Journal of Classification 14:51-74 (1997)

Splitting an Ordering into a Partition to Minimize Diameter

Charles J. Alpert
IBM Austin Research Laboratory

Andrew B. Kahng
University of California at Los Angeles

Abstract: Many algorithms can find optimal bipartitions for various objectives including minimizing the maximum cluster diameter (''min-diameter''); these algorithms are often applied iteratively in top-down fashion to derive a partition P^k consisting of k clusters, with $k > 2$. Bottom-up agglomerative approaches are also commonly used to construct partitions, and we discuss these in terms of worst-case performance for metric data sets. Our main contribution derives from a new restricted partition formulation that requires each cluster to be an interval of a given *ordering* of the objects being clustered. Dynamic programming can optimally split such an ordering into a partition P^k for a large class of objectives that includes min-diameter. We explore a variety of ordering heuristics and show that our algorithm, when combined with an appropriate ordering heuristic, outperforms traditional algorithms on both random and non-random data sets.

Keywords: Dynamic programming; Vertex ordering; Restricted partition; Constrained clustering; Sequencing; Seriation; Diameter criterion; Partition in metric space.

1. Introduction

Given $G(V,E)$, a *cluster* is a subset of V and a *partition* P^k is set of k nonempty clusters $\{C_1, C_2, \ldots, C_k\}$ such that every $v_i \in V$ is a member of exactly one C_j, $1 \leq j \leq k$. An instance (G,d,k,f) of the partition problem

---

Partial support for this work was provided by a Department of Defense Graduate Fellowship and by NSF MIP-9110696 and MIP-9257982.

First author's address: Charles J. Alpert, IBM Austin Research Laboratory, 11400 Burnet Road, MS 9460, Austin, TX 78660 USA.

consists of:

- A weighted graph $G(V,E)$, with n vertices $V = \{v_1, v_2, \ldots, v_n\}$ and $E \subseteq V \times V$.
- A desired number of clusters k, with $1 < k < n$.
- A dissimilarity function d: $E \rightarrow \Re^+$, with $d(v_i, v_j)$ denoting the dissimilarity between vertices v_i and v_j. As examples, the $d(v_i, v_j)$ quantities may form the matrix of dissimilarities between all distinct pairs of n entities, or they may indicate the Euclidean distances between all distinct pairs of n data points in multidimensional space. In the latter case, we say (G,d,k,f) is a *metric instance*. Alternatively, $d(v_i, v_j)$ may denote the similarity between v_i and v_j, e.g., the connectivity between nodes in a network.
- An objective f: $P^k \rightarrow \Re^+$ that is a function of the partition.

For a given instance (G,d,k,f), we seek a partition P^k that optimizes $f(P^k)$. We use $\hat{P}^k$ to denote the optimal partition for f. Finally, an algorithm has a *performance ratio* of r if it always returns a solution P^k such that $f(P^k)/f(\hat{P}^k) \leq r$.

Much previous work has focused on cases where f is a function of *diameter* and/or *split*. The *diameter* of a cluster C is defined as $diam\ (C) = \max\ \{d(v_i, v_j) \mid v_i, v_j \in C\}$, and the *split* between two clusters C_1 and C_2 is defined as $split\ (C_1, C_2) = \min\ \{d(v_i, v_j) \mid v_i \in C_1, v_j \in C_2\}$. Diameter captures the notion that clusters are small and compact, while split maintains that clusters are well-separated. For example, the following objectives are well studied:

- **Min-Diameter:** minimize $f(P^k) = \max_{1 \leq i \leq k} diam(C_i)$.
- **Min-Sum-Diameters:** minimize $f(P^k) = \sum_{i=1}^{k} diam(C_i)$.
- **Max-Split:** maximize $f(P^k) = \min_{1 \leq i < j \leq k} split(C_i, C_j)$.

The min-diameter partition problem is NP-Complete in two or more dimensions for $k \geq 3$ (e.g., Brucker (1978) and Hansen and Delattre (1978)). Non-polynomial optimal algorithms have been proposed by Rao (1971), Hansen and Delattre (1978), and Guénoche (1993) for this case; such algorithms can be effective in practice provided n is not too large. Brucker (1978) showed that the min-diameter problem is solvable in polynomial time in one dimension. Rao (1971), Hubert (1973), and Monma and Suri (1989) proposed optimal polynomial algorithms for finding min-diameter partitions for $k = 2$, and speedups for metric instances have been found by Asano, Bhattacharya, Keil, and Yao (1986) and by Avis (1986). These results form the basis of top-down divisive hierarchical heuristics for $k > 2$, while an agglomerative

implementation of the classic "complete-linkage" method forms the basis of bottom-up approaches. For metric instances and other instances for which G satisfies the triangle inequality, min-diameter algorithms with a performance ratio of 2 have been given by Gonzalez (1985) and by Feder and Greene (1988). Brucker (1978) showed that the min-sum-diameters objective is also NP-hard for $k > 2$; optimal bipartition algorithms for this objective have been given by Hansen and Jaumard (1987), Monma and Suri (1989), and Hershberger (1991). Optimal max-split partitions can be found via the well-known "single-linkage" method; Hansen, Jaumard, and Frank (1989) and Hansen, Jaumard, and Musitu (1990) have addressed other split-related objectives. Finally, Delattre and Hansen (1980) and Glasbey (1987) have proposed approaches that trade off between diameter and split criteria.

In this work, we present a new non-hierarchical approach for constructing min-diameter partitions based on a *restricted partition* (RP) formulation. A *vertex ordering* is either a Hamiltonian path or a Hamiltonian circuit in G, and the RP formulation seeks a partition such that each cluster in the partition is an interval of the ordering. The RP formulation enables optimal partitions to be determined in $O(kn^3)$ time for Hamiltonian cycles using dynamic programming for a large class of objectives. For Hamiltonian paths, an $O(kn^2)$ optimal solution is possible. In addition, prescribed upper and lower bounds on cluster size can be handled transparently. For the min-diameter objective in particular, an $O(kn^2 \log n)$ implementation can be obtained for Hamiltonian cycles.

The remainder of this paper is organized as follows. Section 2 reviews the divisive min-diameter and complete-linkage algorithms for constructing min-diameter partitions. Section 3 describes the restricted partition formulation. Section 4 presents a dynamic programming algorithm for solving this formulation and describes extensions that afford possible speedups. Section 5 discusses several possible ordering heuristics, Section 6 presents experimental results, and we conclude in Section 7.

2. Review of Min-Diameter Approaches

The Divisive Min-Diameter (DQ) Algorithm

Many algorithms can find optimal min-diameter bipartitions, e.g., by bicoloring a maximum spanning tree over G and assigning vertices with the same color to the same cluster (Hubert 1973). To construct a hierarchy of (possibly non-optimal) partitions, a given bipartition algorithm can be iteratively applied to the largest remaining cluster: this "divisive min-diameter" (DQ) algorithm is shown in Figure 1. Iteratively applying Hubert's (1973) algorithm yields an $O(kn^2)$ implementation for DQ. Guénoche, Hansen, and

Divisive Min-Diameter (DQ) Algorithm $(G, d, k, f = \textit{min-diameter})$
Output: $P^k = \{C_1, C_2, \ldots, C_k\} \equiv$ Partition into k clusters **Variables:** $P^1, P^2, \ldots, P^k \equiv$ Set of partitions
1. Set $C_1 = V$ and $P^1 = \{C_1\}$ 2. **for** $m = 1$ **to** $k - 1$ **do** 3. Let $C_i \in P^m$ be such that $diam(C_i)$ is maximum 4. Apply an optimal min-diameter bipartition algorithm to C_i, yielding C', C'' 5. $P^{m+1} = (P^m \cup \{C'\} \cup \{C''\}) - \{C_i\}$

Figure 1. Divisive Min-Diameter (DQ) Algorithm.

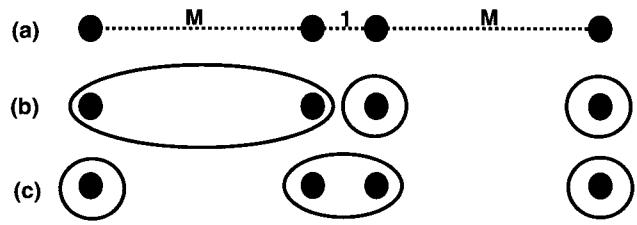


Figure 2. Illustration of the worst-case performance of DQ. Part (a) shows a set of points on the line using Euclidean distances. Part (b) shows the DQ solution P^3 with diameter M. Part (c) shows the optimal solution $\hat{P}^3$ with diameter 1; hence, the ratio of the DQ solution diameter to the optimal diameter is M. Since M is an arbitrary constant, the performance ration for DQ with $k = 3$ is infinite.

Complete-Linkage (CL) Algorithm $(G, d, k, f = \textit{min-diameter})$
Output: $P^k = \{C_1, C_2, \ldots, C_k\} \equiv$ Partition into k clusters **Variables:** $P^n, P^{n-1}, \ldots, P^k \equiv$ Set of partitions
1. **for** $i = 1$ **to** n **do** $C_i = \{v_i\}$ 2. $P^n = \{C_1, C_2, \ldots, C_n\}$ 3. **for** $m = n$ **downto** $k + 1$ **do** 4. Find clusters C_i, C_j such that $diam(C_i \cup C_j)$ is minimum 5. $P^{m-1} = (P^m \cup \{C_i \cup C_j\}) - C_i - C_j$.

Figure 3. Complete-Linkage (CL) Algorithm.

Jaumard (1991) showed how to modify Rao's (1971) algorithm to yield an $O(n^2 \log n)$ implementation for the whole hierarchy.

Remark 1: For the min-diameter objective and $k \geq 3$, DQ may have an infinite performance ratio, even for metric instances (see Figure 2). ▪

In practice, such pathological behavior is a real concern, e.g., a partition P^3 will likely be unbalanced when the partition P^2 above P^3 in the hierarchy is balanced (see Section 6).

The Complete-Linkage (CL) Agglomerative Algorithm

Complete-linkage (CL) can be programmed as a bottom-up approach: each vertex begins in its own cluster, and iteratively a pair of clusters is merged into a single cluster such that the increase in the maximum cluster diameter is minimized (see Figure 3). Benzécri (1982) has given an $O(n^2)$ implementation using chains of nearest neighbors.

Remark 2: For weighted graph instances, complete-linkage has an infinite performance ratio (see Figure 4). ▪

For metric instances, a tight worst-case performance ratio is not known. For 1-dimensional data sets, it can be shown that for $k = 2$ and $k = 3$, CL has a performance ratio of 2 for the min-diameter objective, and that this bound is tight. However, this bound does not hold for $k = 4$; to our knowledge, the following provides the first demonstration that CL does not have a performance ratio of 2.

Remark 3: For metric instances, CL may construct a partition with diameter more than twice optimal, and has a performance ratio of at least 2.5.

Proof: Consider the instance in Figure 5(b). A solid line segment denotes an existing cluster that may have been formed from an arbitrary number of points lying on the segment, along with two points at the ends of the segment. The instance shown has 14 clusters. The points that create the clusters with diameter 1.1 (e.g., the dotted box contains two such clusters) must be distributed more carefully so that no cluster contains two of the endpoints at distance 0.2 from each other. A possible distribution of these points is given in (h). The execution of CL with appropriate tie-breaking on this instance is shown in (c)-(g); the result is a 4-cluster partition with largest cluster diameter 12.4, which is 12.4/5 = 2.48 times worse than the largest cluster diameter

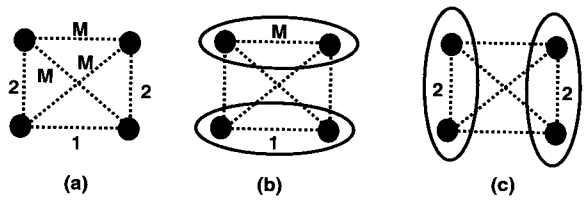


Figure 4. Illustration of the worst-case performance of CL. Part (a) shows a non-metric instance over 4 points. Part (b) shows a possible CL bipartition with maximum diameter M: CL will first merge the bottom two points into a cluster, then any subsequent merge sill create a cluster with diameter M. Part (c) shows the optimal bipartition with maximum diameter 2, hence the ratio of the CL solution diameter to the optimal diameter is $M/2$. Since M is an arbitrary constant, the performance ratio for CL with non-metric instances is infinite.

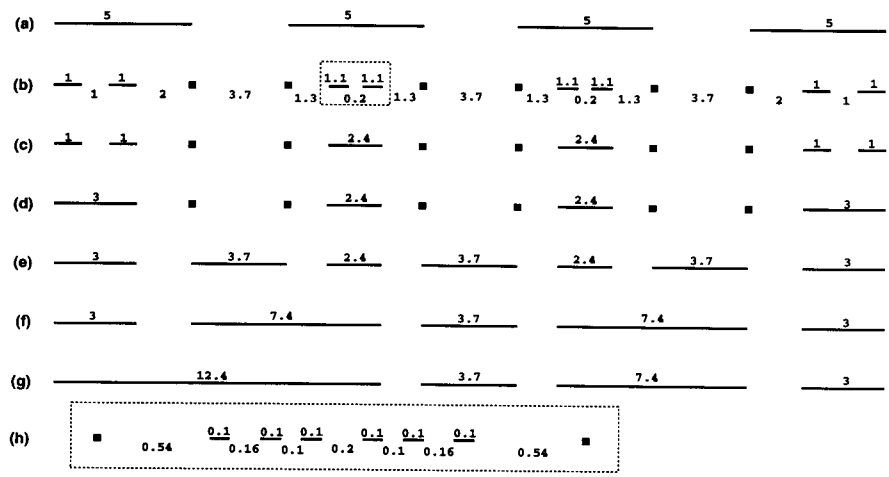


Figure 5. (a) gives the optimal solution $\hat{P}^4$ for (b) the geometric problem instance. (c)-(g) show the execution of CL on this instance. The resulting maximum diameter of 12.4 gives CL a performance ratio of 2.48 for this instance.

in the optimal solution (a). Distances in (b) can be slightly perturbed to avoid ties and deterministically yield this partition; the instance can also be perturbed to force the performance ratio arbitrarily close to 2.5. ■

Tight upper bounds on the performance ratio of CL remain unknown for 1-dimensional and multidimensional metric instances and for weighted graphs that satisfy the triangle inequality.

3. The Restricted Partition (RP) Formulation

While the above approaches use the min-diameter objective, their underlying paradigms can be extended to other objectives. We now present a new approach for constructing min-diameter partitions; because our approach easily extends to other objectives, we cast the discussion in terms of a generic objective f.

A well-known Traveling Salesman Problem (TSP) heuristic of Karp (1977) uses a partition of a planar data set to construct a heuristic tour (i.e., a Hamiltonian cycle): every point in a given cluster is visited before the tour moves on to the next cluster, until all points in all clusters have been visited. The genesis of our approach lies in asking whether an "inverse" methodology can succeed, i.e., whether we can use a tour of the vertices to generate a partition. Given a tour, we require each cluster to be an interval of the tour, thereby obtaining the following general approach: (i) construct a "good tour" over the vertices, then (ii) split the tour to obtain a partition into k clusters. We will use the term *ordering* to refer to either a tour or a *linear ordering* (i.e., a Hamiltonian path); the context should be clear from the discussion.

We represent an ordering $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}$ over $V = \{v_1, v_2, \ldots, v_n\}$ by the bijection π: $[1 \cdots n] \rightarrow [1 \cdots n]$. We say that v_i is the j-th vertex in the ordering if $\pi(j) = i$ (so $v_i = v_{\pi(j)}$). In other words, $v_{\pi(1)}$ is the first vertex in the ordering, $v_{\pi(2)}$ is the second, etc. An interval $[i,j]$ 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 $i > j$. An instance of the restricted partition (RP) problem consists of a partition instance (G, d, k, f), a vertex ordering pi, and lower and upper cluster size bounds L and U. While not necessarily an intuitive part of the RP formulation, cluster size bounds are easily integrated into the algorithms that we present.

The Restricted Partition Problem (RP): Given a restricted partition instance (G, d, k, f, π, L, U), find a partition P^k that optimizes $f(P^k)$ such that the following conditions hold.

> **Condition 1:** if $v_{\pi(i)}, v_{\pi(j)} \in C$ for some cluster $C \in P^k$, and $i < j$, 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 intervals of π, and Condition 2 enforces cluster size bounds.

4. Splitting Orderings Optimally

We apply an $O(kn^2(U-L))$ dynamic programming algorithm to solve the RP formulation optimally. Removing Condition 1(b) from RP requires clusters to be intervals from a *linear ordering* rather than from a tour, and allows an $O(n)$ factor speedup. Other speedups are possible when addressing specific objectives, e.g., we give an $O(nU \log n)$ solution for the min-diameter objective.

4.1 A Dynamic Programming Solution

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., we may write the min-diameter objective as $f(P^k) = \max_{1 \le i \le k} w(C_i)$ where $w(C_i) = diam\ (C_i)$. The cluster corresponding to interval $[i,j]$ is denoted by $C_{[i,j]}$, and we say that i and j are respectively the left and right *endpoints* of the cluster. We let $P^k_{[i,j]}$ denote a restricted partition of the interval $[i,j]$ into k clusters. Notice that $P^1_{[i,j]} = \{C_{[i,j]}\}$ is the optimal partition into one cluster over the interval $[i,j]$. The set of partitions $P^k_{[i,j]}$ will serve as "building blocks" for solutions of the form $P^{k'}_{[i',j']}$ where $[i,j] \subset [i',j']$ and $k < k'$.

Since each cluster $C_{[i,j]}$ is uniquely determined by its first and last vertices $v_{\pi(i)}$ and $v_{\pi(j)}$, only $(U-L+1)n$ clusters can be part of any RP solution. Our 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 4.2). These clusters form the set of all optimal partitions of the form $P^1_{[i,j]}$. We then build partitions into two clusters $P^2_{[i,j]}$ from the $P^1_{[i,j]}$ solutions, etc., until a partition into k clusters is derived over the interval $[i, i-1]$. (All index manipulations are performed modulo n, i.e., $i + j \equiv (i + j - 1)\ mod\ (n+1)$, so that $[i, i-1] = [1,n]$ if $i = 1$.) Figure 6 formally describes this algorithm, which we call DP-RP for "Dynamic Programming for Restricted Partitions".

As an example, consider the ordering of 8 nodes shown in Figure 7. We show the DP-RP construction of the partition $P^3_{[1,8]}$ with $L = 2$ and $U = 5$, by considering all combinations of the partition $P^2_{[1,m]}$ with $C_{[m+1,8]}$ $(3 \le m \le 6)$ as in Steps 6 and 7 of Figure 6. The partitions $P^2_{[1,m]}$ $(3 \le m \le 6)$ are assumed to have been previously constructed. The solution assigned to $P^3_{[1,8]}$ is $P^2_{[1,m]} \cup \{C_{[m+1,8]}\}$, where m is chosen to minimize $f(P^2_{[1,m]} \cup \{C_{[m+1,8]}\})$. Notice that in (a), the cost of the partition $P^2_{[1,3]}$ is infinite since it cannot satisfy the cluster size lower bound constraint.

DP-RP Algorithm (G, d, k, f, π, L, U)

Output: $P^k \equiv$ Optimal restricted partition

Variables: $P^{k'}_{[i,j]} \equiv$ Subsolutions

$k' \equiv$ Index denoting current number of clusters

$m+1 \equiv$ Beginning index of possible new cluster

$f_{best} \equiv$ Value of objective function for best current $P^{k'}_{[i,j]}$

1. **for each** i, j **do** compute $f(P^1_{[i,j]}) = w(C_{[i,j]})$ 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** $f_{best} > f(P^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\})$ **then**
7. $f_{best} = f(P^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\})$, $P^{k'}_{[i,j]} = P^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\}$
8. **return** $P^k = P^k_{[i,i-1]}$ for the i that minimizes $f(P^k_{[i,i-1]}), 1 \le i \le n$

Figure 6. The DP-RP algorithm. The template assumes that f is to be minimized. To maximize f, Steps 4, 6, and 8 are changed in the obvious manner.

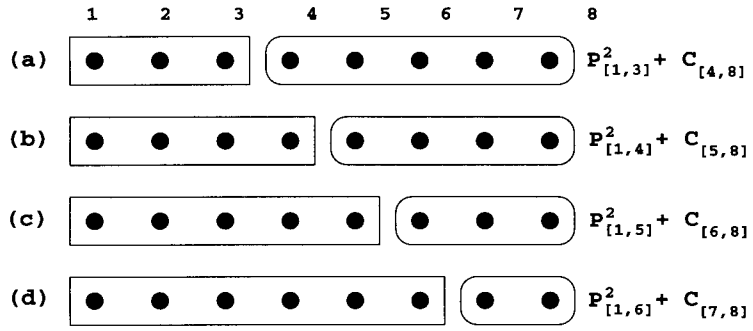


Figure 7. DR-RP example. For an ordering of 8 nodes, with $L = 2$ and $U = 5$, the DP-RP algorithm constructs a partition $P^3_{[1,8]}$ by choosing the best combination of a partition $P^2_{[1,m]}$ with a cluster $C_{[m+1,8]}$ for m equal to (a) 3, (b) 4, (c) 5, and (d) 6. Note that the solution in (a) uses a partition $P^2_{[1,3]}$ which cannot exist, so (a) cannot be chosen.

Any partition P^k can be expressed as the union of a set of subpartitions $P_1, P_2, \ldots, P_r$. If two partitions P^k and Q^k are expressible as

$P_1 \cup P_2 \cup \cdots \cup P_r$ and $Q_1 \cup Q_2 \cup \cdots \cup Q_r$ respectively, with $f(\{P_i\}) \leq f(\{Q_i\})$ for $1 \leq i \leq r$, then f is *monotone nondecreasing* (monotone nonincreasing) if and only if $f(P^k) \leq f(Q^k)$ ($f(P^k) \geq f(Q^k)$) for all such P^k and Q^k. For example, the min-diameter and sum-of-diameters objectives are monotone nondecreasing functions of the diameter.[1] We have the following result:

Theorem 1: *If f is monotone nondecreasing, DP-RP returns an optimal restricted partition for the instance (G, d, k, f, π, L, U).*

Proof: Let $\hat{P}^k = \{C_1, C_2, \ldots, C_k\}$ be an optimal restricted partition for (G, d, k, f, π, L, U), and let i be the left endpoint of cluster C_1. Assume that the clusters in $\hat{P}^k$ are labeled consecutively with respect to π, i.e., if cluster C_j has right endpoint m, then cluster C_{j+1} has left endpoint $m + 1$. We proceed by induction on k' where $k' \leq k$.

Inductive Hypothesis: Let j be the right endpoint of cluster $C_{k'}$. Then DP-RP returns (or stores in memory) a partition $P^{k'}_{[i,j]}$ with $f(P^{k'}_{[i,j]}) \leq f(\{C_1, C_2, \ldots, C_{k'}\})$.

Basis: ($k' = 1$) DP-RP computes the partition $P^1_{[i,j]} = C_{[i,j]} = C_1$ over the interval $[i,j]$, hence $f(P^1_{[i,j]}) \leq f(\{C_1\})$.

Induction: Let m be the right endpoint of cluster $C_{k'-1}$ (so $C_{k'} = C_{[m+1,j]}$). Assume the hypothesis holds for $k' - 1$, i.e., DP-RP finds a partition $P^{k'-1}_{[i,m]}$ with $f(P^{k'-1}_{[i,m]}) \leq f(\{C_1, C_2, \ldots, C_{k'-1}\})$. We show that the hypothesis also

---

1. In addition, common graph and hypergraph partition objectives from the communications and VLSI CAD literatures are also monotone. Let $H(V,E)$ denote a hypergraph where a hyperedge $e \in E$ denotes a subset of V with size at least 2, and let $E_i = \{e \in E \mid \exists\, u,v \in e, u \in C_i, v \notin C_i\}$ be the set of hyperedges cut by cluster C_i. The following objectives are monotone:

- Min Cut: Minimize:
$$f(P^k) = \sum_{1 \leq i \leq k} w(C_i) \text{ with } w(C_i) = |E_i|$$
- Scaled Cost (Chan, Schlag, and Zien 1994): Minimize:
$$f(P^k) = \frac{1}{n(k-1)} \sum_{1 \leq i \leq k} w(C_i) \text{ with } w(C_i) = \frac{|E_i|}{|C_i|}$$
- Absorption (Sun and Sechen 1993): Maximize:
$$f(P^k) = \sum_{1 \leq i \leq k} w(C_i) \text{ with } w(C_i) = \sum_{\{e \in E \mid e \cap C_i \neq \emptyset\}} \frac{|e \cap C_i| - 1}{|e| - 1}$$

holds for k'. Express the partition $\{C_1, C_2, \ldots, C_{k'}\}$ as $P_1 \cup P_2$ where $P_1 = \{C_1, C_2, \ldots, C_{k'-1}\}$ and $P_2 = \{C_{k'}\}$. Consider the partition $Q^{k'}_{[i,j]} = P^{k'-1}_{[i,m]} \cup \{C_{[m+1,j]}\}$, which can be expressed as $P_3 \cup P_2$ where $P_3 = P^{k'-1}_{[i,m]}$. By the inductive hypothesis $f(P_3) \leq f(P_1)$, and since $f(P_2) \leq f(P_2)$, the fact that f is monotone nondecreasing implies $f(Q^{k'}_{[i,j]}) \leq f(\{C_1, C_2, \ldots, C_{k'}\})$. The algorithm finds and evaluates the partition $Q^{k'}_{[i,j]}$ in Step 6 (Figure 6). DP-RP considers other solutions as well, storing the best one as $P^{k'}_{[i,j]}$; hence $f(P^{k'}_{[i,j]}) \leq f(Q^{k'}_{[i,j]})$. This implies that $f(P^{k'}_{[i,j]}) \leq f(\{C_1, C_2, \ldots, C_{k'}\})$, proving the hypothesis.

Thus, the algorithm stores a partition $P^k_{[i,i-1]}$ such that $f(P^k_{[i,i-1]}) \leq f(\hat{P}^k)$, and in Step 8 it returns a partition P^k with $f(P^k) \leq f(P^k_{[i,i-1]})$. ■

The proof can be slightly modified for the case of monotone nonincreasing functions.

4.2 Computing Cluster Costs

For the case of $w(C_i) = diam\ (C_i)$, Figure 8 gives a simple $O(nU)$ implementation of Cluster_Costs. The algorithm starts with clusters $C_{[i,i]}$ of diameter zero. The key observation is that any edge within cluster $C_{[i,j]}$ either is (i) contained in $C_{[i+1,j]}$, or (ii) contained in $C_{[i,j-1]}$, or (iii) is the edge (v_i, v_j). Step 5 obtains the diameter of each new cluster in constant time, yielding the $O(nU)$ complexity bound.

Theorem 2: *With an $O(nU)$ implementation of Cluster_Costs, DP-RP has $O(k(U-L)n^2)$ time complexity, and $O(kn^3)$ complexity when there are no cluster size bounds.*

Proof: From Figure 6, Step 1 takes $O(nU)$ time. Steps 6 and 7 take constant time, so the loop in Step 5 takes $O(U-L)$ time. This loop is executed kn^2 times from the loops in Steps 2 and 3, yielding $O(k(U-L)n^2)$ overall complexity. If $U = n$ and $L = 1$, the complexity becomes $O(kn^3)$. ■

4.3 A More Efficient Min-Diameter Implementation

For certain choices of f and w, dynamic programming may not be necessary. In particular, an $\frac{n}{\log n}$ speedup is possible when the objective is of the form $f(P^k) = \max_{1 \leq i \leq k} w(C_i)$. Here we will restrict ourselves to the case $L = 1$. (Although this restriction is not required, extending this special case to encompass $L > 1$ is nontrivial.) For this case, $f(P^k)$ can take on only a polynomial number of possible values: since there are at most nU possible clusters, there can be at most nU possible values for $f(P^k)$ despite an exponential number of possible partitions P^k.

Cluster_Costs (G, d, π, U)
Output: $w(C_{[i,j]}) = diam(C_{[i,j]})$ for every cluster $C_{[i,j]}$ with no more than U vertices **Variables:** δ - one less than size of current clusters
1. **for** $i = 1$ **to** n **do** $w(C_{[i,i]}) = 0$ 2. **for** $\delta = 1$ **to** $U - 1$ **do** 3. **for** $i = 1$ **to** n **do** 4. $j = ((i + \delta - 1)$ **mod** $n) + 1$ 5. $w(C_{[i,j]}) = \max\{w(C_{[i,j-1]}), w(C_{[i+1,j]}), d(v_{\pi(i)}, v_{\pi(j)})\}$

Figure 8. Cluster_Costs for $w(C_i) = diam\ (C_i)$.

Decide_Cluster $(G, d, k, f = min\text{-}diameter, \pi, L = 1, U), M$
Input: $M \equiv$ Upper cost bound on $w(C)$ for every C **Output:** P^k iff $\exists P^k$ with $f(P^k) \leq M$, NO otherwise **Variables:** $first \equiv$ Leftmost index of cluster C_1 $k' \equiv$ Index denoting current number of clusters
1. **for** $first = 1$ **to** U **do** 2. $i = j = first$; $k' = 1$ 3. **repeat** 4. **while** $((\lvert C_{[i,j]}\rvert \leq U)$ **and** $(w(C_{[i,j]}) \leq M)$ **and** $(j \neq first - 1))$ **do** $j = j + 1$ 5. $C_{k'} = C_{[i,j-1]}$; $i = j$; $k' = k' + 1$ 6. **until** $(j = first$ **or** $k' = k)$ 7. **if** $(k' < k$ **and** $j = first)$ **return** $P^{k'} = \{C_1, C_2, \ldots, C_{k'}\}$ 8. **return** NO

Figure 9. Decide_Cluster Algorithm.

Consider the decision question, "does there exist an RP solution P^k with $f(P^k) \leq M$?" Given an oracle that answers this question in $O(T)$ time for any given value of M, we can solve the RP formulation by computing all cluster costs and performing a binary search over the nU possible cost values, using a total of $O(nU + T \log n)$ time. We may implement the oracle efficiently if w is *monotone nondecreasing* in the size of the cluster (note this definition is different from the monotonicity of f in Section 4.1), i.e., if $[i,j] \subset [i', j']$, then $w(C_{[i,j]}) \leq w(C_{[i',j']})$. For instance, $w(C) = diam\ (C)$ is monotone nondecreasing. Monotonicity of w allows us to solve the decision problem greedily by simply growing each cluster C_i as large as possible while keeping $w(C_i) \leq M$.

Figure 9 describes the Decide_Cluster algorithm that achieves the desired oracle. Decide_Cluster begins with $v_{\pi(first)}$ as the left endpoint of C_1 (initially *first* = 1) and traverses the ordering while constructing each cluster

to be as large as possible. When a cluster violates either the size or cost constraint (failure of Step 4), Decide_Cluster stores the cluster from just prior to the constraint violation (Step 5) and begins constructing a new cluster. The loop repeats until k clusters are generated or until every vertex belongs to a cluster (Step 7). If every vertex belongs to a cluster then a legal P^k exists and Decide_Cluster exits; otherwise, Decide_Cluster starts constructing a new partition with $v_{\pi(first+1)}$ as the left endpoint for C_1. If all possible values for first fail, then no partition exists with $f(P^k) \leq M$ and Decide_Cluster returns NO. Decide_Cluster has $O(nU)$ time complexity, giving us an $O(nU \log n)$ RP solution for min-diameter. We now prove the correctness of Decide_Cluster.

Theorem 3: *Decide_Cluster returns a $P^{k'}$ if and only if such an RP solution with $f(P^k) \leq M$ exists.*

Proof: If Decide_Cluster returns $P^{k'} = \{C_1, C_2, \ldots, C_{k'}\}$ with $k' < k$, we can easily form a P^k without increasing f by arbitrarily splitting clusters. Assume that Decide_Cluster returns NO but there exists a partition $Q^k = \{C'_1, C'_2, \ldots, C'_k\}$ with $f(Q^k) \leq M$. Assume that the left endpoint of C'_1 is $v_{\pi(first)} \equiv v_{\pi(i)}$ where i is the smallest index of all left cluster endpoints in Q^k, and that the clusters of Q^k are labeled consecutively with respect to π. Let $P^{k'} = \{C_1, C_2, \ldots, C_{k'}\}$ be the partition computed by Decide_Cluster with $v_{\pi(first)}$ as the left endpoint for C_1.

Inductive Hypothesis: $|C'_1| + |C'_2| + \ldots |C'_m| \leq |C_1| + |C_2| + \ldots |C_m|$ for $1 \leq m \leq k'$.

Basis ($m = 1$): Decide_Cluster constructs C_1 to be as large as possible while satisfying $|C_1| \leq U$ and $w(C_1) \leq M$. Since C'_1 also satisfies these constraints, $|C'_1| \leq |C_1|$.

Induction: Assume $|C'_1| + |C'_2| + \ldots |C'_m| \leq |C_1| + |C_2| + \ldots |C_m|$ holds for m~; we will show this inequality holds for $m + 1$. If not, we must have $C_{m+1} \subset C'_{m+1}$. We can write $C_{m+1} = C_{[i,j]}$ and $C'_{m+1} = C_{[i',j']}$ for $i \geq i'$ and $j < j'$. But since $w(C_{[i',j']}) \leq M$, $C_{[i,j]}$ can be expanded to $C_{[i,j']}$ while still satisfying $w(C_{[i,j']}) \leq M$ (since w is monotone nondecreasing). This is a contradiction since Decide_Cluster constructs a maximal C_{m+1}.

The inductive hypothesis implies that for any subsolution $Q^m_{[first,j]}$ with $f(Q^m_{[first,j]}) \leq M$, Decide_Cluster will find a subsolution $P^{m'}_{[first,j']}$ with $f(P^{m'}_{[first,j']}) \leq M$ with $m' \leq m$ and $j' \geq j$. Thus, if $f(Q^k_{[first,first-1]}) \leq M$, then Decide_Cluster returns $P^{k'}_{[first,first-1]}$ with cost also bounded by M and with $k' \leq k$.

DP-RP Algorithm for Linear Orderings (G, d, k, f, π, L, U)

Output: $P^k \equiv$ Optimal RP solution (without Condition 1(b))

Variables: $P^{k'}_{[i,j]} \equiv$ Subsolutions

$k' \equiv$ Index denoting current number of clusters

$m+1 \equiv$ Starting index of possible new cluster

$f_{best} \equiv$ Objective value for best current $P^{k'}_{[i,j]}$

1. **for each** i, j **do** compute $f(P^1_{[i,j]}) = w(C_{[i,j]})$ 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** $f_{best} > f(P^{k'-1}_{[1,m]} \cup \{C_{[m+1,j]}\})$ **then**
7. $f_{best} = f(P^{k'-1}_{[1,m]} \cup \{C_{[m+1,j]}\})$, $P^{k'}_{[1,j]} = P^{k'-1}_{[1,m]} \cup \{C_{[m+1,j]}\}$
8. **return** $P^k = P^k_{[1,n]}$

Figure 10. DP-RP Algorithm for Linear Orderings. The template assumes the f is to be minimized. To maximize f, Steps 4 and 6 are changed in the obvious manner.

4.4 Linear Orderings

So far, we have considered the RP formulation where both Conditions 1(a) and 1(b) apply. In this case, DP-RP may have up to $O(kn^3)$ complexity, which may be prohibitive for large instances. However, eliminating Condition 1(b) changes π from a tour into a linear ordering, restricting the solution space but allowing a factor of n speedup (see Figure 10). The speedup arises since we are guaranteed that some cluster has $v_{\pi(1)}$ as its left endpoint: for each value of k', we need record only $O(n)$ subsolutions of the form $P^{k'}_{[1,j]}$ instead of $O(n^2)$ optimal subsolutions. Since Steps 3-7 have time complexity $O(n(U-L))$ (as opposed to $O(n^2(U-L))$ in Figure 6), and since Step 1 has $O(nU)$ time complexity, a linear ordering can be optimally split into a restricted partition in $O(kn(U-L))$ time.

5. Possible Ordering Heuristics

For DP-RP to work well, a "good" ordering is required, yet it is not completely clear what criteria apply in constructing such an ordering. Intuitively, the ordering should correspond to a low-cost tour if small consecutive subsequences of points are to form low-diameter clusters. In addition, the ordering should reveal "natural structure", visiting an entire cluster before moving to the next one, as opposed to wandering out of and then back into

the same cluster.

One problem formulation that captures ideas similar to these is called *seriation*, i.e., metric unidimensional scaling. The n points are to be respectively assigned to 1-dimensional coordinates $\mathbf{x} = \{x_1, \ldots, x_n\}$ to minimize the objective $\Phi(\mathbf{x}) = \Sigma_{1 \le i < j \le n} (d_{ij} - |x_i - x_j|)^2$. An ordering π can then be induced by sorting the coordinates of $\mathbf{x}$. Given the correct objective, points v_i and v_j which are close to each other should have similar x_i and x_j coordinates; v_i and v_j far from each other should have a large difference between x_i and x_j. Seriation is well-studied; it was first formulated as a combinatorial optimization by Defays (1978), and various other approaches have been proposed including dynamic programming (Hubert and Arabie 1986), branch search (Defays 1978) and smoothing (Pliner 1986, 1996). However, it is not clear how well the $(d_{ij} - |x_i - x_j|)^2$ terms capture our loose intuition. Nor is it clear whether seriation is a more appropriate framework than, e.g., the squared linear placement objective of minimizing $\Sigma_{1 \le 1 < j \le n} d_{ij}(x_i - x_j)^2$ (Hall 1970). We leave open the effectiveness of ordering heuristics based on such formulations.

Because of their widespread availability, ease of implementation and previous heuristic study, we choose to apply three ordering heuristics from the traveling salesman problem (TSP) literature; these are the 3-Opt heuristic of Lin (1965), the heuristic of Lin and Kernighan (1973), and the spacefilling curve heuristic of Bartholdi and Platzman (1989). Johnson (1990) has reported that 3-Opt and Lin-Kernighan (along with an iterative version of Lin-Kernighan) generally return tours with very low cost using very little runtime, and that these heuristics are preferable to constructive, simulated annealing, and genetic approaches.

The 3-Opt heuristic performs local optimization by iteratively constructing an improved solution from the current solution. The iteration finds three edges in the current tour which can be deleted and replaced by three new edges to yield a lower-cost tour. 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, i.e., the edges (v_h, v_{h+1}), (v_i, v_{i+1}) and (v_j, v_{j+1}) are deleted and replaced by the edges (v_h, v_{i+1}), (v_j, v_{h+1}) and (v_i, v_{j+1}). (In general, there are four different sets of edges that can possibly replace a given triple of deleted edges.) When no triple of edges can be profitably replaced, the algorithm terminates and returns the current (local minimum) tour. We execute a single run ("descent") of 3-Opt from a random starting tour to generate the partitions discussed in the next section.

Lin-Kernighan (Lin-K) also uses iterative improvement, but with a powerful asymmetric neighborhood structure that can avoid, e.g., 3-Opt local minima. Given a tour T, Lin-K generates perturbations of T into a sequence

of Hamiltonian paths, then chooses the prefix of this sequence which results in the Hamiltonian path whose completion into a tour has lowest cost. Initially, Lin-K finds edges $x_1 \in T$ and $y_1 \notin T$ incident to the same vertex, such that y_1 has smaller cost than x_1. Edge x_1 is removed from T to yield a Hamiltonian path; then, edge y_1 is added to T, which determines a unique edge x_2 whose removal from T will maintain a Hamiltonian path. A tour can then be completed by joining the endpoints of the path using edge $\hat{y}_2$, or a new Hamiltonian path can be computed. The Hamiltonian paths in the sequence have edges $\{x_1, \ldots, x_i\}$ removed from T and edges $\{y_1, \ldots, y_{i-1}\}$ added to T. In iteration i, KL considers all edges $y_i \notin T$ such that the cost of edges $y_1, \ldots, y_i$ minus the cost of edges $x_1, \ldots, x_i$ is less than zero. If no such edge y_i exists, the procedure terminates and KL joins the endpoints of the Hamiltonian path by edge $\hat{y}_i$ to complete the tour. Of these edges y_i, the one which maximizes the difference in cost between x_{i+1} and y_i (x_{i+1} is the unique edge that must be removed from T to form a Hamiltonian path) is added to T, and x_{i+1} is removed from T. When the iteration terminates (say, at j), KL evaluates the cost of each tour $T_i = T - \{x_1, \ldots, x_i\} + \{y_1, \ldots, y_{i-1}, \hat{y}_i\}$, $2 \le i \le j$ constructed during the procedure; the tour with lowest cost becomes the new current tour T, and the entire process is repeated until no further improvement is possible. Empirical studies by Lin and Kernighan (1973) on Euclidean and nonmetric problems have shown that the run time in practice grows approximately as $n^{2.2}$; however, the algorithm is PLS-complete (Papadimitriou 1992), i.e., no polynomial bounds on the number of iterations exist.

The spacefilling curve (SFC) heuristic of Bartholdi and Platzman (1989) uses a recursive construction of Sierpinski (1912), the 2-dimensional case of which is shown in Figure 11. In the Figure, the successive approximations become progressively refined until the curve "fills" up the unit square, passing arbitrarily close to every point. For a given fixed precision, the curve actually passes through every point, and the order in which points are visited by the curve yields the heuristic SFC ordering. The tour can be computed in $O(n \log n)$ time and can be extended to data sets in arbitrary dimensions. The SFC ordering will typically have higher cost than its 3-Opt or Lin-Kernighan counterparts, but offers the possible advantage of a predetermined structure in that it entirely visits one orthant before moving to the next. Figure 12 shows the tours constructed by (a) 3-Opt, (b) Lin-Kernighan and (c) SFC on a uniformly random instance of 200 points in the unit square.

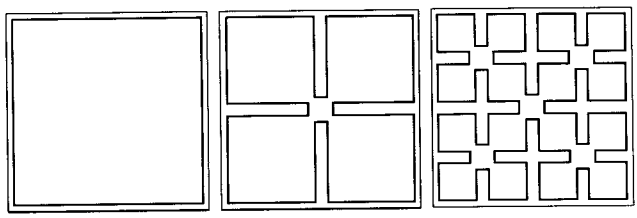
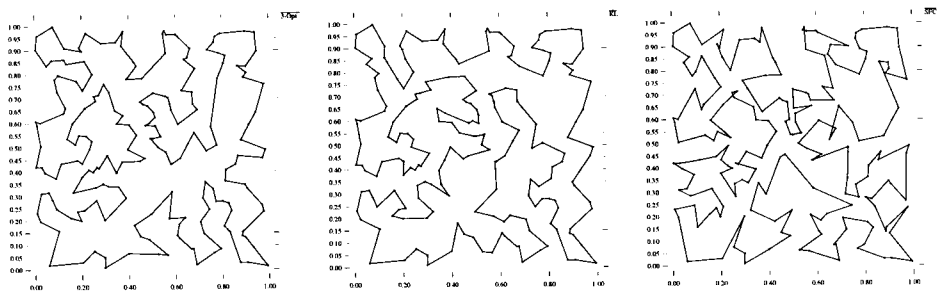


Figure 11. The Sierpinski spacefilling curve in the plane is the limit of a sequence of recursive constructions.



(a) (b) (c)

Figure 12. Tours generated by (a) 3-Opt (cost 11.307), (b) Lin-Kernighan (cost 11.224) and (c) SFC (cost 13.945) over 200 random points in the Euclidean unit square.

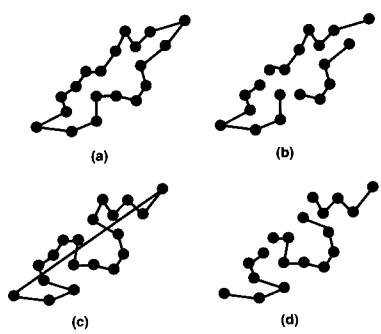


Figure 13. (a) a nonuniform geometric data set and a low-cost tour, (b) a partition into 3 clusters resulting from splitting this tour, and (c) a low-cost linear ordering (plus the diameter edge), and (d) a partition resulting from splitting this linear ordering.

Below, we will see that DP-RP yields its best results using the SFC ordering for $k = 2$ and $k = 4$, because of the structure of this ordering. However, as k increases the other orderings perform significantly better. We also experimented with applying 3-Opt post-processing to the SFC ordering, to see if the structure of an SFC tour could be preserved while minimizing cost. However, SFC + 3-Opt typically led to results similar to those of 3-Opt alone, suggesting that 3-Opt post-processing does not adequately preserve the structure of the original SFC tour.

Note that for some instances, neither a tour with low cost nor one with a predetermined structure will lead to success. Figure 13 illustrates such an instance: (a) gives a tour with low cost, but (b) shows that a partition of this tour into three clusters will not be very good. For this instance, which resembles a correlated data set, it is advantageous to construct a tour as in (c), with the diameter of the entire data set being an edge of the tour (i.e., a "good" linear ordering is constructed from one endpoint of the diameter to the other). Splitting this tour yields the solution P^3 shown in (d). Thus, for correlated instances, we propose to construct linear orderings via a modification of 3-Opt called 3-OptL, which fixes a diameter as an edge in the tour and then runs 3-Opt.

Many other linear ordering constructions may work just as well, if not better. For example, one could orthogonally project the points onto the line determined by least-squares fit or linear regression to induce an ordering. For a partition instance in which G is sparse, effective spectral approaches have used the ordering given by the second eigenvector of the Laplacian of G (see, for example Fiedler (1973) and Pothen, Simon, and Liou (1990)). Alternatively, Bartholdi and Platzman (1989) outlined a space-filling curve construction that can be specifically tuned to given classes of non-uniform data sets. The Sierpinski tour that we study is best suited for uniformly random data points.

6. Experimental Results

We compare the CL and DQ algorithms discussed in Section 2 against our DP-RP approach using each of the ordering constructions discussed in Section 5. We present results for both random and non-random data sets and conclude that DP-RP generally yields partitions with lower maximum diameter than previous approaches, although the best tour construction varies with n, k, and the nature of the instance.

TABLE 1

Diameters ($\times 10^4$) for 100 points (2-dimensional unit square). Average tour costs for 3-Opt, Lin-Kernighan, SFC and SFC + 3-Opt were 79.28, 78.25, 96.30 and 79.80 respectively.

#	CL	DQ	DP-RP			
Clusters			3-Opt	Lin-K	SFC	SFC+3-Opt
2	10408	**10024**	10041	10042	**10024**	10027
3	9425	9760	8547	8528	**8375**	8434
4	7075	6208	7053	6993	**6089**	6429
5	6328	5944	5897	5873	**5716**	5689
6	5747	5635	5316	5240	5355	**5218**
7	5267	5296	4868	4882	4986	**4814**
8	4843	4741	4498	4504	5350	**4409**
9	4503	4577	4160	4164	4140	**4037**
10	4182	4443	3864	3828	3925	**3801**

TABLE 2

Diameters ($\times 10^4$) for 100 points (3-dimensional cube). Average tour costs for 3-Opt, Lin-Kernighan, SFC and SFC + 3-Opt were 172.0, 170.4, 234.9 and 172.2 respectively.

#	CL	DQ	DP-RP			
Clusters			3-Opt	Lin-K	SFC	SFC+3-Opt
2	12851	**12176**	12518	12590	12455	12554
3	11697	11825	11458	11498	**11260**	11487
4	10651	10441	10401	10404	**10130**	10430
5	9799	10148	9679	9649	**9555**	9669
6	9049	9874	**8834**	8844	8959	8845
7	8160	9460	8144	8164	8218	**8052**
8	7477	8055	7590	7503	**6641**	7385
9	7017	7676	7090	6973	**6299**	6900
10	6671	7296	6607	6577	**6140**	6425

TABLE 3

Diameters ($\times 10^4$) for 400 points (2-dimensional square). Average tour costs for 3-Opt, Lin-Kernighan, SFC and SFC + 3-Opt were 152.8, 150.7, 191.7 and 154.9, respectively.

#	CL	DQ	DP-RP			
Clusters			3-Opt	Lin-K	SFC	SFC+3-Opt
2	11061	**10565**	10650	10629	**10565**	10572
3	10216	10414	9494	9534	**9274**	9321
4	7842	6607	7987	8366	**6570**	7038
5	7178	6441	6888	7017	**6351**	6482
6	6590	6318	6255	6275	6158	**6040**
7	6102	6145	5847	**5822**	5898	5658
8	5674	5152	5441	5456	**4731**	5095
9	5272	5064	5148	5163	**4645**	4825
10	4975	4982	4848	4885	**4566**	4587

6.1 Random Data Sets

We generated uniformly random instances of sizes 100 and 400 in the 2- and 3-dimensional unit cubes, using the Euclidean distance metric. Tables 1-4 give the maximum diameter values (multiplied by 10^4) averaged over 100 instances for $2 \leq k \leq 10$. The smallest diameter for each value of k is given in boldface. We make a number of observations.

- In comparing CL and DQ, Guénoche, Hansen, and Jaumard (1991) conclude that CL usually creates partitions with larger diameter. We confirm this behavior for small k and $d = 2$, but find that CL performs better for larger k. In addition, we observe the practical relevance of the worst-case example of Figure 2: DQ performs poorly when k is just smaller than a power of 2 (e.g., $k = 3$ and $k = 7$).
- DP-RP when used with any of the four ordering constructions performs better overall than either CL or DQ; the SFC ordering construction performs best. Like DQ, SFC seems to perform best when k is a power of 2 or slightly larger, and worst when k is just smaller than a power of 2. The phenomenon occurs because the structure of an SFC tour naturally breaks the plane or the cube into 4, 8 or 16, but has a more difficult time with, e.g., 3, 7 or 15 clusters. However, this behavior is not nearly as pronounced as with DQ.
- 3-Opt post-processing of SFC improves results on the instances for which SFC does badly, but also can make the SFC results worse. Whether the 3-Opt step is beneficial depends on the values of both n and k.

We note that random instances lack natural clusters, hence the above observations cannot be generalized to non-random types of data. Thus, we examine two non-random instances below.

6.2 Non-Random Data Sets

We present results for two data sets:

- The Fisher (1936) Iris data consist of four measurements on 150 flowers of three varieties of Iris. Results are given in Table 5 using Euclidean distance in $\Re^4$ as a dissimilarity measure. Results for this data set have previously been given by Delattre and Hansen (1980) and by Guénoche, Hansen, and Jaumard (1991).
- Average temperatures in Farenheit for January, April, July, and October for 88 cities in the United States were published in *The*

TABLE 4

Diameters ($\times 10^4$) for 400 points (3-dimensional cube).
Average tour costs for 3-Opt, Lin-Kernighan, SFC and
SFC + 3-Opt were 414.7, 409.6, 597.0 and 415.9, respectively.

#	CL	DQ	DP-RP			
Clusters			3-Opt	Lin-K	SFC	SFC+3-Opt
2	13878	**13233**	13807	13875	13399	13861
3	13094	13037	13188	13092	**12671**	13117
4	11771	11159	12338	12332	**10873**	12367
5	10952	10996	11533	11479	**10522**	11607
6	10505	10793	10921	10843	**10170**	10949
7	9940	10626	10431	10394	**9831**	10377
8	8837	8259	9971	9904	**7379**	9839
9	8324	7973	9464	9371	**7156**	9365
10	7993	7782	8964	8944	**6984**	8908

TABLE 5

Fisher data, 150 points, 4 dimensions.

#	CL	DQ	DP-RP			
Clusters			3-Opt	Lin-K	SFC	3-OptL
2	40.25	**38.24**	**38.24**	**38.24**	48.14	40.25
3	32.11	36.82	36.82	36.82	43.59	**27.44**
4	**24.29**	**24.29**	25.50	25.50	42.26	**24.29**
5	22.36	22.43	23.81	23.81	39.75	**20.62**
6	**17.06**	20.74	22.67	22.67	29.72	**17.06**
7	16.61	15.84	18.81	18.81	23.69	**15.62**
8	**14.63**	15.84	16.58	16.58	22.83	14.66
9	14.53	**13.89**	16.28	16.28	20.32	14.63
10	14.49	**13.82**	14.07	14.07	20.32	14.07

TABLE 6

Temperature data, 85 cities, 4 months.

#	CL	DQ	DP-RP			
Clusters			3-Opt	Lin-K	SFC	3-OptL
2	96.41	**76.37**	77.75	**76.37**	**76.37**	**76.37**
3	**50.59**	75.44	73.40	73.40	75.44	53.18
4	48.37	**40.93**	50.59	50.59	**40.93**	**40.93**
5	40.93	38.13	45.93	48.48	38.13	**36.63**
6	34.47	37.35	37.71	39.72	34.47	**31.97**
7	**28.09**	33.56	35.62	35.62	33.56	28.93
8	27.53	33.54	32.88	29.56	31.08	**26.40**
9	26.19	27.53	29.38	29.38	27.53	**24.15**
10	24.92	25.28	26.87	28.20	25.28	**22.32**

World Almanac (Hoffman 1992). Results are given in Table 6, again using Euclidean distance as a dissimilarity measure for this 4-month instance.

For these non-random data sets, SFC + 3-Opt gave performance equivalent to or worse than 3-Opt for each value of k, hence these results are not reported. We also include 3-OptL results for these data sets since they are both strongly correlated, as can be seen from the product-moment correlations for each pair of the four data attributes: for the Fisher data set, pairwise correlations are 0.978, 0.948, 0.898, 0.871, 0.809 and 0.984; for the Temperature data set, pairwise correlations are 0.959, 0.913, 0.954, 0.989, 0.998 and 0.992. We make the following observations:

- 3-OptL generally performs better than the other approaches, especially for small k. Hence, for correlated data sets and small k, a good linear ordering may be preferable to a low-cost tour. It remains to be seen whether other linear ordering constructions, e.g., eigenvector (principal components) or least-squares fit approaches, can lead to even better partitions.
- CL and DQ comparatively perform much better for non-random, as opposed to random, data sets.

7. Conclusion

We have discussed two previous and one new approach to constructing partitions for the min-diameter objective. Our new approach solves a *restricted partition* formulation that requires clusters to be contiguous with respect to a given tour or linear ordering. The approach is efficient for a large class of objectives, in addition to min-diameter.

We have used our approach in conjunction with a variety of ordering constructions. Our experimental results lead us to conclude that a spacefilling curve construction is best suited for uniformly random data and that a low-cost linear ordering — rather than a tour — is best suited for strongly correlated data. For other types of data sets, we believe that Lin-Kernighan tours, which have low cost, will generally yield good results. The best tour construction clearly depends on the specific nature of the data set, and explicitly identifying appropriate constructions remains an open direction for future work.

References

ALPERT, C. J., and KAHNG, A. B. (1993), "Multi-Way Netlist Partitioning Using Spacefilling Curves," Technical Report No. #930016, University of California at Los Angeles, Computer Science Department.

ASANO, T., BHATTACHARYA, B. K., KEIL, J. M., and YAO, F. F. (1988), "Clustering Algorithms Based on Minimum and Maximum Spanning Trees," *Proceedings of the Fourth Annual Symposium on Computational Geometry*, New York: ACM Press, 252-257.

AVIS, D. (1986), "Diameter Partitioning," *Discrete and Computational Geometry, 1*, 265-276.

BARTHOLDI, J. J., and PLATZMAN, L. K. (1988), "Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space," *Management Sciences, 34(3)*, 291-305.

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

BRUCKER, P. (1978), "On the Complexity of Clustering Problems," in *Optimization and Operations Research, Lecture Notes in Economics and Mathematical Systems, 157*, Eds., M. Beckman and H. P. Kunzi, Heidelberg: Springer, 45-54.

CHAN, P. K., SCHLAG, M. D. F., and ZIEN, J. (1994), "Spectral K-Way Ratio Cut Partitioning and Clustering," *IEEE Transactions on Computer-Aided Design, 13(9)*, 1088-1096.

DEFAYS, D. (1978), "A Short Note on a Method of Seriation," *British Journal of Mathematical and Statistical Psychology, 31*, 49-53.

DELATTRE, M., and HANSEN, P. (1980), "Bicriterion Cluster Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-2(4)*, 277-291.

FEDER, T., and GREENE, D. H. (1988), "Optimal Algorithms for Approximate Clustering," *Proceedings of the Twentieth Annual ACM Symposium on the Theory of Computing*, New York: ACM Press, 434-444.

FIEDLER, M. (1973), "Algebraic Connectivity of Graphs," *Czechoslovak Mathematical Journal, 23 (98)*, 298-305.

FISHER, R. A. (1936), "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics, VII(II)*, 179-188

GLASBEY, C. A. (1987), "Complete Linkage as a Multiple Stopping Rule for Single Linkage Clustering," *Journal of Classification, 4*, 103-109.

GONZALEZ, T. F. (1985), "Clustering to Minimize the Maximum Intercluster Distance," *Theoretical Computer Science, 38*, 293-306.

GUÉNOCHE, A. (1993), "Enumération des Partitions de Diamètre Minimum," *Discrete Mathematics, 111*, 277-287.

GUÉNOCHE, A., HANSEN, P., and JAUMARD, B. (1991), "Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion," *Journal of Classification, 8*, 5-30.

HALL, K. M. (1970), "An r-dimensional Quadratic Placement Algorithm," *Management Science, 17*, 219-229.

HANSEN, P., and DELATTRE, M. (1978), "Complete Link Cluster Analysis by Graph Coloring," *Journal of the American Statistical Association, 73*, 397-403.

HANSEN, P., and JAUMARD, B. (1987), "Minimum Sum of Diameters Clustering," *Journal of Classification, 4*, 215-226.

HANSEN, P., JAUMARD, B., and FRANK, O. (1989), "Maximum Sum-of-Splits Clustering," *Journal of Classification, 6*, 177-193.

HANSEN, P., JAUMARD, B., and MUSITU, K. (1990), "Weight Constrained Maximum Split Clustering," *Journal of Classification, 7*, 217-240.

HERSHBERGER, J. (1991), "Minimizing the Sum of Diameters Efficiently," *Proceedings of the Third Canadian Conference on Computational Geometry*, 62-65.

HOFFMAN, M. S. (1992), Ed., *The World Almanac*, New York: Pharos, 209, 292-293.

HUBERT, L. (1973), "Monotone Invariant Clustering Procedures," *Psychometrika, 38(1)*, 47-62.

HUBERT, L. J., and ARABIE, P. (1986), "Unidimensional Scaling and Combinatorial Optimization," in *Multidimensional Data Analysis*, Eds., J. De Leeuw, W. Heiser, J. Meulman, and F. Critchley, Leiden, The Netherlands: DSWO Press, 181-196.

JOHNSON, D. S. (1990), "Local Optimization and the Traveling Salesman Problem", *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, Ed., M. S. Paterson, Berlin: Springer-Verlag, 446-460.

JOHNSON, S. C. (1967), "Hierarchical Clustering Schemes," *Psychometrika, 32(3)*, 241-254.

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

LIN, S. (1965), "Computer Solutions of the Traveling Salesman Problem," *The Bell System Technical Journal, 44*, 2245-2269.

LIN, S., and KERNIGHAN, B. W. (1973), "An Effective Heuristic Algorithm for the Traveling Salesman Problem with Thousands of Nodes," *Operations Research, 21*, 498-516.

MONMA, C., and SURI, S. (1991), "Partitioning Points and Graphs to Minimize the Maximum or the Sum of Diameters," *Proceedings of the Sixth International Conference on the Theory and Applications of Graphs*, Eds., Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, New York: Wiley.

PAPADIMITRIOU, C. H. (1992), "The Complexity of the Lin-Kernighan Heuristic for the Traveling Salesman Problem", *SIAM Journal on Computing, 21(3)*, 450-465.

PLINER, V. (1986), "The Problem of Multidimensional Metric Scaling," *Automation and Remote Control, 47*, 560-567.

PLINER, V. (1996), "Metric Unidimensional Scaling and Global Optimization," *Journal of Classification, 13*, 3-18.

POTHEN, A., SIMON, H. D., AND LIOU, K.-P. (1990), "Partitioning Sparse Matrices with Eigenvectors of Graphs", *SIAM Journal on Matrix Analysis and Applications, 11(3)*, 430-452.

RAO, M. R. (1971), "Cluster Analysis and Mathematical Programming," *Journal of the American Statistical Association, 66(335)*, 622-626.

SUN, W., and SECHEN, C. (1993), "Efficient and Effective Placements for Very Large Circuits," *Proceedings of the IEEE International Conference on Computer-Aided Design*, Los Alamitos, California: IEEE Computer Society Press, 170-177.