Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

GEOMETRIC EMBEDDINGS FOR FASTER (AND BETTER) MULTI-WAY NETLIST PARTITIONING

C. J. Alpert
A. B. Kahng

December 1992
CSD-920052

Geometric Embeddings for Faster (and Better) Multi-Way Netlist Partitioning*

C. J. Alpert and A. B. Kahng

UCLA CS Dept., Los Angeles, CA 90024-1596

Abstract

We give new, effective algorithms for k-way circuit partitioning in the two regimes of $k \ll n$ and $k = \Theta(n)$, where n is the number of modules in the circuit. The central idea is that if we execute partitioning algorithms on a partition-specific *geometric embedding* of the netlist rather than on traditional graph representations of the netlist, we achieve large speedups. Surprisingly, we find that for appropriately designed geometric embeddings, this speedup can actually be accompanied by an increase in solution quality. We derive d-dimensional geometric embeddings of the netlist via: (i) a new "partitioning-specific" net model for constructing the Laplacian of the netlist, and (ii) computation of d eigenvectors of the netlist Laplacian using fast Lanczos code. We then apply (iii) fast top-down and bottom-up geometric clustering methods, e.g., given the geometric embedding, we achieve a k-way partition of the netlist in $O(n \log k)$ time. Each of these ingredients contributes to the success of our approach. For small k, we achieve very high-quality solutions as compared to the recent results of [55] and [9], quality being measured according to the multi-way partitioning objectives proposed in each of these previous works. For large k, we achieve clusterings that are significantly more useful than those of [6] [24] in a two-phase Fiduccia-Mattheyses (F-M) "clustered bipartitioning" approach. These results suggest that our particular combination of geometric embeddings and clustering algorithms preserves and highlights "natural" partitionings.

1 Preliminaries

In top-down layout synthesis of complex VLSI systems, partitioning algorithms are used to reveal the *natural* structure of the circuit, via a decomposition into k subcircuits which minimizes the connectivity between subcircuits. A generic problem statement is as follows:

k-Way Partitioning: Given a circuit netlist $G = (V, E)$, where the circuit has $|V| = n$ modules, construct a *k-way partitioning* P_k which divides the n modules into k disjoint subcircuits (also called *partitions*) $C_1, C_2, \ldots, C_k$, such that a given objective function $F(P_k)$ is minimized.

Much of the existing work on partitioning focuses on the problem of 2-way partitioning ($P_2 = \{C_1, C_2\}$). The most common formulation of this problem minimizes the *minimum-width bisection* [39] objective: $F(P_2) = e(C_1, C_2)$ where $e(C_1, C_2)$ gives the number of edges in $\{(u, v) \in E \mid u \in C_1 \text{ and } v \in C_2\}$ and

*Partial support for this work was provided by a Department of Defense Graduate Fellowship, by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050, and by Cadence Design Systems and Zycad Corporation under the State of California MICRO program.

the two partitionings must be perfectly balanced, i.e. $|C_1| = |C_2|$. To achieve more flexiblility in the 2-way partitioning, the *minimum ratio cut objective*[52] [22] [9] has been well-studied. In this case $F(P_2) = \frac{e(C_1,C_2)}{|C_1|\cdot|C_2|}$

Donath [14] and Lengauer [39] have shown that these and other standard partitioning formulations are NP-complete. Hence, previous work has used heuristics such as iterative methods (simulated annealing [49], r-opt interchange [37]), or analytic methods based on a quadratic objective (relaxation methods [10] [50], spectral computations [25] [23]). In our work, we are interested in algorithms that can achieve k-way partitionings, with the value of k being either small or large relative to the number of modules n. The motivating applications for k-way partitioning vary dramatically with the relative size of k. Indeed, we may divide the k-way partitioning problem into two distinct subclasses which we call the ***small-k partitioning*** (SKP) and *large-k partitioning* (LKP) problems.

- **Small-k Partitioning (SKP):** In the SKP regime, $k \ll n$ and is for all practical purposes bounded by a small constant, say, $k \leq 10$. The SKP problem arises in high-level layout synthesis and floorplanning, where standard approaches compute a heuristic k-way partitioning for some *prescribed* value of k (typically, $k = 2$). As observed by Wei and Cheng [53], prescribing an inappropriate value of k may prevent identification of a "natural" division which could have been easily discovered by allowing k to vary, or by applying other (e.g., bottom-up) algorithmic approaches. Furthermore, methods which achieve a k-way partition by applying a bipartitioning algorithm $k - 1$ times in succession may also fail to return a good result: it is simple to construct inputs for which optimal 2-way partitions cannot be refined into even remotely "natural" 3-way partitions. Note that the "natural" value of k is rarely known a priori, since this in effect requires knowledge of the natural circuit structure. We say that SKP can require *true* multi-way partitioning, where the value k is not necessarily fixed, and must be chosen automatically by the algorithm in minimizing the prescribed multi-way partitioning objective.

- **Large-k Partitioning (LKP):** In the LKP regime, $k = \Theta(n)$ and is typically some relatively large fraction of n, e.g., $k = n/5$. The LKP problem arises in enhancing the performance of standard top-down partitioning algorithms. As problem sizes grow very large, traditional iterative partitioning methods show clear instability and scaling effects, where the solution quality (particularly for Kernighan-Lin variants and simulated annealing) becomes less predictable and indeed decreases with problem size [39]. Recent work has shown that this reflects the so-called "error catastrophe" of local search methods, namely, that local optima follow a central limit phenomenon such that almost all tend to be of only average quality [3] [36]. To compensate for instability, practical implementations must use multiple trials with random starting points, e.g., the recent work of Wei and Cheng [52] achieves stability by returning the best of 20 random runs, and Johnson et al. [33] suggest that 500 is an appropriate number of trials for Kernighan-Lin r-opt bisection of graphs with vertex cardinality $n \approx 10^5$. Thus, maintaining high-quality iterative partitioning solutions requires either that we use a very large number of trials (resulting in impractical CPU expense), or that we somehow reduce the problem size. To this end, k-way partitioning for large $k = \Theta(n)$ has been shown to actually improve the results of iterative partitioning methods via a two-phase application of Fiduccia-Mattheyses optimization [6] [7] [39] [24] [45]. In some sense, while the clustering restricts the space of partitioning solutions, the restricted solution space is small enough to be searched effectively by traditional methods; more importantly, if the clustering is good, only "bad" solutions are eliminated from the search space (cf. the discussion in Lengauer [39]). Thus, we wish to achieve LKP solutions where k is just small enough for standard iterative approaches to once again become useful. The quality of the k-way partitioning is the amount of improvement achieved by the two-phase F-M optimization versus the standard F-M optimization on the flat netlist.

1.1 Previous Work for the SKP Problem

Early approaches to multi-way partitioning involved seeded epitaxial growth, or the "direct" method [14], in which the most well-connected unclustered module is iteratively inserted into the current clustering [35] [44] [47]. This greedy method is highly local and depends on ad hoc choices of the number of clusters, the cluster seeds, the rules for assigning modules to clusters, etc., with quality of the result being quite unpredictable.[1] In 1989, Sanchis [46] extended the Fiduccia-Mattheyses iterative interchange bipartitioning algorithm [16] [37] to k-way partitioning. Subsequently, Yeh et al. [54] proposed a primal-dual iteration motivated by a generalization of the minimum ratio cut metric [52]. These two methods use simple objective functions based only on the number of nets crossing partition boundaries; they moreover require k and the partition sizes to be specified in advance.

Yeh et al. [55] propose a "shortest-path clustering" method, where "shortest paths" between random pairs of modules are iteratively deleted from the netlist graph until it becomes disconnected into multiple components (corresponding to the partitions). This algorithm elegantly captures, in a probabilistic sense, the relationship between multicommodity flow and minimum ratio cut [10]. Moreover, the shortest-path clustering has been shown to yield true k-way partitionings $P_k = \{C_1, C_2, \ldots, C_k\}$ that are of high quality when measured by the so-called *cluster ratio*, which Yeh et al. argue to be the "proper" k-way generalization of the ratio cut bipartitioning objective [52] [22].

Minimum Cluster Ratio SKP Problem: Find a k-way partitioning $P_k = \{C_1, C_2, \ldots, C_k\}$, for any

[1] According to Lengauer [39], seeded epitaxial growth has been discarded in favor of more recent approaches such as recursive min-cut partitioning; it is unlikely that the method will perform any better in the LKP regime.

$2 \leq k \leq |V|$, that minimizes

$$F(P_k) = \frac{|c(C_1, C_2, \ldots C_k)|}{\sum_{j=i+1}^{k} \sum_{i=1}^{k-1} |C_i| \times |C_j|}$$

where $c(C_1, C_2, \ldots C_k)$ is the number of nets which cross between two or more of the partitions $C_1, C_2, \ldots C_k$ [55].

In [9], Chan et al. generalize a theoretical result [23] concerning 2-way ratio cut partitioning to k-way ratio cut partitioning. Based on their work, the first k eigenvectors of the netlist Laplacian (see Section 2 below) are use to construct an orthogonal "projector" which maps an n-dimensional space into a k-dimensional space. (There are strong similarities between this approach and what we propose below, in that netlist eigenvectors are used as the basis for the constructive partitioning.) Ideally, the n elementary unit vectors in the n-space (corresponding to the n netlist modules) would be mapped to exactly k distinct points in the k-space (corresponding to the k partitions) by this projector. However, this is not the case in practice, and therefore the authors of [9] use heuristic clustering methods to group the n images into a k-way partitioning. The authors of [9] propose a different generalization of the ratio cut objective, and define a novel, dimensionless *scaled cost* of the k-way partitioning to be:

Minimum Scaled Cost SKP Problem: Find a k-way partitioning $P_k = \{C_1, C_2, \ldots, C_k\}$, for any $2 \leq k \leq |V|$, that minimizes

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

where E_i is the number of signal nets crossing the boundary of its C_i partition [9].

Each of these last two works establishes a promising new metric to generalize the ratio cut concept. The ratio cut generalization implicitly accounts for both nets cut and size balance among the partitions; in contrast, previous algorithms could not easily achieve size balance without resorting to user-prescribed partition size limits. Moreover, both [55] and [9] are "true" k-way partitioning methods which do not require a prescribed value of k. Note that Yeh et al. and Chan et al. show that their methods give very high quality solutions according to their respective metrics; thus, we will compare our new methods against the results in [55] and [9], using their respective metrics.

1.2 Previous Work for the LKP Problem

The k-way partitioning problem for large $k = \Theta(n)$ has been well-studied in the context of "condensing" netlists to enhance the solution quality and stability of standard partitioning algorithms. In some sense, the SKP formulations of the previous section are not relevant here since all solutions will cut a very large percentage of the nets. Rather, in LKP we simply wish to obtain many small clusters which are each as densely connected as possible. While no definition which captures this LKP formulation has been proposed, solutions are typically evaluated by their utility within a two-phase "compacting", or "condensing", enhancement to the standard Fiduccia-Mattheyses bipartitioning algorithm.

The first two-phase approach, called "matching-based compaction" (MBC), was proposed by Bui et al. [6] [7]. The MBC algorithm uses the pairings of modules given by a maximal random matching in the netlist graph to induce a partitioning instance on the $n/2$ vertices, corresponding to the matching edges. The approach can be iterated so that matching is recursively performed on the compacted netlist until the problem size becomes manageable [7]. After the compaction is finished, a heuristic Fiduccia-Mattheyses bipartitioning of this compacted netlist can be found efficiently; then, the netlist is re-expanded into a "flat" initial configuration for a second Fiduccia-Mattheyses phase. The heuristic justification for this approach [6] [7] is that the Kernighan-Lin r-opt method yields significantly better results when the graph topology is sufficiently dense, i.e., has large average degree.[2]

Recently, Hagen and Kahng [24] have proposed a probabilistic method for finding regions of high connectivity (i.e., "natural clusters") by taking a *random walk* in the circuit netlist. During the random walk, strongly connected regions of the netlist will be detected as multiple revisitations of modules within these regions. This RW-ST method achieves speedups over more brute-force approaches through *implicit* examination of the netlist graph, much in the same spirit as [55] and [18]. In [24], it is reported that the RW-ST clusters lead to significant improvements in the performance of two-phase F-M, versus the matching-based compaction strategy proposed in [6] [7].

1.3 Outline of the Present Work

Our work is motivated by the fact that existing k-way partitioning methods can exhibit inferior time complexity, inferior solution quality, or both. As examples:

[2]Bui et al. claim that compacting until average degree in the netlist is ≥ 3 suffices for K-L to become essentially optimal. Lengauer [39] and the authors of [6] conjecture that this is because there are fewer local minima in the k-interchange neighborhood structure when the netlist graph has higher average degree. Note that matching-based compaction may be viewed as finding cliques of size 2, i.e., the matching edges. This may be generalized to finding cliques of size c for $c > 2$, which in turn generalizes to finding netlist subgraphs that have size c and a prescribed *density*. For example, if more than $\epsilon \cdot C(c, 2)$ edges are present among c modules of the netlist, then these c modules induce a sufficiently dense subcircuit and can be "compacted" into a cluster. Garbers et al. [18] proposed the concept of (k, l)-connectivity as a means of evaluating the *density* of clusters, without facing the combinatorial complexity of actually checking the density of $C(n, c)$ possible clusters. In the work of [18], two modules u and v are said to be (k, l)-connected if there are k edge-disjoint paths of length l between them; [18] showed that for certain structured classes of random inputs, the transitive closure of the (k, l)-connectedness relation gives a clustering equivalent to that induced by the edge density criterion. However, as noted in [24], the (k, l)-connectivity method suffers from several main weaknesses: it can yield highly nonintuitive results; the values of k and l that lead to the "correct" clustering are difficult to determine; and even assessing (k, l)-connectivity is NP-complete for $l \geq 5$ (with NP-completeness of the case $l = 4$ an open problem).

1. The SKP method of Yeh et al. [55], while of high quality, requires (i) shortest-path computations in the netlist graph, (ii) exhaustive enumeration of all partitions of a possibly large number of disconnected components obtained through the iterative shortest-path deletion, as well as (iii) an $O(mn\ log\ n)$ time complexity that has further dependence on two "accuracy" parameters b and $\frac{1}{\Delta}$.
2. The SKP method of Chan et al. [9] yields significantly worse solution quality than our methods below, while requiring additional matrix manipulations and a more complicated clustering methodology.
3. The LKP method of Hagen and Kahng [24] based on random walks yields clusterings which achieve the best-known results in terms of two-phase enhancement to F-M bisection, but $\Theta(n^3)$ time is required to process a random walk of the recommended $\Theta(n^2)$ length.

In the remainder of this paper, we describe a new methodology which achieves significant speedups of multi-way netlist partitioning by first embedding the netlist into a d-dimensional geometry, $\Re^d$ (d may be chosen by the user and in practice is bounded by a small constant), and then applying fast *geometric clustering* algorithms to the embedded netlist in order to achieve a good heuristic k-way partition. Our methodology consists of two major phases. Phase I, described in Section 2 below, constructs a geometric embedding of the netlist, i.e., we map each module of the netlist to a point in d-dimensional Euclidean space. Our embedding exploits the well-known correspondence between placement/partitioning and the eigenvectors of the Laplacian of the weighted netlist graph. We note that this idea is not new (e.g., Hall [26] proposed use of the Laplacian as early as 1970); however, our successful eigenvector-based embedding is strongly influenced by an additional ingredient, namely, a new "partitioning-specific" net model used in constructing the weighted netlist graph. A second, implicit ingredient in this first embedding phase is our use of fast Lanczos codes to solve the sparse symmetric eigenvector problem; these codes allow us to match the efficiency of the Fiduccia-Mattheyses partitioning approach.

Phase II of our methodology, described in Section 3 below, computes fast top-down and bottom-up *clusterings* of the points in the embedding, and then maps the clustering back to the original netlist, returning it as a k-way netlist partitioning. While this might seem "old hat" (e.g., Hall [26] in 1970 proposed a similar approach for classification problems), we show that the proper choices of both the clustering objective and the clustering algorithm are critical for success. We propose two fast methods, called KCENTER and AGGLOM, which yield consistently good results and which achieve k-way netlist partitions in $O(nk)$ and $O(n^2)$ time respectively (the former may also be improved to $O(n \log k)$ time). Section 4 of the paper presents experimental data showing that our multi-way partitioning approach is quite robust. For SKP, our results are an average of 15.6 % better than the previous work of Chan et al. [9], and are also competitive with the method of [55] while using much less CPU time (note that we compare our algorithms against each of these works according to their specific objective functions; we are also successful versus the spectral and interchange-based successive bipartitioning results that were reported in these papers). For LKP in the well-studied context of two-phase F-M bisection, our partitioning outputs lead to significant improvements over the two-phase results of [6] [24] [45], and an overall 27% improvement over standard F-M bipartitioning. Our discussion concludes in Section 5 with directions for future research.

2 Phase I: Fast, High-Quality Geometric Embeddings

A basic precept of optimization is that a problem instance might be solved more efficiently if it satisfies certain constraints. A number of graph optimizations have been found to be amenable to speedups when the input is embedded in a geometric space: a classic result of Vaidya [51] is that optimal matching speeds up by a factor of $\frac{n^{1/2}}{log\ n}$ when a graph input is embedded in the plane, and even greater speedups are achieved when minimum spanning tree and diameter computations are similarly embedded [43]. Therefore, we seek to achieve a geometric embedding of the weighted netlist graph, so that k-way partitioning can be more efficiently performed. For typical sizes of real netlists, the resulting speedups are very significant. We begin by noting that the embedding should be "distance-preserving", i.e. distance between two points in the embedding reflects the relationship between the corresponding vertices of the netlist graph. In our approach, we embed the netlist into d-dimensional Euclidean space via the well-established relationship between eigenvectors of the netlist *Laplacian* and minimum-cut partitionings (or minimum-wirelength placements). This correspondence is developed as follows.

We represent the circuit netlist by a simple undirected graph $G = (V, E)$ with $|V| = n$ vertices $v_1, \ldots, v_n$ representing the n modules, and edges in E capturing the hyperedges of the netlist hypergraph via a clique net model [39]. The $n \times n$ *adjacency matrix* $A = A(G)$ has A_{ij} is equal to the weight of $\{v_i, v_j\} \in E$, and by convention $A_{ii} = 0$ for all $i = 1, \ldots, n$. If we let $d(v_i)$ denote 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 $D_{ii} = d(v_i)$. The *Laplacian* of the netlist graph is given by $Q = D - A$ [23] [41]. The eigenvalues and eigenvectors of the netlist Laplacian have been extensively studied in the context of VLSI placement and partitioning (see, e.g., [2] [14] [17] [50] [5] [25] [23]; also see [23] for a good review of the related literature). In general, by viewing partitioning as the assignment of modules to bounded-size clusters and then adopting a quadratic objective function, a Lagrangian formulation can be obtained which leads to an eigenvector computation.

Hall [26] showed that the eigenvectors of the Laplacian solve the problem of finding the vector $x =$

$(x_1, x_2, \ldots, x_n)$ which minimizes

$$z = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2 A_{ij}$$

subject to the constraint $|x| = (x^T x)^{1/2} = 1$. (In other words, x is a solution to the quadratic minimum wirelength placement problem, since z is exactly the sum of squared wirelengths.) Hall's contribution was in showing that $z = x^T Qx$, so that the Lagrangian formulation $L = x^T Qx - \lambda(x^T x - 1)$ could be used to minimize z. Setting the first partial of L to zero yields $2Qx - 2\lambda x = 0$, implying $(Q - \lambda I)x = 0$ where I is the identity matrix. This is readily recognizable as an eigenvalue formulation for λ, and the eigenvectors of Q are the only nontrivial solutions for x. The key observation is that the magnitude of each eigenvalue is equal to the "total squared wirelength" corresponding to the placement given by the associated eigenvector (recall that $|x| = 1$). As noted by Hall, the minimum eigenvalue 0 gives the "trivial" solution $x = (1/\sqrt{n}, 1/\sqrt{n}, \ldots, 1/\sqrt{n})$, which just correspond to all modules placed at the same location. Therefore, the eigenvector corresponding to the second smallest eigenvalue λ_2 is used as a heuristic minimum squared wirelength placement. Hagen and Kahng [22] [23] established a close relationship between the second smallest eigenvalue λ_2 and the optimal ratio cut partitioning solution. The work of [22] [23], as well as the closely related work of Hadley et al. [25], showed that an eigenvector computation led to fast, stable, and high-quality heuristic partitioning solutions.[3]

At the heart of our methodology is the observation that every eigenvector of Q gives a ***distinct***, one-dimensional spatial embedding of the circuit graph wherein strongly connected modules will tend to be placed close to each other. Because the squared wirelength of each eigenvector placement is given by its corresponding eigenvalue, the lowest eigenvalues will correspond to eigenvectors that are the most "distance-preserving". Thus, to achieve a d-dimensional embedding of the netlist, we use the d eigenvectors corresponding to the lowest d nonzero eigenvalues of the Laplacian. The i^{th} components of these d eigenvectors give the d coordinates of the i^{th} module in $\Re^d$ (see the small example for $d = 2$ in Figure 1).[4]

We now make a brief digression to establish the ***computational practicality*** of this geometric embedding.

[3]The result in [22] suggested that the eigenvector x corresponding to λ_2, i.e., the solution of the matrix equation $Qx = \lambda_2 x$, be used to guide the partitioning. For ratio cut partitioning, [22] used x to induce a linear ordering of the modules, and the best "split" in terms of ratio cut cost was returned. In other words, the n modules x_i of the eigenvector were sorted to yield an ordering $v = v_1, \ldots, v_n$ of the modules, called the *spectral ordering*. The splitting index r, $1 \leq r \leq n-1$ was then found which yields the best ratio cut cost when modules with index $> r$ were placed in U and modules with index $\leq r$ were placed in W. A similar heuristic was proposed by Hadley et al. [25].

[4]Note that this naive embedding approach is not new by any means: for example, Hall [26] proposed using the eigenvectors corresponding to the second and third eigenvalues in order to achieve a two-dimensional embedding of a standard classification problem. However, as we discuss in Section 2.2 below, additional refinements are necessary before the naive eigenvector embedding approach becomes a useful basis for netlist partitioning.

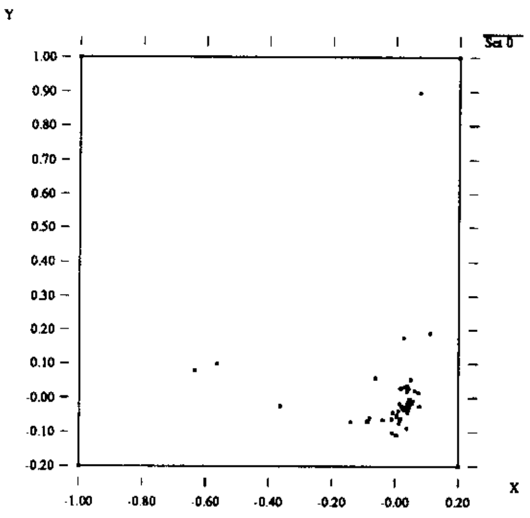

Figure 1: Geometric embedding of the IC67 netlist (from ILLIAC IV circuit board) in two dimensions, using eigenvectors corresponding to the two smallest nonzero eigenvalues of the netlist Laplacian.

2.1 Practicality of the Eigenvector Embedding

As noted in [22] [23], computing the first few eigenvectors of the netlist Laplacian $Q = D - A$ is simplified due to the symmetry and sparsity of this matrix, which naturally suggests application of the Lanczos algorithm for the sparse symmetric eigenproblem. The Lanczos iteration is heavily dependent on the sparsity of the input [19]. Note that for the smallest and largest benchmarks considered in our studies (i.e., MCNC Primary1 and Primary2), the Q matrices are of size 883×883 and 3014×3014, but respectively have only 5,541 and 30,090 nonzeros when the standard clique net model [39] is used. The least sparse benchmark is Test05, which has 219,811 nonzeros according to the clique net model.

Using the same Lanczos implementation that was used by [22], we find that computing the second eigenvectors for Primary1, Primary2 and Test05 respectively required 19, 83 and 619 CPU seconds on a Sun 4/60 (Sparc-1) machine. These runtimes confirmed the dependence on sparsity, and indeed, sparsification techniques such as removal of large nets (e.g., with number of pins > 100) have been found to yield tremendous speedups (e.g., Hagen [21] reports that the second eigenvector computation for such a sparsified industry netlist with 40,258 modules and 36,208 nets required 714 CPU seconds on a Sun Sparc-2 (a machine that is about twice as fast as the Sparc-1)). We emphasize that these runtimes are very competitive with those of iterative methods. As noted in [22], the 10 F-M computations used by the RCut1.0 ratio cut program of Wei and Cheng [52] required 204 CPU seconds on a Sun 4/60 for the Primary2 benchmark, and this is significantly higher than the 83 seconds noted above for the eigenvector computation. Finally, we note that the cost of computing *multiple* eigenvectors (i.e., a higher-dimensional geometric embedding) is by no means prohibitive. According to our studies, achieving a d-dimensional embedding for Primary1 requires 1.15, 1.83, 2.00, 2.58, 2.64, 3.25, 3.29, 4.31 and 4.44 times the CPU cost of achieving a 1-dimensional embedding (i.e., just the second eigenvector), for $d = 2, 3, \ldots, 10$ respectively.

2.2 Refinement of the Eigenvector-Based Geometric Embedding

The d-dimensional geometric embedding that we proposed above, based on the d eigenvectors of the lowest d nonzero eigenvalues of the Laplacian, is somewhat naive. As noted in Footnote 5, this was exactly the idea of Hall [26] in 1970. Based on our experiments, we have found that at least one "missing ingredient" must be supplied in order to obtain a robust eigenvector-based geometric embedding that is useful for both the SKP and LKP problems. (A possible second ingredient is discussed under future research in Section 5 below.)

A New "Partitioning-Specific" Net Model. Our enhancement to the traditional eigenvector embedding concerns the clique net model that is used to construct the matrix A in the Laplacian $Q = D - A$. According to the clique net model, a p-pin signal net will induce a clique of $C(p, 2)$ edges among its p modules; the cliques for all signal nets are superposed to yield the matrix A. The early survey by Hanan et al. [27] details several clique weighting variants which propose uniform weighting of the $C(p, 2)$ edges by such values as $\frac{2}{p}$, $\frac{1}{p}$, $\frac{1}{p-1}$, etc. Simple dimensional analysis shows that all of these net models are very similar in practice. Thus, present work has adopted a "standard" weighted clique model [39], wherein a p-pin net contributes $\frac{1}{p-1}$ to each of $C(p, 2)$ A_{ij} values.

We observe that this standard clique model seems to be motivated by the following consideration: in bipartitioning, when one p-pin signal net crosses the partition boundary, at least $(p-1)$ edges in its clique must be cut, hence the weight of cut edges will be $\geq (p-1) \cdot \frac{1}{(p-1)} = 1$. In other words, the standard net model ensures that a cut net will make a *minimum contribution* of 1 to the partitioning cost function. Unfortunately, a cut net can also contribute up to $\frac{p^2}{4} \cdot \frac{1}{(p-1)}$ to the partitioning objective, meaning that large nets in particular will receive disproportionate attention in the partitioning process. To remedy this, we propose a new *partitioning-specific* clique net model, whereby any cut net will make an *expected contribution* of 1 to the cost function. In this way, all nets will receive uniform weight in the optimization, hopefully yielding improved heuristic partitions. We achieve this uniform weighting via the following analysis. By enumerating all possible assignments of the p pins across a 2-way partitioning, we see that the probability of an $(i, p-i)$ partitioning is $\frac{C(p,i)}{2^p}$. There are also $i \cdot (p-i)$ clique edges that will cross an $(i, p-i)$ partitioning of the net. To achieve an expected net cut cost of 1, the following equation should hold for the uniform edge weight W in the clique: $\sum_{i=0}^{p} \frac{W \cdot i \cdot (p-i) \cdot C(p,i)}{2^p} = 1$. It is then simple to show that the relation $W = \frac{1}{\sum_{i=0}^{p}(i \cdot (p-i) \cdot C(p,i)/2^p)}$ implies $W = \frac{4}{p(p-1)}$, and it is this revised clique net model that we propose to use in constructing the Laplacian.[5]

Given this enhancement, our geometric embedding of the netlist is computed as shown in Figure 2.

```
d-Dimensional Geometric Embedding
Input: $H = (V, E')$ = netlist hypergraph with $|V| = n$; dimension $d$
Output: $d$-dimensional point set $v_1, \ldots, v_n \in \Re^d$
1. Transform $H$ into graph $G = (V, E)$, using clique model with weight $\frac{4}{k(k-1)}$
2. Compute $A$ = adjacency matrix and $D$ = degree matrix of $G$
3. Compute $d$ smallest non-zero eigenvalues $\lambda_i(Q)$ of $Q = D - A$, $i = 2, \ldots, d+1$
4. Compute real eigenvectors associated with each of the $\lambda_i(Q)$
5. Generate $d$ coordinates of $v_i$ as $i^{th}$ components of the $d$ eigenvectors
```

Figure 2: Eigenvector-based, d-dimensional geometric embedding of circuit netlist.

3 Phase II: Geometric Clustering

The geometric embedding in Phase I maps the n netlist modules to n points in d-dimensional Euclidean space, which we denote by $V = \{v_1, v_2, \ldots, v_n\} \subset \Re^d$. The goal of Phase II is to divide V into k disjoint *clusters*, which will correspond to the output partitions. We define a k-way clustering of V as $P_k = \{C_1, C_2, \ldots, C_k\}$ where $C_1 \cup C_2 \cup \ldots \cup C_k = V$ and $C_i \cap C_j = \emptyset \;\; \forall\; 1 \leq i < j \leq k$. A large body of work has established various criteria that capture whether a given clustering of V is ***natural.*** Intuitively, we would like our clusters to be compact and well-separated. A cluster with small ***diameter***, defined as $diam(C) = \max_{v_i, v_j \in C} d(v_i, v_j)$ (where $d(v_i, v_j)$ denotes the standard Euclidean distance between $v_i, v_j \in V$), satisfies the former goal; on the other hand, two clusters with large ***split***, defined as $split(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} d(v_i, v_j)$ will achieve the latter goal. Diameter and split are frequently used in analyzing the quality of a given clustering. Before describing our chosen clustering algorithms, we now give a taxonomy of major clustering objectives and approaches.[6]

[5]Actually, note that we must also ignore the $(0,p)$ and $(p,0)$ partitioning cases, since these do not actually cut the signal net. Thus, the actual uniform edge weight that we use is given by $W = \frac{4}{p(p-1)} \cdot \frac{2^p - 2}{2^p}$.

[6]Two points should be made. First, we use the term "cluster" in keeping with standard notation in the classification, phylogenetics, and numerical taxonomy literatures. For all purposes, the terms "partition" and "cluster" are henceforth interchangeable. Second, our discussion slightly abuses the terms "point" and "distance", with the obvious extensions to "vertex" and "shortest path length" being implicit for weighted graph inputs.

3.1 Clustering Objectives

For a given k, geometric clustering divides V into a clustering P_k which minimizes or maximizes $f(P_k)$ for some objective function f. The main objective functions in the classification literature are formulated as follows:

1. *Max-Split Clustering:* Maximize $f(P_k)$ where
$$f(P_k) = \min_{C_i, C_j \in P_k} \{split(C_i, C_j)\}.$$
2. *Min-Diameter Clustering:* Minimize $f(P_k)$ where
$$f(P_k) = \max_{C \in P_k} \{diam(C)\}.$$
3. *Min-Sum-Diameter Clustering:* Minimize $f(P_k)$ where
$$f(P_k) = \sum_{1=i}^{k} diam(C_i).$$
4. *Min-Radius* or *k-Center Clustering:* Find $W \subset V$, with $|W| = k$, that minimizes
$$f(W) = \max_{v \in V} \{ \min_{w \in W} \{dist(v, w)\}\}.$$
(Here, the choice of W implicitly creates the clustering P_k, with each $C_i \in P_k$ consisting of an element $w_i \in W$ along with all points of V that are closer to w_i than to any other point of W. In other words, w_i is the center of cluster C_i, and the cluster centers and cluster memberships are chosen to minimize the largest *radius* of any cluster.)

Each of these is an intuitively reasonable objective for our purposes. Previous works [26] [9] simply indicate that they use "geometric clustering", but do not specify an objective function f. As we will see in Section 4, the choice of f is critical. (Indeed, the differences among Formulations 1-4 become more noticeable given a highly nonuniform distribution of point locations in the eigenvector-based embedding; recall the example of Figure 1.)

Formulation 1 can be solved optimally in $O(n \log n)$ time by the so-called Single Linkage algorithm [34] (see Figure 3). The algorithm begins with an initial n-way clustering P_n in which each point is in its own cluster. Iteratively, the two clusters with least split in P_i are merged to yield the P_{i-1} clustering. Each P_i generated is an optimal i-way max-split solution (see [12] for proof). Efficiency of the Single Linkage approach follows from the observation that the minimum edge from any cluster to its nearest neighboring cluster (i.e., the "split" edge) must be an edge of a minimum spanning tree (MST) [12]. Hence, one can build the same clusterings $P_2, P_3, \ldots P_n$ more efficiently by bulding the MST on the n points of V, and then iteratively removing the largest edge of the MST, in order to obtain an optimal i-way max-split clustering for $i-1$ removed edges. This can be accomplished in $O(n \log n)$ time in the plane using the Voronoi diagram [43].

```
Optimal Max-Split Clustering
Input:   Set of points $|V| = n$ in $d$-space and parameter $k$
Output: Clusterings $P_2, P_3, \ldots, P_k$
1. Compute $T$, the minimum spanning tree of $V$
2. for $i = 2$ to $k$ do
3.     $C_j = \emptyset, 1 \leq j \leq i$
4.     Remove the largest edge from $T$, yielding $i$
       connected components $T_1, T_2, \ldots, T_i$ of $T$
5.     for $j = 1$ to $i$ do
6.         for all $v \in T_j$ add $v$ to $C_j$.
7.     $P_i = \{C_1, C_2, \ldots, C_i\}$
   endfor
```

Figure 3: Single Linkage algorithm for optimal max-split clustering P_k (Formulation 1).

With respect to Formulations 2, 3, and 4, the following sets of results have been established.

Fact 1: Formulations 2, 3, and 4 are all NP-complete in general weighted graphs [8].

Fact 2: Formulations 2, 3, and 4 are all NP-Complete for $k \geq 3$ when V is embedded in $\Re^d$ with $d > 1$ [12] [13] [40].

Fact 3: Finding solutions to Formulation 2 or to Formulation 4 that are within a factor $\alpha < 2$ of optimal is NP-complete in $\Re^d$ for $d > 2$.[7]

Fact 4: In general graphs whose edge weights do not satisfy the triangle inequality, neither Formulation 2 nor Formulation 3 may be approximated within any fixed constant factor of optimal for $k \geq 3$ [20]. (Note that Formulation 4 holds little meaning in general graphs, though relevant algorithms may still be applied).

For geometric embeddings and graphs which satisfy the triangle inequality, Formulations 2 and 4 may often be treated together in the sense that a guaranteed max cluster radius $\leq D$ (Formulation 4) will imply a guaranteed max cluster diameter $\leq 2D$ (Formulation 2) by the triangle inequality. There are several algorithms which achieve a performance ratio of 2 for these two formulations. In our work, we use a very simple technique due to Gonzalez [20] which runs in $O(nk)$ time and 2-approximates both formulations. This algorithm, which we call KCENTER, or the "Farthest-Point Algorithm", is stated in Figure 4.

[7]For $d = 2$, Feder and Greene [15] showed the α-approximation problem is NP-complete for $\alpha < \approx 1.97$ for the Euclidean metric and $\alpha < \approx 1.82$ for the Manhattan metric.

```
KCENTER (Farthest Point) Algorithm [20]
Input:    Set of points $|V| = n$ in $d$-space and parameter $k$
Output: Clusterings $P_2 \ldots P_k$
    $W \equiv$ set of cluster centers
    $neighbor(v) \equiv$ nearest point to $v$ that is in $W$
    $dist(v) \equiv$ distance from $v$ to $neighbor(v)$
1. $W = \emptyset$
   $dist(v) = \infty$ for all $v \in V$
2. while $|W| \leq k$ do
3.     $D = max\{dist(v) | v \in V - W$
4.     choose $w$ from $V - W$ s.t. $dist(w) = D$
5.     $W = W \cup w$
6.     update $neighbor(v)$ and $dist(v)$ for all $v \in V - W$
7.     for each $w_i \in W, 1 \leq i \leq |W|$ let $C_i = \{v | neighbor(v) = w_i\}$
8.     $P_{|W|} = \{C_1 \ldots C_{|W|}\}$
   endwhile
```

Figure 4: Simple algorithm for computing a sequence of i-way clusterings P_i, $i = 1, 2, \ldots, k$, which have performance ratio of 2 with respect to Formulations 2 or 4. For each point not in W, KCENTER maintains $neighbor(v)$ (the nearest point to v in W) as well as $dist(v)$ (the distance from v to $neighbor(v)$). Initialization (step 1) as well as the steps in the **while** loop (steps 3-7) all require $O(n)$ time, yielding an $O(nk)$ algorithm.

At the end of the KCENTER execution, there are $k + 1$ points in W which are all at least distance D from each other; the first k of these serve as the centers of the k clusters. By the construction, every point in V is within D of some cluster center, hence each cluster radius is at most D (Formulation 4). By the triangle inequality, each cluster diameter is at most $2D$ (Formulation 2). Furthermore, note that by the pigeonhole principle, two of the $k + 1$ points in W must belong to the same cluster. This means that for a k-way clustering, the min diameter achievable is at least D (and the min radius achievable is at least $D/2$). From this argument, the yielding the desired 2-approximation can be shown:

Theorem 1: For $V \subset \Re^d$, the KCENTER algorithm solves both Formulation 2 and Formulation 4 to within a factor of 2 of optimality. [20] □

By applying techniques due to Vaidya [51], Feder and Greene [15] used the KCENTER construction as the basis for a more complicated but faster scheme which for $V \subset \Re^d$ yields a 2-approximate clustering in optimal $O(n \log k)$ time. However, our experimental results below use the slightly slower KCENTER algorithm due to its simplicity.

3.2 General Clustering Strategies

Recall from the statement of Fact 2 that Formulations 2, 3, and 4 are known to be NP-complete only for $k \geq 3$. For $k = 2$, there are well-known efficient algorithms for each of these formulations,[8] suggesting iterative use of an optimal 2-way partitioning (e.g., to divide the largest current cluster) in order to obtain a heuristic k-way partitioning. This approach leads to the concept of a ***hierarchical clustering*** where clusterings $P_1, P_2, \ldots P_n$ are generated with successive P_i, P_{i+1} differing only in that a single $C \in P_i$ has been separated into $C_1, C_2 \in P_{i+1}$.

The ***top-down hierarchical*** approach reflects current practice for k-way partitioning of VLSI netlists. Hansen and Jaumard [29] survey a variety of implementations for this "divisive" approach in the context of min-diameter clustering. It is also reasonable to construct a ***bottom-up hierarchical*** clustering (recall that this was optimal for Formulation 1). In particular, we propose to use an "agglomerative" algorithm, which we call AGGLOM (see Figure 5; also, cf. the "Complete Linkage Algorithm" of Johnson [34]).

```
AGGLOM ("Complete Linkage") Algorithm
Input:   $G(V, E)$ with $|V| = n$ and parameter $k$
Output: Clusterings $P_n \ldots P_k$
1. Set $P_n = C_1, \ldots, C_n$ with $C_i = \{v_i\}$ for each $i$
2. Set Priority Queue $Q = \emptyset$
3. For every $1 \leq u \leq n, 1 \leq v \leq n$ insert $(u, v)$
       into $Q$ with key $dist(C_u, C_v)$
   $i = n$
4. while $i > k$ do
5.     $(u, v) = delete\_min(Q)$
6.     if $C_u, C_v \in P_i$ then
7.         $C_{new} = C_u \cup C_v$
8.         $P_{i-1} = (P_i \cup C_{new}) - C_u - C_v$
9.         For every $j$ s.t $C_j \in P_{i-1}$ insert $(new, j)$
               into $Q$ with key $dist(C_{new}, C_j)$
           $i = i - 1$
       endif
   endwhile
```

Figure 5: AGGLOM algorithm, which computes a set of hierarchical k-way clusterings P_k for $k = n, n-1, \ldots, 3, 2$. AGGLOM minimizes the diameter objective (Formulation 2) of the k-way clustering at each iteration.

The AGGLOM algorithm starts with n clusters of size one and then at each iteration merges two clusters such that the overall objective function is minimized; this process is iterated until a k-way clustering is obtained. In our experiments below, we use the simple implementation shown in Figure 5, which re-

[8]See Monma and Suri [42] and Asano et al. [1] for $O(n \log n)$ min-diameter algorithms in the plane. Hershberger [30] has given a very fast algorithm for min-sum-diameter clustering which is $O(n \log^2 n / \log \log n)$ in the plane and which remains subquadratic in dimension ≥ 3. Hansen and Jaumard [28] and [42] discuss min-sum-diameter clustering in general graphs.

quires $O(n^2 \log n)$ time. It should be noted that Benzecri [4] has given an elegant and efficient $O(n^2)$ implementation of AGGLOM, based on constructions of chains of nearest neighbors.[9]

4 Results and Discussion

4.1 Results for the SKP Problem

While the four objectives given by Formulations 1 through 4 are all intuitively reasonable, our experiments show clear differences in the partition quality afforded by these objectives. The partition quality also seems heavily dependent on the overall strategy, e.g., top-down divisive versus bottom-up agglomerative.

Our first set of experiments broadly compared the standard net model against our new partitioning-specific net model, and also contrasted six clustering algorithms: AGGLOM, KCENTER, Single Linkage, Divisive Min-Diameter, Divisive Min-Sum-Diameters, and Agglomerative Min-Sum-Diameters (identical to AGGLOM, except that the merging criterion is sum-of-diameters rather than the maximum cluster diameter). We generated k-way clusterings of the Primary1 benchmark using all methods, for all values $2 \leq k \leq 9$, and over all d-dimensional geometric embeddings for $1 \leq d \leq 5$.

The data in Table 1 show the effect of the partitioning-specific net model in obtaining high-quality geometric embeddings that are amenable to processing by our clustering algorithms. In the Table, we show data for AGGLOM and KCENTER. For these algorithms, the partitioning-specific net model led to clearly improved results as measured by the Scaled Cost objective of Chan et al. [9]: for AGGLOM, an average of 12.1% improvement in partitioning quality was observed, and KCENTER results improved by an average of 17.4% with the new net model. Similar phenomena were observed for all other clustering algorithms tested, and thus our subsequent experiments all rely on the new partitioning-specific net model.

In Table 2, we show the best results obtained by each of the six algorithms for the Primary1 benchmark, again for the embeddings in dimension $1 \leq d \leq 5$, and again in terms of the Scaled Cost objective [9]. Here, the poor performance of the Single Linkage and (Agglomerative) Sum of Diameters clusterings is quite striking. We may conclude that minimization of split is not a good objective function vis-a-vis the

[9]The time complexity of our AGGLOM implementation is analyzed as follows. First, $O(n^2)$ distances are inserted into the priority queue (initialization step 3), requiring $O(n^2 \log n)$ time; each distance in the priority queue reflects the "merged diameter" of a pair of existing clusters. Then, the **while** loop in lines 4-9 is executed at most $O(n^2)$ times. In this loop, each deletion of an element from the priority queue (line 5) requires $O(\log n)$ time, and the **if** condition (line 6) can be tested in constant time by recording the sizes of each "merged" cluster when their indices are inserted into the queue. Hence, when the **if** condition fails, the **while** loop requires $O(\log n)$ time. The **if** condition succeeds exactly $n - k$ times: within lines 7-9, the merging process of lines 7-8 can be done in $O(n)$ time, and $O(n)$ more elements are added to the priority queue at line 9, resulting in $O(n \log n)$ total time for each successful **if** test. Therefore, (i) the initialization, (ii) all of the **if** condition failures, and (iii) all of the **if** condition successes will each take $O(n^2 \log n)$ time. The work of Day [11] describes such an implementation as ours. A more practical implementation of AGGLOM would use Benzecri's method [4] for the speedup that it affords.

Algorithm	Net Model	Number of Clusters (k)							
		9	8	7	6	5	4	3	2
AGGLOM	$1/(p-1)$	37.8	35.0	30.5	31.2	27.8	30.7	18.6	14.6
	$4/(p \cdot (p-1))$	35.5	32.7	29.8	24.8	26.0	17.4	17.9	13.5
KCENTER	$1/(p-1)$	51.6	41.2	42.5	39.5	43.4	47.6	14.3	14.2
	$4/(p \cdot (p-1))$	54.4	33.6	34.7	30.7	27.5	16.4	17.4	13.5

Table 1: Comparison of the two net models (best clusterings over embeddings in $\Re^d$, $1 \leq d \leq 5$, for AGGLOM and KCENTER. Results are obtained for the Primary1 benchmark netlist, and evaluated using the Scaled Cost objective of Chan et al. [9].

eigenvector-induced embeddings. Moreover, heuristics to minimize the sum of diameters seem to completely isolate outliers in the embedding, thus generating clusterings that are poorly balanced in terms of cluster sizes. In retrospect, these phenomena seem closely tied to the non-uniform nature of the point distributions in $\Re^d$ (recall Figure 1). The three algorithms which address the min-diameter objective give clearly better results, although Divisive Min-Diameter clustering is noticeably inferior to AGGLOM and KCENTER. We again hypothesize that this is due to the nature of our embeddings, which are dense with very few outliers (See Appendix).

Objective/Algorithm	Number of Clusters (k)							
	9	8	7	6	5	4	3	2
AGGLOM	35.5	32.7	29.8	24.8	26.0	17.4	17.9	13.5
KCENTER	54.4	33.6	34.7	30.7	27.5	16.4	17.4	13.5
SGL-LINK	126.0	109.7	87.9	49.5	39.2	39.9	23.7	13.5
Div Min-Diam	59.5	62.0	66.1	59.5	46.8	42.9	32.8	13.5
Div Min-Sum-Diam	135.1	102.9	100.0	96.0	74.9	59.8	72.3	13.5
Agglom-Sum-Diam	218.4	199.8	189.1	154.5	127.2	88.3	72.3	13.5

Table 2: Comparison of the various clustering objectives and algorithms (best clusterings over embeddings in $\Re^d$ with $d = 1, \ldots, 5$). Results are obtained for the Primary1 benchmark netlist using the partitioning-specific net model in constructing the Laplacian. Numbers reported give the Scaled Cost objective of Chan et al. [9].

Given these results, we now examine the performance of KCENTER and AGGLOM more closely. In Table 3, we show clustering results in each of the dimensions $d = 1, \ldots, 10$ for the Primary1 benchmark (we center on Primary1 in this portion of our discussion, since it is the only circuit treated in the work of Chan et al. [9]). (The most expensive runtimes, corresponding to $k = 9$, are also shown for KCENTER.) The best result that we obtain for each value of k is highlighted for each algorithm, and these values represent an average 11.2% win for KCENTER, and an average 15.6% win for AGGLOM, over the results of [9]. As would be expected, even larger improvements are achieved over the recursive spectral bipartitioning (RSBipart) results, which we quote from [9]. We note that for a given value of k, the most favorable dimension d seems to grow roughly with k thereby creating a *diagonal effect* in the Table.[10]

Algorithm	d	Number of Clusters (k)								
		9	CPU	8	7	6	5	4	3	2
RSBipart		**48.8**		**45.5**	**40.6**	**37.1**	**41.0**	**36.6**	**27.3**	**13.5**
Chan et al.		**45.0**		**51.1**	**32.1**	**37.8**	**25.9**	**25.7**	**15.9**	**13.5**
KCENTER	1	214.1	75s	194.1	206.8	171.3	152.2	102.7	79.8	**13.5**
	2	136.9	101s	123.8	114.7	119.6	112.2	84.4	17.5	14.2
	3	87.3	111s	87.6	79.8	54.7	**27.5**	**16.4**	**17.4**	20.4
	4	75.2	137s	70.4	34.7	**30.7**	32.4	51.2	35.4	29.3
	5	54.4	135s	**33.6**	46.5	48.0	51.4	46.1	41.5	33.6
	6	51.0	151s	51.9	**34.4**	60.2	46.9	47.2	40.4	29.3
	7	**34.6**	173s	54.1	54.1	49.2	39.5	33.6	31.4	33.4
	8	60.3	175s	59.7	54.4	48.3	48.1	65.5	60.1	59.5
	9	53.1	198s	54.1	51.9	57.7	54.9	54.3	64.6	58.3
	10	52.6	187s	55.5	53.8	69.5	66.7	65.6	67.4	59.6
AGGLOM	1	190.2		164.7	166.6	143.0	122.4	134.7	147.8	**13.5**
	2	137.2		140.4	144.7	143.7	126.8	81.8	**17.9**	14.1
	3	96.4		98.2	73.6	39.2	26.1	**17.4**	18.1	20.4
	4	63.6		65.6	**29.8**	**24.8**	**26.0**	22.6	23.8	20.4
	5	35.5		32.7	32.1	34.3	35.8	36.0	31.2	20.4
	6	46.7		34.4	30.0	39.0	30.7	33.2	23.8	20.4
	7	**33.1**		**31.7**	32.2	34.6	29.5	27.9	25.8	27.6
	8	39.8		39.8	40.1	37.1	34.0	34.6	37.4	31.1
	9	39.5		40.3	37.9	35.5	36.4	38.9	35.7	40.3
	10	41.8		39.9	38.2	38.3	36.0	38.4	35.7	40.3

Table 3: Performance of KCENTER and AGGLOM algorithms versus results reported of Chan et al. [9] for the Primary1 benchmark. AGGLOM clearly has the best results, but has $O(n^2)$ complexity. KCENTER results remain better than those of Chan et al., and can be obtained in $O(n \log k)$ time once we have obtained the geometric embedding. We show runtimes for KCENTER on a Sun Sparc IPC; these reflect the slower $O(nk)$ implementation of Figure 4. Our AGGLOM runtimes are impractical since we use an $O(n^2 \log n)$ implementation and $k \ll n$.

Although Table 3 illustrates the superiority (indeed, almost uniform dominance) of AGGLOM over KCENTER, the best known implementation of AGGLOM requires $O(n^2)$ time, while our KCENTER implementation is $O(nk)$ and can be improved to $O(n \log k)$ [15]. For the small number of clusters desired in k way partitioning, the speedup gained by KCENTER is very significant. To facilitate comparison with future work by other researchers, we give in Table 4 the best KCENTER results over the geometric embeddings for $d = 1, \ldots, 10$, again measured by the Scaled Cost metric. Again, the diagonal effect is strongly apparent for most of the benchmarks.

Finally, Table 5 shows comparisons between KCENTER and the Shortest-Path Clustering (SPC) results given by Yeh et al., as measured by the Cluster Ratio objective [55]. In the Table, we also include the successive bipartitioning ("Recursive Ratio Cut") results that were reported in [55]. To match [55], we assume uniform module areas. For each benchmark, we report the best k-way partition obtained for $2 \leq k \leq 9$, over embedding dimensions ranging from $d = 1$ to $d = 10$. This is in the same spirit as the

[10]This seems to confirm the ideas of Chan et al., who use k-dimensional embeddings to obtain k-way partitions; alternatively, this may also be an artifact of the objective function proposed in [9]. The "diagonal effect" implies that for any given value of k, only a few embeddings corresponding to $d \approx k$ need to be examined, and for k small, only a few eigenvectors need to be computed.

Test	Number of Clusters - k (Best dimension)							
Case	9	8	7	6	5	4	3	2
19ks	15.0(10)	15.8(6)	15.6(1)	15.1(1)	14.4(1)	13.1(1)	12.5(1)	17.6(1)
bm1	27.6(6)	30.6(10)	28.6(10)	19.8(10)	17.9(6)	11.1(4)	7.0(2)	5.8(1)
Prim1	34.6(7)	33.6(5)	34.4(6)	30.7(4)	27.5(3)	16.4(3)	17.5(3)	13.5(1)
Prim2	11.7(6)	12.0(6)	11.8(9)	11.5(5)	10.4(4)	9.0(2)	7.5(1)	5.9(1)
Test02	21.5(8)	21.2(7)	21.1(6)	21.2(5)	23.1(3)	23.6(3)	19.1(2)	30.1(1)
Test03	21.0(9)	22.4(9)	23.2(9)	22.4(3)	22.2(2)	19.3(2)	21.4(2)	16.7(3)
Test04	22.1(6)	23.8(9)	24.4(6)	24.3(6)	27.2(5)	27.4(3)	36.0(2)	66.1(1)
Test05	11.0(6)	10.6(7)	10.7(5)	11.1(4)	10.3(10)	8.8(10)	10.2(6)	10.6(1)
Test06	31.0(10)	32.4(10)	33.6(4)	26.4(4)	28.8(4)	25.9(2)	19.3(2)	28.6(1)

Table 4: Scaled Cost measures of best k-way partitions obtained by KCENTER using d-dimensional embeddings, $1 \leq d \leq 10$. Value in parentheses tells which dimension had the best Scaled Cost, illustrating the diagonal effect.

results of Yeh et al., who generate the entire partitioning hierarchy (i.e., all k-way partitions for $2 \leq k \leq n$) and report the best partitioning quality found. Although our results are better than those of the recursive ratio cut approach, they are noticeably inferior to the SPC results. However, it should be noted that the Shortest-Path Clustering is extremely expensive in comparison with our method. (Also, if $O(n^2)$ complexity is acceptable to the user, our AGGLOM method will likely yield better results than KCENTER, particularly with respect to balance among the partition sizes.)

Test	Recursive Ratio Cut(RR)			Shortest Path Clustering(SPC)			KCENTER		
Case	partitions	cut	R_C	partitions	cut	R_C	partitions	cut	R_C
19ks	73:2771	11	5.43	876:909:1059	127	4.72	1:1:62:185:2595	32	4.86
PrimSC1	91:742	11	16.29	76:76:681	14	12.81	152:681	14	13.53
PrimSC2	739:2275	83	4.93	731:2283	77	4.61	627:2387	88	5.88
Test02	6:43:46: 345:429:794	152	16.62	33:34:1596	9	8.32	2:70:1591	10	8.72
Test03	321:1286	48	11.62	764:843	69	10.71	2:346:1259	52	11.85
Test04	394:1121	51	11.54	73:1442	6	5.69	1:70:1444	6	5.85
Test05	287:2308	48	7.24	103:2492	8	3.11	4:13:102:2469	15	4.78
Test06	66:127:179:213: 421:404:342	91	7.23	133:133:133:135:135: 135:170:268:510	81	6.22	1:2:9:130: 354:1256	63	9.34

Table 5: Results for KCENTER, compared against those reported for Shortest-Path Clustering [10] according to the Cluster Ratio objective R_C (reported as a multiple of 10^{-5}).

4.2 Results for the LKP Problem

Recall from the discussion of Section 1.2 that LKP solutions are typically evaluated by the improvement afforded to Fiduccia-Mattheyses bipartitioning, via the two-phase compaction approach. Here, we use the $O(n^2)$ AGGLOM algorithm, since it is bottom-up and hence fairly efficient for large values of $k = \Theta(n)$.[11] Our motivation for this choice also stems from separate studies wherein we have confirmed that AGGLOM

[11] Intuitively, if the maximum cluster diameter remains small, AGGLOM will not have to delete or insert too many "merge distance" edgelengths into the priority queue. Of course, KCENTER has $O(n \log n)$ complexity even for large k and may be preferable in practice.

tends to yields very high-quality solutions for Formulation 2 (min-diameter), again especially when k is large.

In Table 6, we compare two-phase FM results using AGGLOM clustering of the geometrically embedded netlist against the results for (i) standard F-M (on flat netlists), (ii) random matching-based clusterings, and (iii) RW-ST clusterings, all of which are quoted from [24]. For each benchmark, we use the same number of clusters k as was used in [24]; these values were obtained via [21]. For each benchmark, we report the best two-phase F-M result using agglomerative clustering in each of ten embedding dimensions ($1 \leq d \leq 10$) and 20 runs for each clustering. The use of 20 random runs follows common practice (e.g., [52]) and also reflects the experimental protocol in [24]. Our clusterings give an average of 26.9% improvement over the "flat" F-M partitioning results, in contrast to the 17% improvement given by RW-ST clusterings [24] using the same k values.[12] Last, we note that the AGGLOM clusterings also seem quite good when compared to the "Cluster+" results of Roy and Sechen, who report two F-M bisection instances in [45]. We can make a direct comparison with [45] for the PrimSC1 benchmark: their method achieves net cut of 67, and they report also net cut of 198 for the "Stable Algorithm" of [53] (best of 30 trials). In contrast, the AGGLOM clustering results in a bisection net cut of 49 for this circuit.

5 Conclusions

There are several interesting directions for future research. First, we believe that better eigenvector-based geometric embeddings are still possible. Recall that the i^{th} eigenvector will "spread out" the netlist modules along the i^{th} coordinate axis, thus making a contribution to the distances among points in the overall embedding. However, each eigenvector x gives a one-dimensional placement which has squared wirelength equal to the magnitude of the corresponding eigenvalue. Since eigenvectors of the smaller eigenvalues correspond to better placements, these eigenvectors intuitively should make larger contributions to the distances among the points of the embedding. With this in mind, we believe that *scaling* heuristics may be useful for perturbing the eigenvector-based embedding. For example, one could compute the ratio of the i^{th}-smallest eigenvalue λ_i to the smallest eigenvalue λ_2, then reduce the components of the i^{th} eigenvector by this ratio. Second, we hope to improve the running time of the AGGLOM algorithm since this method

[12]The reader will observe from this description that our results are the best of $10 \times 20 = 200$ F-M executions for each benchmark, giving us an unfair advantage. With respect to this inequity, we can only offer the following two observations. First, our reported cut values occur fairly frequently throughout the 200 values, and in particular are always present for the lowest values $d \leq 3$; thus, if we were to restrict our clusterings to only $d \leq 3$, i.e., a total of 60 F-M runs per benchmark, we would obtain essentially the same result. (The fact that low-dimensional embeddings are superior for LKP is itself quite interesting; recall that for SKP, the best embedding dimension generally grew with the value of k.) Second, we were not able to obtain 200-trial results for the MBC and RW-ST algorithms; the results we quote from [24] used 20 random trials. However, it should be noted that we were able to execute standard F-M on the flat benchmark netlists for 200 trials; these resulting cut values are reported in parentheses in Table 5, and are very similar to the results for 20 trials. Thus, we believe that our improvements are "meaningful" despite the perceived inequity.

Test		Standard FM		MBC		RW-ST		AGGLOM	
Case	k	Areas	Cuts	Areas	Cuts	Areas	Cuts	Areas	Cuts
19ks	737	5501:5501	151 (140)	5501:5501	156	5501:5501	146	5501:5501	124
bm1	216	1740:1740	65 (61)	1740:1740	54	1740:1740	58	1740:1740	48
PrimGA1	191	1716:1715	66 (66)	1718:1713	48	1716:1715	47	1716:1715	49
PrimSC1	191	1377:1376	59 (59)	1377:1376	61	1377:1376	58	1377:1376	49
PrimGA2	702	4187:4186	242 (234)	4187:4186	187	4187:4186	165	4187:4186	146
PrimSC2	702	3853:3853	235 (235)	3858:3848	175	3853:3853	159	3853:3853	144
Test02	445	37132:19918	42 (42)	37132:19918	42	37132:19918	42	37132:19918	42
Test03	327	11115:11114	84 (84)	13729:8500	59	13188:9041	71	15211:7018	50
Test04	317	40732:1308	12 (12)	40938:1102	20	40932:1108	14	40732:1308	12
Test05	423	38753:33845	24 (24)	62586:10012	4	39089:33509	5	62513:10085	9
Test06	477	8484:8484	87 (65)	8484:8484	83	8484:8484	82	8477:8491	63

Table 6: Utility of AGGLOM results within two-phase Fiduccia-Mattheyses partitioning. We compare against the results for (i) standard F-M (on flat netlists), (ii) random matching-based clusterings, and (iii) RW-ST clusterings, all of which are quoted from [24]. For each benchmark, we use the same number of clusters k as was used by Hagen and Kahng in [24] (obtained via [21]). "Cuts" refers to the number of signal nets cut by the bipartition. Numbers in parentheses for Standard FM reflect results that are the best over 200 executions of the algorithm, rather than the 20 executions used in [24].

gave uniformly superior results to KCENTER. Currently, the $O(n^2)$ method of [4] for general graphs is also the most efficient method known for geometric instances. However, we believe that a subquadratic implementation would be possible using tools from computational geometry. Finally, additional geometric clustering algorithms hold promise. For example, Delattre and Hansen [12] observe that min-split and min-diameter seem to be conflicting goals: clusterings that minimize split tend to be long chains, while algorithms which try to minimize diameter give clusterings which are often poorly separated and unnatural. Consequently, [12] present an algorithm which generates a family of clusterings that trade off between split and diameter, allowing the user to pick the result which seems most natural.

In conclusion, we have presented new results showing that by first computing a fast geometric embedding, marked speedups in k-way netlist partitioning can be achieved. While this is in some sense an "old" idea, we have introduced basic advances that result in fast k-way partitioning solutions for the two distinct regimes of $k \ll n$ and $k = \Theta(n)$. These basic advances include: (i) a new "partitioning-specific" net model used in constructing the Laplacian of the netlist; (ii) the use of fast Lanczos implementations to compute the d-dimensional geometric embedding efficiently; and (iii) a careful choice of the clustering objective and the clustering methodology which together afford the best netlist partitions. We believe that the spectral geometric embedding preserves important graph properties with respect to partitioning and clustering; moreover, the simplicity of geometric clustering opens the door to many elegant heuristics whose results can be transferred back to the original netlist graph. Extensive experiments show that our new algorithms are not only efficient, but also actually *improve* solution quality over the best previous methods.

6 Acknowledgements

We are very grateful to the authors of [9], [24] and [55] for providing early access to these works.

References

[1] T. Asano, B. Bhattacharya, J. M. Keil, and F.F. Yao, "Clustering Algorithms Based on Minimum and Maximum Spanning Trees", in *Proc. 4th Annual ACM Symp. on Computational Geometry*", 1988, pp. 252-257.

[2] E.R. Barnes. An Algorithm for Partitioning the Nodes of a Graph. *SIAM J. Alg. Disc. Meth.*, 3(4):541–550, 1982.

[3] E. B. Baum, "Iterated descent: a Better Algorithm for Local Search in Combinatorial Optimization Problems", *technical report* TR-164-30, Crellin Laboratory, California Institute of Technology, 1990.

[4] 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, 1982, pp. 209-218.

[5] J. Blanks. Near-optimal Placement Using a Quadratic Objective Function. In *Proc. ACM/IEEE Design Automation Conf.*, pages 609–615, 1985.

[6] T. N. Bui, S. Chaudhuri, F. T. Leighton and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior", *Combinatorica* 7(2) (1987), pp. 171-191.

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

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

[9] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio Cut Partitioning", *technical report* TR-92-44, UC Santa Cruz CE Dept., September 1992 (also submitted to Symposium on Integrated Systems, Seattle, March 1993).

[10] C. K. Cheng and E. S. Kuh, "Module Placement Based on Resistive Network Optimization", *IEEE Trans. on CAD* 3(1984), pp. 218-225.

[11] H. E. Day and H. Edelsbrunner, "Efficient Algorithms for Agglomerative Heirarchical Clustering Methods", *Journal of Classification*, 1, 1984, pp. 7-24.

[12] M. Delattre and P. Hansen, "Bicriterion Cluster Analysis", *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-2(4) (1980), pp. 277-291.

[13] M. Delattre and P. Hansen, "Complete-link Cluster Analysis by Graph Coloring", *Journal of the American Statistical Association*, 73(1978), pp. 397-403.

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

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

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

[17] J. Frankle and R.M. Karp. Circuit placement and cost bounds by eigenvector decomposition. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1986, pp. 414-417.

[18] J. Garbers, H. J. Promel and A. Steger, "Finding Clusters in VLSI Circuits", (*preliminary version* of paper in) *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 520-523. Also *personal communication*, A. Steger, April 1992.

[19] G. Golub and C. Van Loan, *Matrix Computations*, Baltimore, Johns Hopkins University Press, 1983.

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

[21] L. Hagen, *personal communication*, August 1992.

[22] L. Hagen and A. B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1991, pp. 10-13.

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

[24] L. Hagen and A. B. Kahng, "A New Approach to Effective Circuit Clustering", in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1992; also issued as *technical report* UCLA CSD TR-920041.

[25] Scott W. Hadley, Brian L. Mark, and Anthony Vanelli. An Efficient Eigenvector Approach for Finding Netlist Partitions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-11(7):885–892, July 1992.

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

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

[28] P. Hansen and B. Jaumard, "Minimum Sum of Diameters Clustering", *Journal of Classification*, 4(1987), pp. 215-226.

[29] P. Hansen and B. Jaumard, "Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion" 1990.

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

[31] D. S. Hochbaum and D. B. Shmoys, "A Unified Approach to Approximation Algorithms for Bottleneck Problems", *Journal of the Association for Computing Machinery*, 33(3), 1986, pp. 533-550.

[32] A. Itai, Y. Perl and Y. Shiloach, "The Complexity of Finding Maximum Disjoint Paths with Length Constraints", *Networks* 12 (1982), pp. 277-286.

[33] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation, Part I, Graph Partitioning", *Journal of Operations Research* 37(1989), pp. 865-892.

[34] S. C. Johnson, "Hierarchical Clustering Schemes", *Psychometrika* 32(3) (1967), pp.241-254.

[35] S. Kang, "Linear Ordering and Application to Placement", *Proc. ACM/IEEE Design Automation Conf.* 1983, pp. 457-464.

[36] S. Kauffman and S. Levin, "Toward a General Theory of Adaptive Walks on Rugged Landscapes", *J. Theoretical Biology* 128 (1987), pp. 11-45.

[37] B. W. Kernighan and S. Lin, "An Efficient Heuristic for Partitioning Graphs", *Bell Syst. Tech. J.* 49(2) (1970), pp.291-307.

[38] T. Leighton and S. Rao, "An Approximate Max-Flow Min-Cut Theorem for Uniform Multicommodity Flow Problems with Applications to Approximation Algorithms", *IEEE Annual Symp. on Foundations of Computer Science*, 1988, pp. 422-431.

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

[40] N. Megiddo and K. J. Supowit, "On the Conplexity of Some Common Geometric Location Problems", *Siam Journal of Computing*, 13(1), 1984, pp. 182-196.

[41] B. Mohar, "Eigenvalues, Diameter, and Mean Distance in Graphs", *Graphs and Combinatorics*, 7 (1991), pp. 53-64.

[42] C. Monma and S. Suri, "Partitioning Points and Graphs to Minimize the Maximum or the Sum of Diameters", *Proc. Sixth Intl. Conf. Theory and Applications of Graphs*, 1988, pp. 899-912.

[43] F .P. Preparata and M .I. Shamos, *Computational Geometry*, 1, Springer Verlag, New York, 1985.

[44] R. L. Russo, P.H. Oden and P.K. Wolff, Sr., "A Heuristic Procedure for the Partitioning and Mapping of Computer Logic Graphs", *IEEE Trans. on Computers* vol. C-20, Dec. 1971, pp. 1455-1462.

[45] K. Roy and C. Sechen, "A Timing Driven N-Way Chip Partitioner", *Proc. MCNC Layout Synthesis Workshop*, May 1992, pp. 189-190.

[46] L.A. Sanchis. Multiple-way Network Partitioning. *IEEE Trans. on Computers*, 38:62–81, 1989.

[47] D. M. Schuler and E. G. Ulrich, "Clustering and Linear Placement", *Proc. IEEE Design Automation Workshop*, 1972, pp. 50-56.

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

[49] C. Sechen and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1987, pp. 478-481.

[50] R. S. Tsay and E. S. Kuh, "A Unified Approach to Partitioning and Placement" in *Proc. Princeton Conf. on Inf. and Comp.*, 1986.

[51] P.M. Vaidya, "An Optimal Algorithm for the All-Nearest-Neighbors Problem", *Proc. 27th IEEE FOCS*, 1986, pp.117-122.

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

[53] Y.C. Wei and C.K. Cheng, "A Two-Level Two-Way Partitioning Algorithm", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 516-519.

[54] 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.

[55] C. W. Yeh, C. K. Cheng and T. T. Lin, "A Probabilistic Multicommodity-Flow Solution to Circuit Clustering Problems", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, November 1992.

Appendix

We briefly note some intuitions regarding the algorithms which we use to address Formulation 2. Table 7 below compares AGGLOM, Divisive Min-Diameter, and the KCENTER algorithms for 200 random points in the 10000×10000 grid using Euclidean distance. For each k, we give the maximum diameter and error bound versus a lower bound for the optimal diamter value (this lower bound is given by the distance between the two closest of the $k+1$ centers chosen by KCENTER; recall the discussion of section 3.1 above). We observe that AGGLOM is far superior for large values of k while Divisive Min-Diameter generally does better for small k (indeed, it is optimal for $k = 2$). However, the table reveals Divisive Min-Diameter also can have an undesirable *dissection effect* [28] for certain values of k, especially for k one less than a power of two. For example, the algorithm gives an optimal solution for $k = 2$, but then to compute a solution for $k = 3$, it must subdivide one of two relatively equally sized clusters. The result contains one "large" cluster and two "small" clusters, yielding a poorly balanced partition. From our experiments, this effect seems strongly apparent for k up through $\theta(\sqrt{n})$. In general, we see the error increasing from $k = 2^j$ up through $k = 2^{j+1} - 1$ as the clustering becomes progressively more unbalanced; when k is again a power of two the size balance is restored.

When we consider the eigenvector-based geometric embeddings, the distribution of points is no longer uniform as in Table 7, but highly dense with some outliers (see Figure 1). Thus, the dissection effect becomes magnified as outliers become clustered with members of the dense "middle" cluster for $k = 2$, forcing an unnatural solution. For subseqent values of k, the middle cluster has been irrevocably split, making natural clustering very difficult. We hypothesize that the dissection effect accounts for the poor performance of the Divisive Min-Diameter in the SKP regime.

	AGGLOM		Divisive Min-Diam		KCENTER	
k	Max-Diam	Error	Max-Diam	Error	Max-Diam	Error
2	10791	1.21	10337	1.16	12666	1.42
3	9980	1.25	10137	1.27	10024	1.26
4	7519	1.29	6423	1.11	6837	1.18
5	6866	1.54	6215	1.39	7771	1.74
6	6239	1.51	6018	1.45	7310	1.76
7	5733	1.45	5786	1.46	6655	1.68
8	5339	1.43	5033	1.35	6075	1.63
9	4922	1.61	4891	1.60	5371	1.76
10	4610	1.59	4769	1.65	5067	1.75
11	4342	1.55	4675	1.67	4719	1.69
12	4119	1.57	4583	1.74	4371	1.66
13	3922	1.65	4460	1.87	4135	1.73
14	3719	1.65	4334	1.92	3954	1.75
15	3556	1.65	4147	1.92	3840	1.78
16	3394	1.64	3327	1.60	3704	1.79
17	3271	1.62	3168	1.57	3603	1.79
18	3143	1.60	3032	1.54	3506	1.78
19	3031	1.58	2954	1.54	3413	1.78
20	2910	1.56	2884	1.54	3302	1.77
21	2808	1.54	2813	1.55	3179	1.75
22	2719	1.54	2765	1.57	3083	1.75
23	2644	1.55	2704	1.59	2952	1.73
24	2549	1.55	2651	1.61	2856	1.73
25	2485	1.56	2601	1.63	2779	1.74
26	2410	1.55	2555	1.65	2716	1.75
27	2329	1.55	2508	1.67	2647	1.76
28	2271	1.55	2460	1.68	2575	1.75
29	2212	1.54	2398	1.67	2520	1.76
30	2152	1.53	2342	1.66	2471	1.75
31	2085	1.51	2289	1.66	2418	1.75
32	2029	1.50	2236	1.65	2371	1.75
50	1382	1.38	1683	1.68	1757	1.75
75	916	1.23	1112	1.49	1296	1.74
100	656	1.14	815	1.42	974	1.70
125	478	1.08	592	1.34	711	1.61
150	342	1.04	408	1.24	488	1.49
175	215	1.01	245	1.15	244	1.15

Table 7: Results of algorithms AGGLOM, Divisive Min-Diameter, and KCENTER for point sets of size 200 randomly chosen in the Euclidean unit square (average of 100 point sets). Points were chosen as integers in a 10000×10000 grid and the diameter of the largest cluster (rounded to the nearest integer) is given. An error bound was computed based on the distance between the two closest of the $k + 1$ centers chosen by the KCENTER algorithm. Divisive Min-Diameter generally yields the best results for SKP while AGGLOM is significantly better for LKP.