$ is maximized
4. $W = W \cup \{v\}$
5. Let $W = \{w_1, w_2, \ldots, w_k\}$
6. **for** $i = 1$ **to** n **do**
7. Let j be the index s.t. $dist(w_j, v_i)$ is minimized, $1 \leq j \leq k$
8. $C_j = C_j \cup \{v_i\}$

Figure 3: K-Center (KC) Algorithm.

We now examine the performance of these six algorithms (Single-Linkage, Divisive Max-Diameter, Divisive Sum-Diameters, Complete-Linkage, Agglom Sum-Diameters, and K-Center) in terms of netlist partitioning quality.

3.3 Experimental Results

Our first set of experiments compared performance of the six algorithms on nine standard test cases from ACM/SIGDA and Hughes Aircraft Co., with the partitioning-specific net model used to generate the spectral embedding. Some characteristics of these benchmarks are given in Table 1. We generated k-way partitionings of d-dimensional embeddings for $2 \leq k \leq 10$ and $1 \leq d \leq 10$ (recall that a d-dimensional embedding is constructed using the eigenvectors $\vec{\mu}_2, \ldots, \vec{\mu}_{d+1}$). The 2-dimensional spectral embedding for Primary1 is illustrated in Figure 6 (b). In Figure 4, we compare the performance of the six algorithms for each benchmark in terms of Scaled Cost. Each graph point represents the best Scaled Cost value observed for a given number of clusters k from the ten k-way partitionings generated. We make the following observations.

Test Case	# Modules	# Nets	# Pins
19ks	2844	3282	10547
bm1	882	903	2910
Prim1	833	902	2908
Prim2	3014	3029	11219
Test02	1663	1720	6134
Test03	1607	1618	5807
Test04	1515	1658	5975
Test05	2595	2750	10076
Test06	1752	1541	6638

Table 1: Benchmark circuit characteristics.

- Algorithms that address Objective 3 (Divisive Sum-Diameters and Agglom Sum-Diameters) tend to isolate outliers, resulting in poorly balanced clusters and high Scaled Cost values.
- Single-Linkage slightly outperforms Divisive Min-Diameter, but is noticeably inferior to CL and KC. Since Single-Linkage solves Objective 1 optimally, we conclude that Max-Min-Split is not a good objective for netlist partitioning.
- Algorithms that address Objective 2 (Divisive Min-Diameter, KC, and CL) perform best. We conjecture that Divisive Min-Diameter is inferior to KC and CL because our spectral embeddings are dense with few outliers.

CL and KC are clearly the best geometric partitioning algorithms for our purposes, with CL appearing marginally better (0.8%) on average. Because KC has better time complexity ($O(dkn)$ versus $O(dn^2)$), we chose KC for more detailed study, summarized in Table 2. (Comparisons of KC with other spectral algorithms according to Scaled Cost can be made using data in Section 7.) Table 2 suggests that with the KC approach, the partitioning-specific net model leads to slightly improved Scaled Cost, but the results are erratic, ranging from 30.2% worse for Test06 to 43.5% better for Test02. When recording the best of the ten partitionings that KC generated (for $1 \leq d \leq 10$) for each k, we also recorded the embedding dimension that yielded the best solution. This number is shown in parentheses in Table 2 (ties are broken in favor of lower values). Table 2 trivially indicates that using the added information contained in higher-dimensional spectral embeddings leads to improved results.

Digression: Applying CL Clusters in Two-Phase FM

While the time complexity of KC is superior to that of CL, the bottom-up nature of CL suggests that it can yield good results for large $k = \Theta(n)$. Indeed, our separate studies in [1] have shown that CL is effective in addressing Objective 2 for large k.

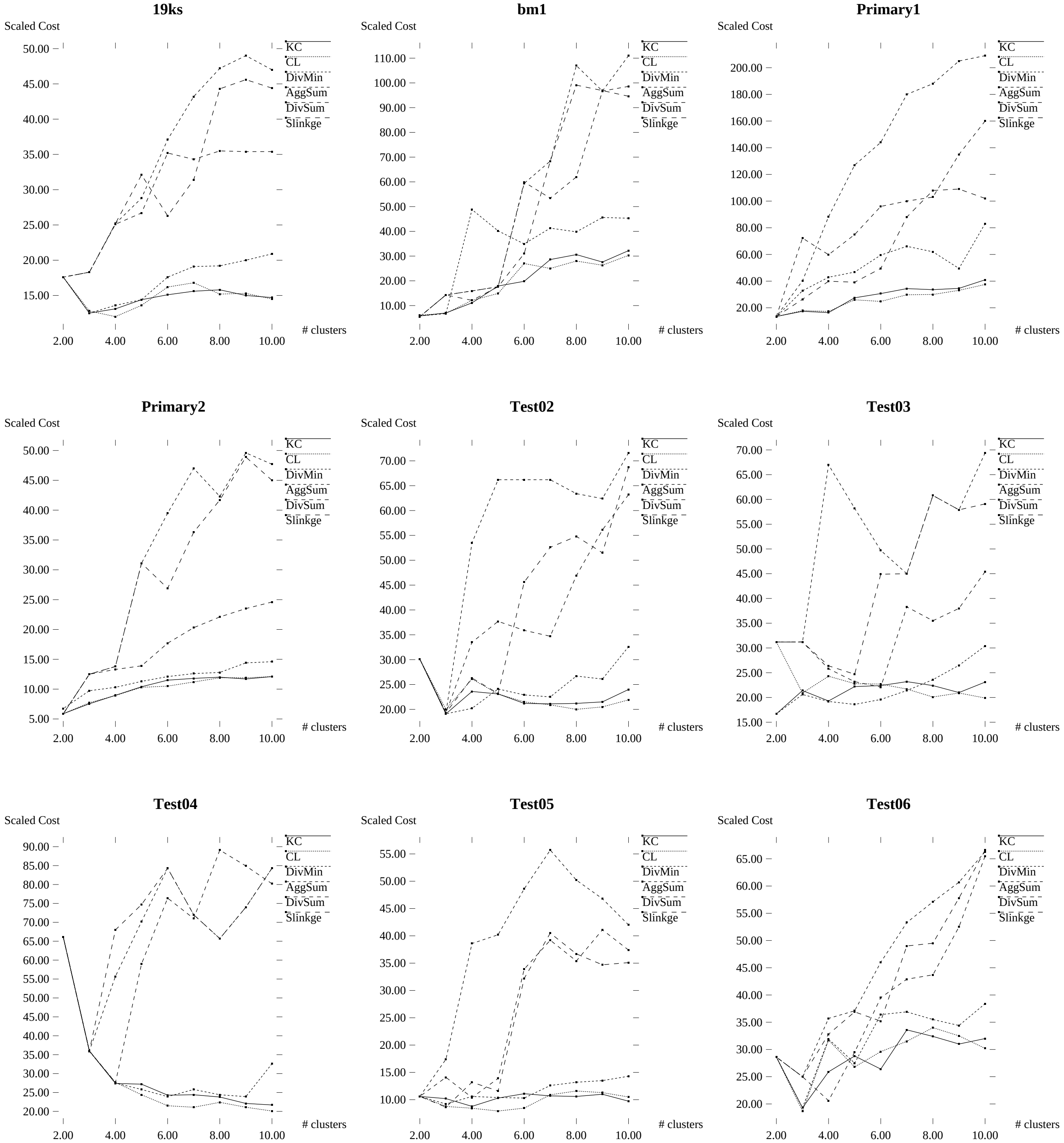


Figure 4: Comparison of the various partitioning objectives and algorithms for nine ACM/MCNC benchmarks, using the partitioning-specific net model to construct the d-dimensional spectral embeddings for $1 \leq d \leq 10$. Each data point represents the lowest Scaled Cost ($\times 10^5$) observed for each of the ten k-way partitionings generated. We use "DivMin" for Divisive Min-Diameter, "Slinkge" for Single-Linkage, "AggSum" for Agglom Sum-Diameters, and "DivSum" for Divisive Sum-Diameters.

Table 3 gives two-phase FM bisection results obtained using the CL partitioning of the geometrically

Test Case	Net Model	Number of Clusters - k (Best dimension)								
		10	9	8	7	6	5	4	3	2
19ks	Part-Spec	14.7(10)	15.0(10)	15.8(6)	15.6(1)	15.1(1)	14.4(1)	13.1(1)	12.5(1)	17.6(1)
	Standard	21.0(2)	14.9(6)	11.1(5)	10.6(5)	9.97(5)	10.0(5)	10.2(8)	9.39(8)	11.7(7)
bm1	Part-Spec	32.2(8)	27.6(6)	30.6(10)	28.6(10)	19.8(10)	17.9(6)	11.1(4)	7.02(2)	5.80(1)
	Standard	36.5(6)	37.0(10)	31.7(6)	28.4(5)	21.3(6)	17.5(6)	11.2(5)	6.88(4)	5.80(1)
Prim1	Part-Spec	41.0(8)	34.6(7)	33.6(5)	34.4(6)	30.7(4)	27.5(3)	16.4(3)	17.4(3)	13.5(1)
	Standard	40.9(7)	45.0(10)	37.4(4)	33.6(7)	34.4(3)	28.4(3)	39.0(4)	18.1(2)	14.4(2)
Prim2	Part-Spec	12.1(7)	11.7(6)	12.0(6)	11.8(9)	11.5(5)	10.4(4)	8.96(2)	7.54(1)	5.88(1)
	Standard	13.5(7)	14.5(7)	15.7(5)	15.1(3)	12.7(6)	11.9(4)	10.8(1)	7.77(2)	8.76(1)
Test02	Part-Spec	24.0(9)	21.5(8)	21.2(7)	21.1(6)	21.2(5)	23.1(3)	23.6(3)	19.1(2)	30.1(1)
	Standard	46.3(7)	47.6(6)	48.0(7)	40.8(10)	41.7(10)	29.5(10)	26.8(9)	25.1(3)	20.1(2)
Test03	Part-Spec	23.1(4)	21.0(9)	22.4(9)	23.2(9)	22.4(3)	22.2(2)	19.3(2)	21.4(2)	16.7(3)
	Standard	38.2(6)	36.6(6)	38.8(6)	39.1(10)	35.1(1)	25.2(9)	23.2(9)	23.7(1)	10.4(1)
Test04	Part-Spec	21.7(7)	22.1(6)	23.8(6)	24.4(9)	24.3(6)	27.2(5)	27.4(3)	36.0(2)	66.1(1)
	Standard	50.5(7)	44.4(8)	43.7(8)	43.5(7)	29.6(10)	18.3(4)	21.2(4)	16.2(1)	22.1(2)
Test05	Part-Spec	9.73(10)	11.0(6)	10.6(7)	10.7(5)	11.1(4)	10.3(10)	8.78(10)	10.2(6)	10.6(1)
	Standard	24.9(7)	24.6(7)	23.3(10)	20.3(4)	19.6(10)	18.6(9)	16.4(8)	12.9(5)	12.9(1)
Test06	Part-Spec	32.0(10)	31.0(10)	32.4(10)	33.6(4)	26.4(4)	28.8(4)	25.9(2)	19.3(2)	28.6(1)
	Standard	39.2(10)	34.5(10)	26.0(9)	25.3(9)	23.3(9)	21.2(6)	18.56(6)	15.4(4)	12.7(2)

Table 2: Scaled Cost values obtained by KC using both the standard and partitioning-specific net models. Each value is the best observed over d-dimensional spectral embeddings for $1 \leq d \leq 10$; the best embedding dimension d (ties broken in favor of lower dimension) is shown in parentheses.

embedded netlist. These are compared against analogous results for (i) standard FM (on flat netlists), (ii) two-phase FM using MBC partitionings [9], and (iii) two-phase FM using RW-ST partitionings. These results (i)-(iii) are quoted from [26] with the exception of the Test05 example, for which FM min-cuts had to be regenerated due to a faulty area file used in the original experiments of [26]. For each benchmark, we use the same number of clusters k as used in [26]. We report the best two-phase FM result for the CL partitionings generated from 1- through 10-dimensional embeddings with 20 FM runs for each partitioning (the results of 200 "flat" FM runs are shown in parentheses under "Standard FM"). Our partitionings yield an average of 26.9% improvement over the flat FM partitioning results, in contrast to the 17.4% improvement obtained by RW-ST partitionings [26]. (Despite using a total of 200 FM runs, our best results were always obtained with $d \leq 3$; this is perhaps surprising in view of the correlation between k and the best embedding dimension when k is small.) Again, the results suggest that the spectral embedding is conducive to high-quality partitioning solutions.

4 The Restricted Partitioning (RP) Formulation

While a spectral embedding seems to usefully capture connectivity properties of the netlist, purely geometric partitioning approaches such as CL or KC have an obvious shortcoming: the geometric partitioning objectives (e.g., Min-Max-Diameter) have only a rough correlation to netlist partitioning objectives (e.g, Scaled Cost). When applied to a given spectral embedding, a purely geometric partitioning algorithm will always

Test Case	k	Bisection Cutsize			
		Standard FM	MBC	RW-ST	CL
19ks	737	151 (140)	156	146	124
bm1	216	65 (61)	54	58	48
PrimGA1	191	66 (66)	48	47	49
PrimSC1	191	59 (59)	61	58	49
PrimGA2	702	242 (234)	187	165	146
PrimSC2	702	235 (235)	175	159	144
Test02	445	42 (42)	42	42	42
Test03	327	84 (84)	59	71	50
Test04	317	12 (12)	20	14	12
Test05	423	37 (37)	37	28	32
Test06	477	87 (65)	83	82	63

Table 3: Utility of CL results within two-phase Fiduccia-Mattheyses partitioning. The results of 200 flat FM runs are shown in parentheses.

be handicapped since it ignores the actual netlist topology. Seemingly, the spectral embedding should serve as a partitioning *guide* and not an absolute: we therefore seek a partitioning approach that integrates both the spectral embedding and the netlist topology. One such approach has been given in the KP algorithm of Chan, Schlag and Zien [12]. (Given a spectral embedding, KP begins by clustering according to directional cosine proximity, but then uses the netlist topology to make the more difficult clustering decisions.)

The genesis of our new approach can be said to lie in the TSP heuristic proposed by Karp [32], which uses a partitioning of a planar pointset to construct a tour. Karp's heuristic tour visits every point in a particular cluster before moving to the next cluster, until all points in all clusters have been visited. 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, then (ii) minimize some partitioning objective, subject to clusters being contiguous slices of the tour.

We represent a tour $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$ of the points $V = \{v_1, v_2, \ldots, v_n\}$ by the circular *permutation* (i.e., bijection) $\pi : [1 \ldots n] \to [1 \ldots n]$. We say v_i is the j^{th} point visited in the tour if $\pi(j) = i$ (so that $v_i = v_{\pi_j}$). In other words, v_{π_1} is the first point visited, v_{π_2} is the second point visited, etc. A *slice* $[i,j]$ of π is a contiguous subset of $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$ with $[i,j] = \{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi_j}\}$ if $i \leq j$ and $[i,j] = \{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi_n}\} \cup \{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\}$ if $i > j$. We now define the "restricted" k-way partitioning problem.

Restricted k-Way Partitioning (RP): Given a circuit netlist hypergraph $H = (V, E)$, a permutation $\pi : [1..n] \to [1..n]$, cluster size bounds L and U, and a value $2 \leq k \leq |V|$, find the *k-way partitioning* P^k that optimizes a given objective function $f(P^k)$ such that the following conditions hold.

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 bounds. 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. Note that the general partitioning formulations are NP-complete for these objectives.

Given the spectrally embedded netlist modules V, our multi-way partitioning methodology constructs a tour π over V by combining a spacefilling curve heuristic [5] with a single greedy 3-Opt descent [6]. We then apply an efficient dynamic programming algorithm to solve 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. Such a restriction of the solution space (all slices $[i, j]$ have $i \leq j$) allows an $O(n)$ factor speedup, and our experiments below use linear orderings to capitalize on this complexity savings.

5 A Fast TSP Heuristic

The first part of our approach seeks a tour of the points V in the d-dimensional spectral embedding, such that proximity in d-space is preserved. Certainly, one such tour directly results from a 1-dimensional spectral embedding, namely, the sorted entries of the eigenvector that gives the 1-dimensional embedding. However, the experiments given in Section 3.3 (and confirmed below in Section 7) indicate that this single-eigenvector "tour" does not completely capture d-dimensional partitioning information that is needed when $k > 2$.

Recall that two modules that are strongly connected in the netlist will tend to be near each other in d-space. We now require that points of V that are close to each other in d-space should remain near each other with respect to π. In this way, modules that are strongly connected in the netlist will be close to each other in the tour. Additionally, when two strongly connected modules are *not* actually adjacent in the tour, they will ideally be separated by modules with which they may profitably share a cluster. From this intuition, a low cost TSP solution should adequately preserve proximity since the tour is unlikely to wander out of, and then back into, a natural cluster. However, it is not obvious which TSP heuristic to choose.

5.1 Spacefilling Curves in d-Space

Bartholdi and Platzman have used spacefilling curves (SFCs) as the basis of a provably good TSP heuristic [5]. They use the recursive construction due to Sierpinski (1912), the 2-dimensional case of which is shown in Figure 5. In the Figure, the successive approximations become 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 6 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 test case.

The Sierpinski construction appears suitable since it tends to avoid edges that are significantly longer

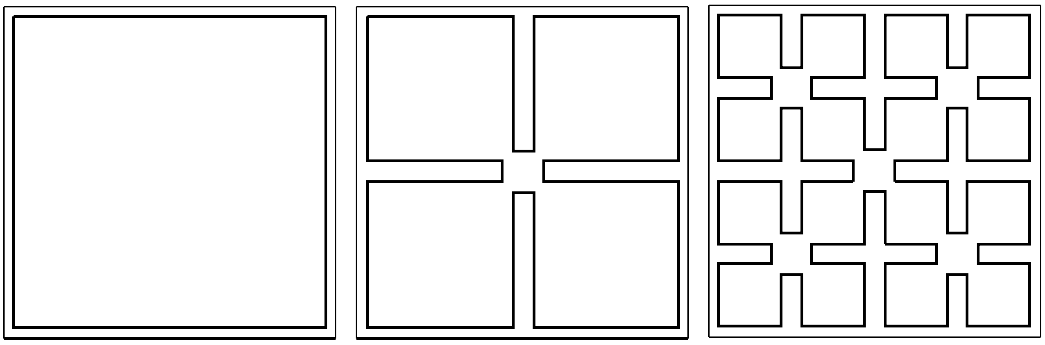
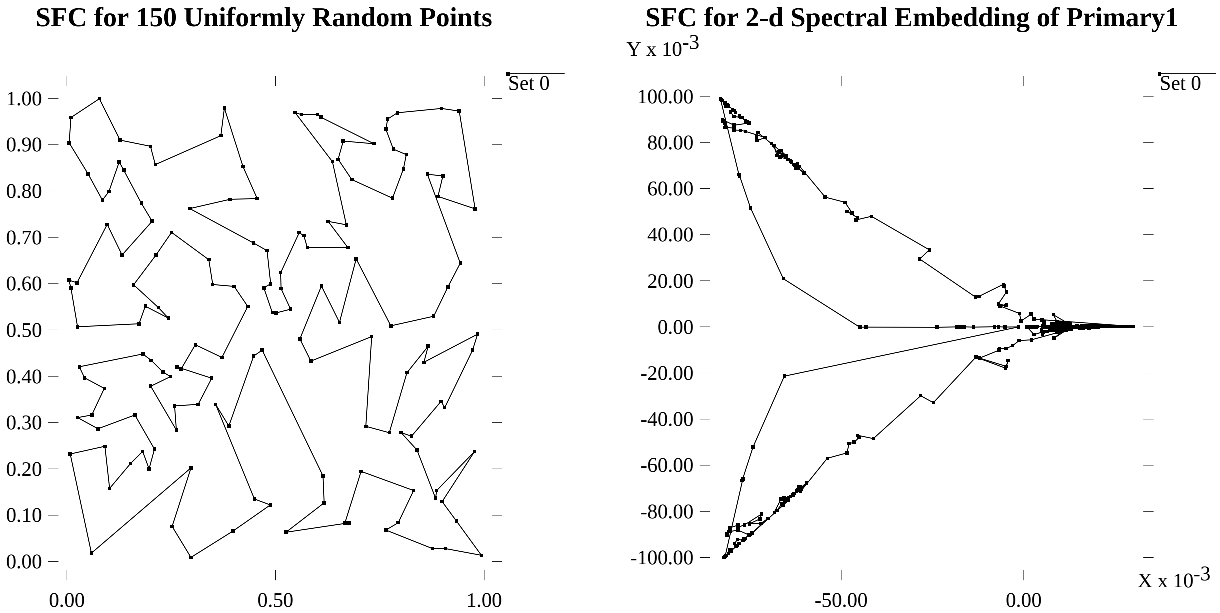


Figure 5: The Sierpinski spacefilling curve in the plane is the limit of a sequence of recursive constructions.



(a) (b)

Figure 6: The tour generated by a spacefilling curve over (a) 150 random points in the plane, and (b) the 2-dimensional spectral embedding of the Primary1 benchmark (833 modules). In (b) the x and y axes give the coordinates of the second and third eigenvectors, respectively.

than necessary (e.g., as compared with the distance to a nearest neighbor), so that if two points are adjacent in the tour, they likely will be close to each other in the d-space. On the other hand, the curve will explore one orthant entirely before moving on to the next, so that two points that are close together but in different orthants might be widely separated in the ordering. In general, this imperfect behavior seems unavoidable and some relatively long edges may occur. The Sierpinski construction is also attractive because it easily extends to higher dimensions and can be computed in just $O(dn + n\log n)$ time.

The Sierpinski construction is also attractive because it easily extends to higher dimensions and can be computed in just $O(dn + n \log n)$ time. We assume that the coordinates for the vertices of V have been scaled to lie inside the unit d-dimensional cube. Since it "fills up" the entire d-space, the Sierpinski curve can be viewed as a mapping from the unit interval into the r-space. To compute the tour, we must find the value on the unit interval that maps to each point in V, i.e., we must compute the *inverse* of the Sierpinski curve mapping. For this inverse mapping $\Theta : [0,1)^d \rightarrow [0,1)$, we require that $\Theta(u) < \Theta(v)$ if and only if u precedes v in the tour. Once $\Theta(v)$ is computed for all $v \in V$, the tour is generated by sorting the points by their Θ values, i.e., the point v with the smallest Θ value is v_{π_1}, the point v with the second smallest Θ value is v_{π_2}, etc. It turns out that for the Sierpinski construction, Θ can be evaluated in constant time (assuming r is constant) for a given point. Hence, the overall construction requires $O(dn)$ time to compute all the Θ values plus $O(n \log n)$ time to sort the values, yielding $O(dn + n \log n)$ total time complexity.

Function Theta $(X, depth)$

Input: point in $\Re^d = X$:array $[1 \ldots d]$ of real
$depth$ - granularity measure
Output: $\Theta \equiv$ real number in $[0,1)$ which indicates place on curve
Vars: Temporary d-dimensional point Y
Orthant number Q
Recursive subsolution - $SubTheta$

1. $C = Grey(1 \ldots 1)/2^d$
2. **if** $(depth = 0$ **or** $X = (1 \ldots 1))$ **then return** C
3. **for** $i = 1$ **to** d **do**
   $Y[i] = \min\{\lfloor 2 \cdot X[i] \rfloor, 1\}$
4. $Q = Grey(Y[1] \ldots Y[d])$
5. **for** $i = 1$ **to** d **do**
   $Y[i] = 2 \cdot |X[i] - 0.5|$
6. $SubTheta =$ **Theta**$(Y, depth - 1)$
7. **if** $(Q$ **mod** $2 = 1)$ **then** $SubTheta = 1 - SubTheta$
8. $Num = (Q + SubTheta - C)/2^d$
9. **return** $Num - \lfloor Num \rfloor$

Figure 7: Computation of Θ, the inverse mapping of the spacefilling curve.

Figure 7 reproduces from [5] the calculation of Θ, both for completeness and also to show its simplicity. Step 7 does not appear in [5] and we believe it was omitted as an oversight. We number the 2^d orthants from 0 to $2^d - 1$, corresponding to the order in which they are visited by the Sierpinski curve. This numbering can be represented by a d-bit Grey code since each orthant will be adjacent to the previous one; specifically, we use the Grey code that flips the rightmost bit possible without repeating an earlier sequence. In Figure 7, the function *Grey* accepts an orthant and returns a value Q between 0 and $2^d - 1$, i.e., *Grey* $(0 \ldots 0) = 0$, *Grey* $(0 \ldots 01) = 1$, *Grey* $(0 \ldots 011) = 2$, *Grey* $(0 \ldots 010) = 3$, *Grey* $(0 \ldots 0110) = 4, \ldots,$ *Grey* $(10 \ldots 0) = 2^d - 1$. The function **Theta** itself accepts parameters X (a point in $\Re^d$) and $depth$ (the minimum number of bits needed to store a point in V). It is easy to see this construction requires $O(d \cdot depth)$ time, where $depth$ reflects the depth of recursion needed to resolve the entire pointset. In other words, $depth$ is the number of significant bits needed; the constant $depth = 10$ corresponding to about three significant figures is generally sufficient.

5.2 3-Opt Post Processing

Although the SFC heuristic tends to avoid long edges, the overall cost of the tour can be quite high. Bartholdi and Platzman [5] observe that in practice, the SFC heuristic yields tours having cost within 25% of optimal for uniformly random instances in 2-space. Yet, Figure 6(b) suggests that the points in the spectral embedding are not uniformly distributed. The authors of [5] noted that other spacefilling curves may be better suited for such nonuniform distributions and even outlined a method for creating application-specific spacefilling curves. Instead, we choose to improve the SFC tour by applying a simple, greedy 3-Opt post-processing heuristic.

3-Opt uses the strategy of iterative improvement, that is, local optimization by iteratively constructing an improved solution from the current solution. The new solution is derived by finding three edges in the current tour which can be deleted, and replaced with three new edges such that the tour cost is reduced. tour with lower cost.[5] If no such edges exist, then the algorithm terminates and returns the current tour, which is also a *local minimum*.

Our 3-Opt code is based on an efficient implementation [6] that utilizes a K-d tree data structure. We also apply th method [35] [40] for restricting the number of candidate edges for the tour to $O(n)$. Intuitively, since a nearly-optimal tour should not contain any long edges, it should be possible to consider a subset of $O(n)$ short edges without significantly degrading the solution quality as compared to a method that considers all $\Theta(n^2)$ edges. Hence, we disallow any edge that connects v_i with v_j if v_j is not one of the r closest neighbors to v_i (we choose $r = 25$), thereby limiting the number of candidate edges for the tour to $O(nr)$. To find a possible 3-opt move, each of the n edges in the tour is considered in turn to be removed. For each such edge, all possible improving moves can be examined by examining $O(r^2)$ edges, and this is reduced further using the "gain" method of Bentley. Hence, finding an improving 3-opt move requires no more than $O(nr^2)$ time in the worst-case. Although the number of 3-opt moves needed to reach a local minima is exponential in the worst-case, in practice, Bentley observed between $\Theta(\log n)$ and $\Theta(n)$ moves were required for random instances. In practice, the runtimes for this post-processing are only a fraction of the runtimes needed for our overall methodology (see Table 7 in Section 7).

6 Dynamic Programming for Restricted Partitioning (DP-RP)

Since each module of the netlist is in one-to-one correspondence with a point of the spectral embedding, the Sierpinski tour over the spectral embedding naturally defines a tour over the netlist modules. With respect to this tour of modules, we now show how dynamic programming efficiently finds optimal RP solutions for a variety of partitioning objectives f.

[5] For example, 3-Opt might find indices h, i, j such that if $\{v_1, v_2, \ldots, v_h, v_{h+1}, \ldots, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots, v_n\}$ is the current tour, then $\{v_1, v_2, \ldots, v_h, v_{i+1}, v_{i+2} \ldots, v_j, v_{h+1}, v_{h+2}, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_n\}$ is a tour with lower cost. In this case, the three deleted edges $(v_h, v_{h+1}), (v_i, v_{i+1})$, and (v_j, v_{j+1}) were replaced with the edges $(v_h, v_{i+1}), (v_j, v_{h+1})$, and (v_i, v_{j+1}). In general, there will be four different sets of edges that may possibly replace the deleted edges.

6.1 Development of the DP-RP Algorithm

Assume that there exists an "intracluster" cost function $w(C)$ defined over clusters C, such that f can be written in terms of w (e.g., $f(P^k) = max_{1 \le i \le k} w(C_i)$).[6] In other words, $w(C)$ is essentially the contribution of C to the value $f(P^k)$. The cluster corresponding to slice $[i,j]$ is denoted by $C_{[i,j]}$ and we let $P^k_{[i,j]}$ denote a k-way RP solution over the slice $[i,j]$. The optimal k-way RP solution over $[i,j]$ is denoted by $\hat{P}^k_{[i,j]}$. Notice that we always have $P^1_{[i,j]} = \hat{P}^1_{[i,j]} = \{C_{[i,j]}\}$. We will use the set of $\hat{P}^k_{[i,j]}$ partitioning solutions as "building blocks" for solutions of the form $\hat{P}^{k'}_{[i',j']}$ 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 begins by computing the cost $w(C_{[i,j]})$ for each of the $(U - L + 1)n$ possible clusters; we assume the existence of a procedure Cluster_Costs that performs this operation (see Section 6.2). 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 a k-way solution is derived. Figure 8 formally describes the DP-RP algorithm.

DP-RP Algorithm	
Input:	Circuit Netlist $H(V,E)$
	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
	$m+1 \equiv$ Beginning index of possible new cluster
	$f_{best} \equiv$ Objective value for best current $\hat{P}^{k'}_{[i,j]}$

1. **for each** i,j **do** 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 $f(\hat{P}^k_{[i,i-1]}), 1 \le i \le n$

Figure 8: The DP-RP algorithm. All index manipulations are performed modulo n. The template assumes that the goal is to minimize f. To maximize f, we change steps 4, 6, and 8 in the obvious manner.

Whether the final $\hat{P}^k$ is indeed optimal depends on the objective f. For DP-RP to generate optimal solutions, a *principle of optimality* must hold: all subsolutions of an optimal RP solution must themselves be optimal RP solutions. For example, if $P^3 = \{C_1, C_2, C_3\}$ is an optimal 3-way RP solution, then $\{C_1, C_2\}$ must be an optimal 2-way RP solution over $\{C_1 \cup C_2\}$. Thus, given the set of all optimal $P^{k'-1}_{[i,j]}$, we can build the set of optimal $P^{k'}_{[i,j]}$ by examining all $P^{k'-1}_{[i,j]}$ partitionings conjoined with single clusters. In other

[6] Of course, not every objective function f can be written in this manner. For example, DP-RP cannot be applied to optimize Cluster Ratio [48].

words, $\hat{P}^{k'}_{[i,j]}$ can be expressed 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 the new partitioning that minimizes f (Step 7), every $\hat{P}^{k'}_{[i,j]}$ retained when the loop of Step 5 terminates must be optimal. DP-RP has $O(k(U-L)n^2)$ time complexity, assuming that Cluster_Costs has no worse than $O(k(U-L)n^2)$ complexity. When there are no cluster size bounds, DP-RP requires $O(kn^3)$ time.

A sufficient condition for the principal of optimality to hold is for f to be *monotone* in w. Given $P^k = \{C_1, C_2, \ldots, C_k\}$ and $Q^k = \{C'_1, C'_2, \ldots, C'_k\}$ with $w(C_i) \leq w(C'_i)$ ($w(C_i) \geq w(C'_i)$) for $1 \leq i \leq k$, we say f is monotone nondecreasing (nonincreasing) if and only if $f(P^k) \leq f(Q^k)$ ($f(P^k) \geq f(Q^k)$). The class of partitioning objectives $f(P^k)$ that are monotone in w includes the following.

- **Min-Max-Diameter:** Minimize:

$$f(P^k) = \max_{1 \leq i \leq k} w(C_i) \quad \text{with} \quad w(C_i) = diam(C_i)$$

- **Min-Sum-Diameters:** Minimize:

$$f(P^k) = \sum_{1 \leq i \leq k} w(C_i) \quad \text{with} \quad w(C_i) = diam(C_i)$$

- **Scaled Cost:** Minimize:

$$f(P^k) = \frac{1}{n(k-1)} \sum_{C_i \in P^k} w(C_i) \quad \text{with} \quad w(C_i) = \frac{|E_i|}{|C_i|}$$

- **Absorption:** Maximize:

$$f(P^k) = \sum_{1 \leq i \leq k} w(C_i) \quad \text{with} \quad w(C_i) = \sum_{\{e \in E \ | \ e \cap C_i \neq \emptyset\}} \frac{|e \cap C_i| - 1}{|e| - 1}$$

Experiments reported in [2] show that when f is Min-Max-Diameter, DP-RP significantly outperforms KC, CL and Divisive Min-Diameter for uniformly random pointsets, illustrating the powerful flexibility of the DP-RP approach. Since KC and CL applied to spectral embeddings yield reasonably good k-way netlist partitionings, DP-RP applied to spectral embeddings when f is Min-Max-Diameter should also yield fairly good solutions. Now, however, we can directly optimize netlist partitioning objectives such as Scaled Cost and Absorption, so the heuristic Min-Max-Diameter objective is unnecessary.

6.2 Optimizing Scaled Cost and Absorption

We now complete our description of the DP-RP algorithm by showing how to efficiently calculate $w(C)$ for all feasible clusters C. (Recall that we had previously assumed the existence of a Cluster_Costs procedure for computing w.) Figures 9 and 10 describe $O(nU)$ implementations of Cluster_Costs for Scaled Cost and Absorption, respectively. The complexity bound assumes that the total number of pins is $p = O(n)$, which holds for real circuit netlists due to fanout and cell I/O limits for any given technology.

Cluster_Costs (Scaled Cost)

Input: Circuit Netlist $H(V, E)$
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
$S_i \equiv \{e \in E \mid v_{\pi_i} \in e\}$

1. **for** $i = 1$ **to** n **do**
2. $\quad w(C_{[i,i]}) = |S_i|$, i.e., the outdegree of v_{π_i}
3. $\quad$ **for** $\delta = 1$ **to** U **do**
4. $\quad\quad j = ((i + \delta - 1)$ **mod** $n) + 1$
5. $\quad\quad w(C_{[i,j]}) = w(C_{[i,j-1]})$
6. $\quad\quad$ **for each** $e \in S_j$ **do**
7. $\quad\quad\quad$ **if** (e is completely contained in $C_{[i,j]}$) **then** decrement $w(C_{[i,j]})$
8. $\quad\quad\quad$ **if** (e contains no modules of $C_{[i,j-1]}$) **then** increment $w(C_{[i,j]})$
9. **for each** $w(C_{[i,j]})$ computed **do** $w(C_{[i,j]}) = \frac{w(C_{[i,j]})}{j-i+1}$

Figure 9: Cluster_Costs (Scaled Cost).

For Scaled Cost (Figure 9), Cluster_Costs returns $w(C_{[i,j]}) \equiv \frac{|E_{[i,j]}|}{|C_{[i,j]}|}$ for each possible cluster $C_{[i,j]}$, where $E_{[i,j]}$ is the "outdegree" of cluster $C_{[i,j]}$. Steps 1-8 calculate $|E_{[i,j]}|$ for each cluster, and $w(C_{[i,j]})$ is computed in Step 9. Given the value of $w(C_{[i,j-1]})$, we 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 7), or whether any previously uncut nets become cut (Step 8).

The desired $O(nU)$ time complexity hinges on Steps 7 and 8 being executed in constant time. To accomplish this, we maintain an array Cut_Sigs over the signal nets, where $Cut\_Sigs[e] = 1$ if e is cut by our current cluster and $Cut\_Sigs[e] = 0$ otherwise. In Step 2, we start with the initial cluster $C_{[i,i]}$ and set $Cut\_Net[e] = 1$ if e contains module v_{π_i}, and $Cut\_Net[e] = 0$ otherwise. We also keep an array $Count[e]$, that records the number of pins of net e in the current cluster, and an array $Pins[e]$, which is the total number of pins of net e. To execute Step 7, we verify the **if** condition using the $Count$ array. If $Count[e] = Pins[e]$, then e is completely contained in the current cluster, and we set $Cut\_Net[e] = 0$ since this net is no longer cut. To execute Step 8, we check that $Cut\_Net[e] = 0$, in which case adding v_{π_j} to the current cluster will cause s to be cut. For these newly cut nets we set $Cut\_Net[e] = 1$ and $Count[e] = 1$. Step 7-8 are executed exactly pU times, and since these steps require only constant time, Cluster_Costs (Scaled Cost) runs in $O(nU)$ time.

For Absorption (Figure 10), Cluster_Costs returns $w(C_{[i,j]}) \equiv \sum_{(e \in E \mid |e \cap C_i| \neq 0)} \frac{|e \cap C_i| - 1}{|e| - 1}$ for each possible cluster $C_{[i,j]}$. For a given $w(C_{[i,j]})$, Cluster_Costs initially assigns $w(C_{[i,j]}) = w(C_{[i,j-1]})$, thereby giving credit to the nets "absorbed" by modules $v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi_{j-1}}$. Steps 6-8 compute the number of nets absorbed when v_{π_j} is added to $C_{[i,j-1]}$. Steps 6-8 require constant time and are executed exactly pU times, hence Cluster_Costs (Absorption) also has $O(nU)$ time complexity.

Cluster_Costs (Absorption)

Input: Circuit Netlist $H(V, E)$
Permutation $\pi = \{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
$L, U \ \equiv$ Lower and upper cluster size bounds

Output: $w(C_{[i,j]}) \ \equiv \sum_{(e \in E \ | \ |e \cap C_i| \neq 0)} \frac{|e \cap C_i| - 1}{|e| - 1}$ for every possible cluster $C_{[i,j]}$

Vars: $Count[e] \ \equiv$ the number of pins of e in the current cluster
$S_i \ \equiv \ \{e \in E \mid v_{\pi_i} \in e\}$

for $i = 1$ **to** n **do**
  $w(C_{[i,i]}) = 0$
  for each $e \in E$ **do** $Count[e] = 0$
  for $j = i$ **to** $\min\{i + U, n\}$ **do**
    if $j \neq i$ **then** $w(C_{[i,j]}) = w(C_{[i,j-1]})$
    for each $e \in S_j$ **do**
      $Count[e] = Count[e] + 1$
      if $(Count[e] > 0)$ **then**
        $w(C_{[i,j]}) = w(C_{[i,j]}) + \frac{1}{|e|-1}$

Figure 10: Cluster_Costs (Absorption).

6.3 Linear Orderings

So far, we have considered the RP formulation where both Rules 1(a) and 1(b) apply. In this case, DP-RP may require up to $O(kn^3)$ complexity, which is prohibitive for practical netlist sizes. However, recall that eliminating rule 1(b) changes the tour into a linear ordering, restricting the solution space but allowing a factor n speedup. Figure 11 illustrates how DP-RP can be modified to solve a given linear ordering with $O(kn(U - L))$ time complexity (i.e., $O(kn^2)$ when there are no cluster size bounds). The speedup arises since we are guaranteed that some cluster begins with index π_1. Thus, for each value of k', we need record only $O(n)$ subsolutions of the form $P^{k'}_{[1,j]}$ instead of the $\Theta(n^2)$ optimal subpartitioning solutions.

In practice, there exists the question of how to derive a linear ordering from a tour. To generate the experimental results in the next section, we initially generated a linear ordering from the tour simply by removing the tour's longest edge. Since this seemed somewhat crude, 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 using any of the three linear orderings.

7 Experimental Results and Conclusions

Our experiments set $L = 1$ and $U = n$ in order to consider the full range of solutions. For each test case, we computed d-dimensional spectral embeddings for $1 \leq d \leq 10$ for both the standard net model and the partitioning-specific net model, and then computed a heuristic SFC + 3-Opt ordering for each embedding. To enable comparison with [12], we also computed orderings using the net model of Frankle and Karp [20],

DP-RP Algorithm for Linear Orderings

Input: Circuit Netlist $H(V, E)$
Linear Ordering $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
$L, U \equiv$ Lower and upper cluster size bounds
$k \equiv$ Number of clusters
Output: $\hat{P}^k \equiv$ Optimal RP solution (without Condition 1(b)
Vars: $\hat{P}^{k'}_{[i,j]} \equiv$ Subsolutions
$k' \equiv$ Index denoting current partitioning size
$m+1 \equiv$ Beginning index of possible new cluster
$f_{best} \equiv$ Objective value for best current $\hat{P}^{k'}_{[i,j]}$

1. **for each** i, j **do** compute $f(\hat{P}^1_{[i,j]}) = w(C_{[i,j]})$ using **Cluster_Costs**
2. **for** $k' = 2$ **to** k **do**
3.   **for** $j = 1$ **to** n **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** $\hat{P}^k = \hat{P}^k_{[1,n]}$

Figure 11: DP-RP Algorithm for Linear Orderings. The template assumes the goal is to minimize f. To maximize f, simply change steps 4 and 6 in the obvious manner.

which uses uniform edge weight $(\frac{2}{p})^{\frac{3}{2}}$ in representing a p-pin net.

In our first set of experiments, we executed DP-RP on each ordering for $k = 2$ through $k = 10$, using the minimum Scaled Cost objective and the three clique net models. Table 4 shows the the lowest Scaled Cost values, along with the embedding dimensions that yielded these partitioning solutions (again, ties were broken in favor of lower dimensions). The partitioning-specific net model averaged 8.9% improvement over the standard net model and 2.7% improvement over the Frankle net model. There were also fluctuations in the relative utility of the net models, e.g., compared to the Frankle net model, the partitioning-specific net model averaged 13.5% improvement for 19ks, but 5.3% disimprovement for Test03. Thus, while we may conclude that the standard net model is not well-suited for partitioning (and that the partitioning-specific net model is better), it is unclear which among the many possible net models is best. We leave this question open for future study.

Our second set of experiments compared the DP-RP methodology against previous methods [2] [12] which address k-way partitioning for minimum Scaled Cost. In Table 5, we use the notation DP-RP(d) to denote the best Scaled Cost values observed for the k-way partitionings that DP-RP constructed from 1- through d-dimensional embeddings. In other words, for each $1 \leq i \leq d$, we computed the spectral embedding with eigenvectors $\vec{\mu}_2, \ldots, \vec{\mu}_{i+1}$ and ran DP-RP on this embedding. Thus, e.g., DP-RP(10) gives the best Scaled Cost results observed over all ten distinct embeddings.[7] We did not conduct experiments for Absorption, since this metric is more appropriate for $k = \Theta(n)$ and since no previous work has reported any Absorption partitioning results. Table 5 compares DP-RP(10) to KC, SFC [2], KP [12] and successive

[7]With the KP algorithm of [12], only the k-dimensional spectral embeddings were used to generate a k-way partitioning.

Test Case	Net Model	Number of Clusters - k (Best dimension)								
		10	9	8	7	6	5	4	3	2
19ks	Part-Spec	8.88(9)	8.58(9)	8.22(9)	7.68(9)	7.12(9)	6.57(9)	5.64(7)	5.45(7)	4.82(5)
	Frankle	9.65(8)	9.37(8)	9.12(8)	8.80(8)	8.62(8)	8.46(4)	6.58(4)	6.28(4)	5.67(4)
	Standard	11.1(4)	9.94(4)	8.95(4)	8.53(4)	8.08(4)	7.40(4)	7.12(2)	6.24(6)	5.87(6)
bm1	Part-Spec	23.4(8)	21.8(8)	20.0(8)	17.9(8)	16.8(8)	14.8(6)	9.02(8)	6.61(2)	5.53(2)
	Standard	25.7(8)	24.3(8)	22.8(8)	21.0(8)	19.4(3)	16.6(4)	12.3(4)	6.61(2)	5.53(2)
	Frankle	25.8(8)	24.5(8)	22.7(6)	20.4(6)	18.0(6)	15.9(6)	13.1(5)	6.61(3)	5.53(3)
Prim1	Part-Spec	40.4(6)	38.5(3)	34.0(3)	31.2(3)	28.2(3)	24.5(3)	20.4(3)	14.7(2)	13.5(2)
	Standard	38.0(8)	36.0(10)	33.8(10)	31.1(10)	29.2(3)	24.3(10)	20.8(10)	14.5(3)	13.4(3)
	Frankle	38.4(5)	35.4(4)	33.2(4)	30.4(4)	28.7(4)	26.4(4)	23.4(3)	18.6(3)	13.5(1)
Prim2	Part-Spec	12.7(9)	12.2(9)	11.6(9)	10.8(4)	10.0(4)	9.20(2)	8.18(2)	6.98(9)	4.77(4)
	Standard	12.5(7)	11.6(7)	11.0(7)	10.4(7)	9.72(7)	9.34(7)	8.36(7)	7.26(2)	4.64(2)
	Frankle	12.0(9)	11.6(9)	11.3(9)	11.1(9)	9.85(3)	8.46(3)	7.93(3)	7.24(8)	4.58(2)
Test02	Part-Spec	20.2(8)	19.4(8)	18.6(8)	17.6(8)	17.2(8)	15.9(3)	14.2(5)	12.0(5)	9.91(5)
	Standard	25.4(6)	23.7(6)	21.7(6)	19.7(6)	18.9(6)	18.3(6)	14.6(6)	13.7(10)	9.22(2)
	Frankle	23.0(10)	22.1(10)	21.1(10)	19.9(6)	17.9(6)	16.1(6)	15.2(6)	12.9(2)	8.07(2)
Test03	Part-Spec	17.3(6)	16.2(6)	15.5(6)	14.9(6)	14.1(6)	13.5(6)	12.7(6)	11.4(6)	8.98(3)
	Standard	19.3(8)	18.7(8)	17.7(8)	16.9(8)	16.2(8)	15.0(8)	13.8(6)	10.8(6)	10.1(6)
	Frankle	16.4(5)	15.5(5)	15.2(5)	14.4(5)	13.7(5)	12.6(5)	11.3(8)	10.4(8)	8.98(3)
Test04	Part-Spec	15.1(4)	14.1(4)	13.1(4)	12.3(5)	11.0(5)	10.2(5)	7.33(5)	7.11(5)	5.78(1)
	Standard	15.9(6)	15.1(6)	13.5(6)	12.3(6)	11.4(6)	10.1(6)	8.50(6)	7.52(6)	5.78(2)
	Frankle	13.6(8)	13.4(8)	12.2(8)	11.0(8)	10.1(8)	9.15(8)	7.98(8)	7.34(8)	5.78(6)
Test05	Part-Spec	7.44(5)	7.19(5)	6.96(5)	6.44(5)	6.24(5)	5.99(8)	5.31(9)	4.78(9)	3.09(9)
	Standard	12.9(9)	12.7(9)	11.3(5)	10.3(5)	9.37(5)	5.98(5)	5.41(5)	4.51(5)	3.09(5)
	Frankle	8.15(8)	7.88(8)	7.41(4)	6.75(4)	6.46(4)	5.61(4)	5.33(4)	5.03(8)	3.12(1)
Test06	Part-Spec	17.9(9)	17.0(6)	16.5(6)	15.7(5)	13.5(7)	12.7(3)	11.7(7)	10.3(3)	8.80(2)
	Standard	22.6(10)	20.9(10)	19.7(10)	18.3(10)	16.2(10)	16.5(10)	14.0(2)	13.5(10)	7.82(4)
	Frankle	19.7(10)	18.5(10)	17.1(10)	15.9(10)	15.1(10)	14.2(10)	12.0(10)	10.6(4)	8.21(5)
Average Part-Spec	vs Frankle	+2.19	+2.76	+3.20	+3.12	+3.39	+1.57	+5.02	+5.12	-2.02
	vs Standard	+13.3	+12.5	+11.0	+10.6	+11.3	+7.52	+8.69	+5.02	+0.60

Table 4: Scaled Cost results for DP-RP, using the partitioning-specific, standard, and Frankle clique net models. The embedding dimension which yielded the lowest Scaled Cost (ties broken in favor of lower dimension) is given in parentheses. The last two rows give the average percent improvement of the partitioning-specific model over the other two models for each value of k.

spectral bipartitioning (SB); the last is equivalent to recursive application of the EIG1 method of Hagen and Kahng [25]. The codes for KP and SB were obtained from the authors of [12]. All results in Table 5 were generated using the partitioning-specific net model. Notice that the SFC results are equivalent to the DP-RP(10) results without the 3-Opt post-processing step (hence, the 13.1% average improvement of DP-RP(10) over SFC reflects the contribution of the single 3-Opt descent in constructing the ordering). Other experiments also showed that the SFC + 3-Opt methodology was more effective than 3-Opt starting from a random initial tour.

Since the results in [12] are based on the Frankle net model, Table 6 presents similar comparisons between DP-RP(10), KP and SB.[8] The last sets of rows in Tables 5 and 6 give the average percent improvement of

[8]Note that the KP and SB results given in Tables 5 and 6 are considerably different than those reported in [12]. The SB and KP experiments performed in [12] removed nets with more than 99 pins before the eigenvectors were computed; this was due to computational hardware limitations [11]. For some of the netlists (Test03, Test04, and Test06), removing the large nets disconnected the graph, implying $\lambda_2 = 0$. Since SB uses $\vec{\mu_2}$ to determine its vertex ordering, and since eigenvectors with eigenvalue zero are degenerate, the SB results were much worse than they would have been had the large nets not been

DP-RP(10) versus the other methods for each value of k; the last column in each table gives the average improvement achieved by DP-RP(10) for each benchmark. Overall, DP-RP(10) averages 13.1%, 17.9% and 10.0% improvement over SFC, KP, and SB respectively when using the partitioning-specific net model and 10.8% and 9.3% improvement over KP and SB when using the Frankle net model. We also observe that DP-RP(10) gives lower Scaled Cost values on average over the range of k values. Interestingly, KP results generated using the Frankle net model were on average 3.0% better than the results generated with the partitioning-specific net model, which only emphasizes that the proper choice of net model is very much an open issue.

We also observe that the 1-dimensional heuristic ordering indeed captures partitioning information from the *multi*-dimensional spectral embedding that is not available from the standard 1-dimensional spectral embedding $\vec{\mu_2}$. To show the utility of our orderings, we have provided the DP-RP(1) data of Table 5, which gives results for DP-RP when run on the ordering corresponding to $\vec{\mu_2}$ (the 3-Opt post-processing clearly does not change this ordering). In Table 5, some DP-RP(10) entries are identical to corresponding DP-RP(1) entries, indicating 1-dimensional orderings that are at least as good as the SFC + 3-Opt orderings derived from multi-dimensional spectral embeddings. However, we have observed that on average DP-RP(2-10) produces Scaled Cost 21.2% lower than DP-RP(1) (ranging from 13.7% to 43.9% higher for bm1 and Primary1 respectively), confirming the utility of multiple eigenvectors in constructing multi-way partitionings. Finally, we note that the overall DP-RP methodology is reasonably efficient, as documented in Table 7.

In conclusion, we have shown that spectral embeddings of a netlist can be an effective basis for multi-way system partitioning. We have confirmed the utility of embeddings using the eigenvectors of the Laplacian when integrated with purely geometric partitioning objectives and algorithms. Since a strictly geometric partitioning methodology can only heuristically optimize topological partitioning objectives such as Scaled Cost and Absorption, we have also developed an approach that uses both the spectral embedding and the netlist topology within the restricted partitioning formulation. Our approach uses a spacefilling curve and a 3-Opt heuristic to achieve an ordering that captures information from the multi-dimensional embedding. We split the ordering into k clusters via dynamic programming, obtaining an optimal RP solution. User-specified cluster size bounds can be handled transparently. Experimental results show significant improvements over previous k-way partitioning methods for the Scaled Cost objective for $2 \leq k \leq 10$.

removed. The SB results presented in Tables 5 and 6 are respectively 11.0% and 9.5% better on average (over the applicable seven test cases) than the SB results in [12]. It also seems that thresholding hurt KP performance for the Frankle net model (by 3.3%), but that results for the partitioning-specific net model are 4.4% worse than the results reported in [12] due to the very poor results for small k for the 19ks, Test02, Test03, Test04, and Test06 benchmarks.

Test	ALG	Number of Clusters - k									Avg %
Case		10	9	8	7	6	5	4	3	2	improv
19ks	DP-RP(10)	8.88	8.58	8.22	7.68	7.12	6.57	5.64	5.45	4.82	+0.00
	DP-RP(1)	15.1	14.3	13.8	13.2	12.2	11.1	9.51	7.48	6.25	+37.4
	SFC	15.1	14.3	13.8	13.2	12.2	11.1	8.37	7.48	5.44	+35.2
	KP	12.6	8.72	10.8	7.37	7.04	7.42	11.4	35.2	17.6	+29.9
	KC	14.7	15.0	15.8	15.6	15.1	14.4	13.1	12.5	17.6	+52.7
	SB	13.9	11.6	9.15	8.74	8.87	7.00	6.51	6.45	6.35	+18.1
bm1	DP-RP(10)	23.4	21.8	20.0	17.9	16.8	14.8	9.02	6.61	5.53	+0.00
	DP-RP(1)	35.2	32.1	28.6	25.4	20.4	15.1	9.02	6.61	5.53	+16.1
	SFC	24.8	22.8	20.7	18.0	14.4	11.5	8.89	6.61	5.53	-3.65
	KP	24.0	19.8	18.2	15.7	12.1	11.4	8.67	6.61	5.53	-11.6
	KC	32.2	27.6	30.6	28.6	19.8	17.9	11.1	7.02	5.80	+20.2
	SB	33.3	31.1	26.9	22.7	17.0	11.3	8.63	6.61	5.53	+8.01
Prim1	DP-RP(10)	40.4	38.5	34.0	31.2	28.2	24.5	20.4	14.7	13.4	+0.00
	DP-RP(1)	80.4	75.4	68.7	61.6	54.9	47.1	38.4	31.7	13.4	+44.0
	SFC	38.9	36.7	35.2	31.7	28.8	26.0	21.8	14.6	13.4	+1.90
	KP	45.9	36.0	32.9	32.2	32.4	24.1	19.5	14.7	13.5	+1.36
	KC	41.0	34.6	33.6	34.4	30.7	27.5	16.4	17.4	13.5	+1.02
	SB	53.1	44.6	40.5	38.1	32.4	28.1	23.9	17.0	13.4	+14.0
Prim2	DP-RP(10)	12.7	12.2	11.6	10.8	9.99	9.20	8.18	6.98	4.77	+0.00
	DP-RP(1)	19.5	18.2	16.8	15.4	13.5	11.2	9.45	7.18	5.55	+22.5
	SFC	13.7	13.3	12.8	12.1	11.0	9.43	7.95	6.86	5.05	+5.36
	KP	13.7	12.3	12.9	12.7	11.3	10.6	10.9	9.60	5.42	+13.6
	KC	12.1	11.7	12.0	11.8	11.5	10.4	8.96	7.54	5.88	+6.92
	SB	11.2	10.9	10.1	9.73	9.33	8.33	7.85	7.69	5.55	-5.51
Test02	DP-RP(10)	20.2	19.4	18.6	17.6	17.2	15.9	14.2	12.0	9.91	+0.00
	DP-RP(1)	34.3	31.9	29.6	27.8	25.5	22.4	19.7	16.0	12.1	+31.9
	SFC	25.5	24.1	22.8	20.9	18.5	16.1	13.4	10.9	8.07	+4.88
	KP	21.4	23.3	21.6	18.7	19.5	20.2	16.5	17.8	30.1	+21.0
	KC	24.0	21.5	21.2	21.1	21.2	23.1	23.6	19.1	30.1	+27.6
	SB	31.3	31.4	20.2	29.8	28.6	28.8	18.4	18.2	12.4	+31.6
Test03	DP-RP(10)	17.3	16.2	15.5	14.9	14.1	13.5	12.7	11.4	8.98	+0.00
	DP-RP(1)	30.6	28.3	25.4	23.1	21.5	19.3	16.5	14.9	11.8	+32.8
	SFC	22.6	21.1	19.2	17.1	16.2	15.2	14.3	13.0	10.2	+15.4
	KP	19.7	20.4	20.0	19.7	22.1	21.8	15.9	13.1	31.2	+28.7
	KC	23.1	21.0	22.4	23.2	22.4	22.2	19.3	21.4	16.7	+35.3
	SB	20.3	19.9	17.7	17.0	16.7	17.6	16.0	14.3	11.9	+18.1
Test04	DPRP(10))	15.1	14.1	13.1	12.2	11.0	10.2	7.33	7.11	5.78	+0.00
	DP-RP(1)	37.9	34.3	30.2	25.1	19.6	15.4	11.6	8.44	5.78	+39.7
	SFC	22.2	19.9	17.8	17.6	16.5	15.1	11.6	8.19	5.78	+26.0
	KP	14.8	16.5	14.1	11.7	14.5	13.9	9.05	13.0	66.1	+24.6
	KC	21.7	22.1	23.8	24.4	24.3	27.2	27.4	36.0	66.1	+58.2
	SB	14.3	12.5	11.8	11.1	10.2	10.3	9.08	8.65	5.85	-0.88
Test05	DP-RP(10)	7.44	7.19	6.96	6.44	6.24	5.99	5.31	4.78	3.09	+0.00
	DP-RP(1)	11.7	10.8	9.60	8.78	7.96	7.04	6.33	4.95	3.09	+20.0
	SFC	9.88	8.66	8.06	7.84	7.32	6.56	5.49	4.90	3.09	+11.4
	KP	7.89	7.96	7.47	6.78	6.81	7.04	7.00	5.88	5.08	+14.7
	KC	9.73	11.0	10.6	10.7	11.1	10.3	8.78	10.2	10.6	+42.4
	SB	7.50	6.82	6.70	6.12	5.38	4.97	4.26	4.06	3.09	-10.3
Test06	DP-RP(10)	17.9	17.0	16.5	15.7	13.5	12.7	11.7	10.3	8.80	+0.00
	DP-RP(1)	31.7	29.5	27.0	24.2	22.9	21.5	19.2	15.6	14.3	+39.7
	SFC	27.1	25.1	23.7	20.2	18.4	16.5	13.7	11.3	9.21	+21.8
	KP	24.4	24.1	25.9	27.6	23.3	22.9	13.0	18.5	28.6	+38.4
	KC	32.0	31.0	32.4	33.6	26.4	28.8	25.9	19.3	28.6	+51.9
	SB	17.8	17.3	16.0	15.6	16.2	17.3	19.9	15.6	14.3	+17.1
Average	vs SFC	+20.6	+18.8	+18.3	+17.1	+14.6	+10.8	+10.5	+5.71	+1.17	+13.1
DP-RP(10)	vs KP	+11.1	+8.49	+11.9	+8.22	+11.9	+15.4	+17.1	+29.5	+47.0	+17.9
	vs SB	+13.5	+10.9	+4.37	+8.66	+8.34	+5.85	+11.0	+14.1	+13.6	+10.0

Table 5: Scaled Cost ($\times 10^5$) comparisons using the partitioning-specific net model.

Test Case	ALG	Number of Clusters - k									Avg % improv
		10	9	8	7	6	5	4	3	2	
19ks	DP-RP(10)	9.65	9.37	9.12	8.80	8.62	8.46	6.58	6.28	5.67	+0.00
	KP	9.09	9.37	10.7	9.52	9.00	9.00	6.95	6.58	6.20	+4.98
	SB	11.1	9.53	9.18	9.12	7.90	6.34	6.28	6.65	6.32	-1.40
bm1	DP-RP(10)	25.8	24.5	22.7	20.4	18.0	15.9	13.1	6.61	5.53	+0.00
	KP	27.5	23.1	18.9	18.2	12.5	10.7	8.67	6.61	5.53	-19.5
	SB	43.4	36.1	30.2	24.7	19.6	13.8	8.67	6.61	5.53	+6.31
Prim1	DP-RP(10)	38.4	35.4	33.2	30.4	28.7	26.4	23.4	18.6	13.5	+0.00
	KP	44.7	41.3	32.3	33.2	31.3	29.9	21.2	14.7	13.5	+1.90
	SB	59.3	56.2	51.0	46.6	43.2	40.3	38.9	22.5	13.5	+29.7
Prim2	DP-RP(10)	12.0	11.6	11.3	11.1	9.85	8.46	7.93	7.24	4.58	+0.00
	KP	15.0	15.2	13.5	11.0	10.5	10.1	9.23	7.25	4.64	+10.8
	SB	11.1	10.6	10.4	9.55	8.44	8.47	7.67	6.56	4.78	-7.62
Test02	DP-RP(10)	23.0	22.1	21.1	19.9	17.9	16.1	15.2	12.9	8.07	+0.00
	KP	24.4	22.6	22.1	19.1	19.3	18.7	17.4	12.0	9.26	+5.27
	SB	32.7	26.8	24.0	21.1	19.2	15.7	13.6	11.1	8.07	+4.58
Test03	DP-RP(10)	16.4	15.5	15.2	14.4	13.7	12.6	11.3	10.4	8.98	+0.00
	KP	20.3	22.4	19.1	17.3	18.4	22.8	19.9	14.7	9.45	+26.1
	SB	19.0	18.6	18.9	17.3	15.4	14.1	13.9	12.8	8.98	+14.0
Test04	DP-RP(10)	13.6	13.4	12.2	11.0	10.1	9.15	7.98	7.34	5.78	+0.00
	KP	18.3	16.3	15.0	12.2	13.7	12.5	8.98	12.0	6.74	+21.0
	SB	15.8	14.3	14.0	12.2	11.8	9.13	8.04	6.79	5.85	+5.66
Test05	DP-RP(10)	8.15	7.88	7.41	6.75	6.46	5.61	5.33	5.03	3.12	+0.00
	KP	10.8	10.6	9.28	6.80	6.60	6.69	7.81	7.34	4.52	+20.4
	SB	11.2	10.2	7.56	6.86	5.96	5.93	5.96	4.48	3.12	+5.43
Test06	DP-RP(10)	19.7	18.5	17.1	15.9	15.1	14.2	12.0	10.6	8.21	+0.00
	KP	21.0	20.9	19.7	18.5	19.1	19.1	18.0	18.2	28.6	+26.4
	SB	22.4	20.7	20.6	21.7	23.1	19.0	21.1	17.4	13.0	+26.3
Average	vs KP	+12.8	+13.3	+9.46	+4.47	+6.32	+12.5	+10.0	+12.4	+11.8	+10.8
DP-RP(10)	vs SB	+19.7	+15.0	+12.8	+11.1	+8.26	+2.72	+4.66	+3.75	+5.83	+9.32

Table 6: Scaled Cost ($\times 10^5$) comparisons using the Frankle net model.

Test Case	Step	Embedding Dimension - d									
		10	9	8	7	6	5	4	3	2	1
19ks	Embedding	326	310	321	238	252	226	207	180	168	189
	SFC+3-opt	125	129	123	123	123	121	123	135	126	122
	DP-RP	161	161	161	161	161	161	161	161	161	161
	Total	612	600	605	522	536	508	491	476	455	472
bm1	Embedding	29	27	25	25	20	15	16	11	10	8
	SFC+3-opt	14	14	13	13	12	12	10	10	7	7
	DP-RP	14	14	14	14	14	14	14	14	14	14
	Total	57	55	52	52	46	41	40	35	31	29
Prim1	Embedding	23	19	17	17	14	12	11	11	8	8
	SFC+3-opt	12	12	12	11	10	10	9	9	7	7
	DP-RP	12	12	12	12	12	12	12	12	12	12
	Total	47	43	41	40	36	34	32	32	27	27
Prim2	Embedding	133	98	77	81	64	56	48	51	36	35
	SFC+3-opt	127	130	117	116	115	111	114	101	82	77
	DP-RP	185	185	185	185	185	185	185	185	185	185
	Total	445	413	379	382	364	352	347	337	303	297
Test02	Embedding	221	208	179	148	142	134	132	114	104	84
	SFC+3-opt	48	47	47	48	43	40	38	37	28	33
	DP-RP	55	55	55	55	55	55	55	55	55	55
	Total	324	310	281	251	240	229	225	206	187	172
Test03	Embedding	140	129	124	105	102	94	95	93	72	63
	SFC+3-opt	47	46	44	45	41	38	35	32	22	20
	DP-RP	98	98	98	98	98	98	98	98	98	98
	Total	285	273	266	248	241	230	228	223	192	181
Test04	Embedding	219	211	195	188	172	164	147	143	137	120
	SFC+3-opt	44	42	42	40	37	35	32	30	21	17
	DP-RP	90	90	90	90	90	90	90	90	90	90
	Total	353	343	327	318	299	289	269	263	248	227
Test05	Embedding	502	505	452	427	419	413	398	360	352	352
	SFC+3-opt	111	106	105	107	99	107	92	85	69	61
	DP-RP	137	137	137	137	137	137	137	137	137	137
	Total	750	748	694	671	655	657	627	582	558	550
Test06	Embedding	181	186	168	161	144	134	127	132	98	93
	SFC+3-opt	62	60	63	56	54	54	46	40	45	38
	DP-RP	83	83	83	83	83	83	83	83	83	83
	Total	326	329	314	300	281	271	256	255	226	214

Table 7: Runtimes in seconds on a Sun Sparc 10, for a single run of DP-RP for each of the ten d-dimensional embeddings (constructed from $\vec{\mu}_2, \ldots, \vec{\mu}_{d+1}$). Each Embedding entry gives the time required to generate d eigenvectors, and a DP-RP entry gives the time needed to generate 2- through 10-way partitioning solutions.

Acknowledgements

We thank Pak K. Chan, Martine Schlag and Jason Zien for past research discussions, for the use of the LASO interface written by Martine Schlag, and for the use of their KP and SB codes. Ken D. Boese supplied the 3-Opt optimization code. Lars Hagen and Jen-Hsin Huang developed the ideas behind the partitioning-specific net model. The anonymous reviewers provided many comments which substantially improved this work, particularly in the experimental design. Part of this work (ABK) was performed in part during a Spring 1993 sabbatical visit to UC Berkeley; support from NSF MIP-9117328 and the hospitality of Professor Ernest S. Kuh and his research group is gratefully acknowledged.

References

[1] C. J. Alpert and A. B. Kahng, "Geometric Embeddings for Faster (and Better) Multi-way Netlist Partitioning," UCLA CSD TR-920052, October 1992 (abbreviated version appeared in *Proc. ACM/IEEE Design Automation Conf.* June 1993, pp. 743-748).

[2] C. J. Alpert and A. B. Kahng, "Multi-Way Netlist Partitioning Using Spacefilling Curves," UCLA CSD TR-930016, June 1993 (abbreviated version appeared in *Proc. ACM/IEEE Design Automation Conf.* June 1994, pp. 652-657).

[3] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey," to appear in *Integration: the VLSI Journal*, 1995.

[4] S. T. Barnard and H. D. Simon, "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems", NASA AMES Research Center *technical report* RNR-92-033, Nov. 1992.

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

[6] J. L. Bentley, "Fast Algorithms for Geometric Traveling Salesman Problems", *ORSA Journal on Computing*, Vol. 4, No. 4, pp. 387-410, 1992.

[7] D. Bertsimas and M. Grigni, "Worst-Case Examples for the Spacefilling curve heuristic for the Euclidean Traveling Salesman Problem", *Operations Research Letters*, Vol. 8, No. 5, pp. 241-244, Oct. 1989.

[8] J. P. Benzécri, "Construction d'une Classification Ascendante Hiérarchique par la Rechereche en Chaine des Voisins Réciproques", *Les Cahiers de l'Analyse des Données* (VII)2, pp. 209-218, 1982.

[9] T. N. Bui, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms", in *Proc. ACM/IEEE Design Automation Conf.*, June 1989, pp. 775-778.

[10] P. Brucker, "On the Complexity of Clustering Problems", *Optimization and Operations Research*, pp. 45-54, 1977.

[11] P. K. Chan, *personal communication*, November, 1994.

[12] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio-Cut Partitioning and Clustering", *IEEE Trans. on CAD*, Vol. 13, No. 9, pp. 1088-1096, Sept. 1994. (Some experimental results are quoted from: J. Zien, "Spectral K-Way Ratio Cut Graph Partitioning", M.S. Thesis, Computer Engineering Dept., UC Santa Cruz, March 1993.)

[13] H. R. Charney and D. L. Plato, "Efficient Partitioning of Components", in *Proc. IEEE Design Automation Workshop*, 1968, pp. 16-0 – 16-21.

[14] C. K. Cheng and Y. C. Wei, "An Improved Two-Way Partitioning Algorithm with Stable Performance", *IEEE Trans. on CAD*, Vol. 10, No. 12, pp. 1502-1511, Dec. 1991.

[15] H. L. Davidson and E. Kelly, *personal communication* at Sun Microsystems Labs, July 1993.

[16] C.-L. Ding, C.-Y. Ho, and M. J. Irwin, "A New Optimization Driven Clustering Algorithm for Large Circuits (Extended Abstract)", *Proc. 4th ACM/SIGDA Physical Design Workshop*, 1993, pp. 13–19.

[17] W. E. Donath, "Logic Partitioning", in B. Preas and M. Lorenzetti, eds., *Physical Design Automation of VLSI Systems*, Benjamin-Cummings, 1988, pp. 65-86.

[18] T. Feder and D. H. Greene, "Optimal Algorithms for Approximate Clustering", *Proc. 20th Annual ACM Symp. on Theory of Computing*, 1988, pp. 434-444.

[19] 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.

[20] J. Frankle and R. M. Karp. "Circuit Placement and Cost Bounds by Eigenvector Decomposition", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1986, pp. 414-417.

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

[22] A. Guénoche, P. Hansen and B. Jaumard, "Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion", *Journal of Classification*, Vol. 8, pp. 5-30, 1991.

[23] S. W. Hadley, B. L. Mark and A. Vanelli, "An Efficient Eigenvector Approach for Finding Netlist Partitions", *IEEE Trans. on CAD*, Vol. 11, No. 7, pp. 885-892, July 1992.

[24] L. Hagen, *Circuit Partitioning*, Ph.D. thesis, UCLA Computer Science Department, 1994.

[25] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD*, Vol. 11, No. 9, pp. 1074-1085, Sept. 1992.

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

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

[28] M. Hanan, P. K. Wolff, and B. J. Agule, "A Study of Placement Techniques", *J. Design Automation and Fault-Tolerant Computing*, Vol. 2, pp. 28-61, 1978.

[29] J. Hershberger, "Minimizing the Sum of Diameters Efficiently", in *Proc. 3rd Canadian Conf. on Computational Geometry*, 1991, pp. 62-65.

[30] J. Huang, *personal communication*, 1992.

[31] S. C. Johnson, "Hierarchical Clustering Schemes", *Psychometrika*, Vol. 32, No. 3, pp. 241-254, 1967.

[32] R. M. Karp, "Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane", *Mathematics of Operations Research*, Vol. 2, No. 3, pp. 209-224, 1977.

[33] E. L. Lawler, J. K. Lenstra, A. Rinnooy-Kan and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Chichester: Wiley, 1985.

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

[35] S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", *Operations Research*, Vol. 21, pp. 498-516, 1973.

[36] N. Megiddo and K. J. Supowit, "On the Complexity of Some Common Geometric Location Problems", *Siam Journal on Computing*, Vol. 13, No. 1, pp. 182-196, 1984.

[37] T.-K. Ng, J. Oldfield, and V. Pitchumani, "Improvements of a Mincut Partition Algorithm," in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1987, pp. 479–473.

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

[39] F. P. Preparata and M. I. Shamos, *Computational Geometry*, New York: Springer Verlag, 1985.

[40] G. Reinelt, "Fast Heuristics for Large Geometric Traveling Salesman Problems", *ORSA Journal on Computing*, Vol. 4, No. 2, pp. 206-217, 1992.

[41] L. A. Sanchis, "Multiple-way Network Partitioning", *IEEE Trans. on Computers*, Vol. 38, No. 1, pp. 62-81, 1989.

[42] D. G. Schweikert and B. W. Kernighan, "A Proper Model for the Partitioning of Electrical Circuits", in *Proc. ACM/IEEE Design Automation Conf.*, 1972, pp. 57-62.

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

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

[45] R.-S. Tsay and E. S. Kuh, "A Unified Approach to Partitioning and Placement", *IEEE Trans. Circuits and Systems*, Vol. 38, No. 5, pp. 521-533, 1991.

[46] Y. C. Wei and C. K. Cheng, "Ratio Cut Partitioning for Hierarchical Designs", *IEEE Trans. on CAD*, Vol. 10, No. 7, pp. 911-921, July 1991.

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

[48] C. W. Yeh, C. K. Cheng and T. T. Lin, "A Probabilistic Multicommodity-Flow Solution to Circuit Clustering Problems", in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1992, pp. 428-431.