Incremental Clustering and Dynamic Information Retrieval

MOSES CHARIKAR* CHANDRA CHEKURI† TOMÁS FEDER‡ RAJEEV MOTWANI§

Abstract

Motivated by applications such as document and image classification in information retrieval, we consider the problem of clustering *dynamic* point sets in a metric space. We propose a model called *incremental clustering* which is based on a careful analysis of the requirements of the information retrieval application, and which should also be useful in other applications. The goal is to efficiently maintain clusters of small diameter as new points are inserted. We analyze several natural greedy algorithms and demonstrate that they perform poorly. We propose new deterministic and randomized incremental clustering algorithms which have a provably good performance. We complement our positive results with lower bounds on the performance of incremental algorithms. Finally, we consider the dual clustering problem where the clusters are of fixed diameter, and the goal is to minimize the number of clusters.

1 Introduction

We consider the following problem: as a sequence of points from a metric space is presented, efficiently maintain a clustering of the points so as to minimize the maximum cluster diameter. Such problems arise in a variety of applications, in particular in document and image classification for information retrieval. We propose a model called *incremental clustering* based primarily on the requirements for the information retrieval application, although our model should also be relevant to other applications. We begin by analyzing several natural greedy algorithms and discover that they perform rather poorly in this setting. We then identify some new deterministic and randomized algorithms with provably good performance. We complement our positive results with lower bounds on the performance of incremental algorithms. We also consider the dual clustering problem where the clusters are of fixed diameter, and the goal is to minimize the number of clusters. Before describing our results in any greater detail, we motivate and formalize our new model.

Clustering is used for data analysis and classification in a wide variety of application [1, 12, 20, 27, 34]. It has proved to be a particularly important tool in *information retrieval* for constructing a taxonomy of a corpus of documents by forming groups of closely-related documents [13, 16, 34, 35, 37, 38]. For this purpose, a distance metric is imposed over documents, enabling us to view them as points in a metric space. The central role of clustering in this application is captured by the so-called **cluster hypothesis**: *documents relevant to a query tend to be more similar to each other than to irrelevant documents and hence are likely to be clustered together*. Typically, clustering is used to accelerate query processing by considering only a small number of representatives of the clusters, rather than the entire corpus. In addition, it is used for classification [11] and has been suggested as a method for facilitating browsing [9, 10].

The current information explosion, fueled by the availability of hypermedia and the World-wide Web, has led to the generation of an ever-increasing volume of data, posing a growing challenge for information retrieval systems to efficiently store and retrieve this information [40]. A major issue that document databases are now facing is the extremely high rate of update. Several practitioners have complained that existing clustering algorithms are not suitable for maintaining clusters in such a dynamic environment, and they have been struggling with the problem of updating clusters without frequently performing complete reclustering [4, 5, 6, 8, 35]. From a theoretical perspective, many different formulations are possible for this dynamic clustering problem, and it is not clear a priori which of these best addresses the concerns of the practitioners. After a careful study of the requirements, we propose the model described below.

*Department of Computer Science, Stanford University, Stanford, CA 94305-9045. E-mail: moses@cs.stanford.edu. Supported by NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

†Department of Computer Science, Stanford University, Stanford, CA 94305-9045. E-mail: chekuri@cs.stanford.edu. Supported by NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

‡E-mail: tomas@theory.stanford.edu

§Department of Computer Science, Stanford University, Stanford, CA 94305-9045. E-mail: rajeev@cs.stanford.edu. Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04-96-1-0007, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

Hierarchical Agglomerative Clustering. The clustering strategy employed almost universally in information retrieval is *Hierarchical Agglomerative Clustering (HAC)* [12, 34, 35, 37, 38, 39]. This is also popular in other applications such as biology, medicine, image processing, and geographical information systems. The basic idea is: initially assign the n input points to n distinct clusters; repeatedly merge pairs of clusters until their number is sufficiently small. Many instantiations have been proposed and implemented, differing mainly in the rule for deciding which clusters to merge at each step. Note that HAC computes hierarchy trees of clusters (also called *dendograms*) whose leaves are individual points and internal nodes correspond to clusters formed by merging clusters at their children. A key advantage of these trees is that they permit refinement of responses to queries by moving down the hierarchy. Typically, the internal nodes are labeled with indexing information (sometimes called *conceptual information*) used for processing queries and in associating semantics with clusters (e.g., for browsing). Experience shows that HAC performs extremely well both in terms of efficiency and cluster quality. In the dynamic setting, it is desirable to retain the hierarchical structure while ensuring efficient update and high-quality clustering. An important goal is to avoid any major modifications in the clustering while processing updates, since any extensive recomputation of the index information will swamp the cost of clustering itself. The input size in typical applications is such that super-quadratic time is impractical, and in fact it is desirable to obtain close to linear time.

A Model for Incremental Clustering. Various measures of distance between documents have been proposed in the literature, but we will not concern ourselves with the details thereof; for our purposes, it suffices to note that these distance measures induce a metric space. Since documents are usually represented as high-dimensional vectors, we cannot make any stronger assumption than that of an arbitrary metric space, although, as we will see, our results improve in geometric spaces.

Formally, the *clustering* problem is: given n points in a metric space $\mathcal{M}$, partition the points into k clusters so as to minimize the maximum cluster diameter. The *diameter* of a cluster is defined to be the maximum inter-point distance in it. Sometimes the objective function is chosen to be the maximum cluster radius. In Euclidean spaces, *radius* denotes the radius of the minimum ball enclosing all points in the cluster. To extend the notion of radius to arbitrary metric spaces, we first select a *center* point in each cluster, whereupon the radius is defined as the maximum distance from the center to any point in the cluster. We will assume the diameter measure as the default.

We define the *incremental clustering* problem as follows: for an update sequence of n points in $\mathcal{M}$, maintain a collection of k clusters such that as each input point is presented, either it is assigned to one of the current k clusters, or it starts off a new cluster while two existing clusters are merged into one. We define the *performance ratio* of an incremental clustering algorithm as the maximum over all update sequences of the ratio of its maximum cluster diameter (or, radius) to that of the optimal clustering for the input points.

Our model enforces the requirement that *at all times* an incremental algorithm should maintain a HAC for the points presented up to that time. As before, an algorithm is free to use any rule for choosing the two clusters to merge at each step. This model preserves all the desirable properties of HAC while providing a clean extension to the dynamic case. In addition, it has been observed that such incremental algorithms exhibit good paging performance when the clusters themselves are stored in secondary storage, while cluster representatives are preserved in main memory [32].

We have avoided labeling this model as *the online clustering problem* or referring to the performance ratio as a *competitive ratio* [25] for the following reasons. Recall that in an online setting, we would compare the performance of an algorithm to that of an adversary which knows the update sequence in advance but must process the points in the order of arrival. Our model has a stronger requirement for incremental algorithms, in that they are compared to adversaries which do not need to respect the input ordering, i.e., we compare our algorithms to optimal clusterings of the final point set, and no intermediate clusterings need be maintained. Also, online algorithms are permitted superpolynomial running times. In contrast, our model essentially requires polynomial-time approximation algorithms which are constrained to incrementally maintain a HAC. It may be interesting to explore the several different formulations of online clustering; for example, when the newly inserted point starts off a new cluster, we could allow the points of one old cluster to be redistributed among the remaining, rather than requiring that two clusters be merged together. The problem with such formulations is that they do not lead to HACs; moreover, they entail the recomputation of the index structures for *all clusters*, which renders the algorithms useless from the point of view of applications under consideration here.

Previous Work in Static Clustering. The closely-related problems of clustering to minimize diameter and radius are also called *pairwise clustering* and the *k-center problem*, respectively [2, 21]. Both are NP-hard [17, 28], and in fact hard to approximate to within factor 2 for arbitrary metric spaces [2, 21]. For Euclidean spaces, clustering on the line is easy [3], but in higher dimensions it is NP-hard to approximate to within factors close to 2, regardless of the metric used [14, 15, 19, 29, 30]. The *furthest point heuristic* due to Gonzalez [19] (see also Hochbaum and Shmoys [23, 24]) gives a 2-approximation in all metric spaces. This algorithm requires $O(kn)$ distance computations, and when the metric space is induced by shortest-path distances in weighted graphs, the running time is $O(n^2)$. Feder and Greene [14] gave an implementation for *Euclidean* spaces with running time $O(n \log k)$.

Overview of Results. Our results for incremental clustering show that it is possible to obtain algorithms that are comparable to the best possible in the static setting, both in terms of efficiency and performance ratio. We begin in Section 2 by considering natural greedy algorithms that choose clusters to merge based on some measure of the resulting cluster. We establish that greedy algorithms behave poorly by proving that a Center-Greedy algorithm has a tight performance ratio of $2k-1$, and a Diameter-Greedy algorithm has a lower bound of $\Omega(\log k)$. It seems likely that greedy algorithms behave better in geometric spaces, and we discover some evidence in the case of the line. We show that Diameter-Greedy has performance ratio 2 for $k = 2$ on the line. This analysis suggests a variant of Diameter-Greedy, and this is shown to achieve ratio 3 for all k on the line. In Section 3 we present the Doubling Algorithm and show that its performance ratio is 8, and that a randomized version has ratio 5.43. While the obvious implementation of these algorithms is expensive, we show that they can be implemented so as to achieve amortized time $O(k \log k)$ per update. These results for the Doubling Algorithm carry over to the radius measure. Then, in Section 4, we present the Clique Algorithm and show that it has performance ratio 6, and that a randomized version has ratio 4.33. While the Clique Algorithm may appear to dominate the Doubling Algorithm, this is not the case since the former requires computing clique partitions, an NP-hard problem, although it must be said in its defense that the clique partitions need only be computed in graphs with $k+1$ vertices. While the performance ratio of the Clique Algorithm is 8 for the radius measure, improved bounds are possible for d-dimensional Euclidean spaces; specifically, we show that the radius performance ratio of the Clique Algorithm in $\Re^d$ improves to $4(1+\sqrt{d/(2d+2)})$, which is 6 for $d=1$, and is asymptotic to 6.83 for large d. In Section 5, we provide lower bounds for incremental clustering algorithms. We show that even for $k = 2$ and on the line, no deterministic or randomized algorithm can achieve a ratio better than 2. We improve this lower bound to 2.414 for deterministic algorithms in general metric spaces. Finally, in Section 6 we consider the dual clustering problem of minimizing the number of clusters of a fixed radius. Since it is impossible to achieve bounded ratios for general metric spaces, we focus on d-dimensional Euclidean spaces. We present an incremental algorithm that has performance ratio $O(2^d d \log d)$, and also provide a lower bound of $\Omega(\log d / \log\log\log d)$.

Many interesting directions for future research are suggested by our work. There are the obvious questions of improving our upper and lower bounds, particularly for the dual clustering problem. An important theoretical question is whether the geometric setting permits better ratios than do metric spaces. Our model can be generalized in many different ways. Depending on the exact application, we may wish to consider other measures of clustering quality, such as: minimum variance in cluster diameter, and the sum of squares of the inter-point distances within a cluster. Then, there is the issue of handling deletions which, though not important for our motivating application of information retrieval, may be relevant in other applications. Finally, there is the question of formulating a model for *adaptive clustering*, wherein the clustering may be modified as a result of queries and user feedback, even without any updates.

2 Greedy Algorithms

We begin by examining some natural greedy algorithms. A *greedy incremental clustering algorithm* always merges clusters to minimize some fixed measure. Our results indicate that such algorithms perform poorly.

Definition 1 *The Center-Greedy Algorithm associates a center for each cluster and merges the two clusters whose centers are closest. The center of the old cluster with the larger radius becomes the new center. It is possible to define variants of Center-Greedy based on how the centers of the clusters are picked but we restrict ourselves to this definition for reasons of simplicity and intuitiveness.*

Definition 2 *The Diameter-Greedy Algorithm always merges those two clusters which minimize the diameter of the resulting merged cluster.*

We can establish the following lower bounds on the performance ratio of these two greedy algorithms. We omit the proofs in this extended abstract.

Theorem 1 *The Center-Greedy Algorithm has performance ratio at least* $2k-1$.

Theorem 2 *The Diameter-Greedy Algorithm has performance ratio at least* $\Omega(\log k)$*, even on the line.*

We now give a tight upper bound for the Center-Greedy Algorithm. Note that for $k=3$ it has ratio 5, but for larger k its performance is worse than that of the algorithms to be presented later.

Theorem 3 *The Center-Greedy Algorithm has performance ratio of* $2k-1$ *in any metric space.*

Proof: Suppose that a set P of n points is inserted. Let their optimal clustering be the partition $S = \{C_1, \ldots, C_k\}$, with d as the optimal diameter. We will show that the diameter of any cluster produced by Center-Greedy is at most $(2k-1)d$.

We define a graph G on the set S of the optimal clusters, where two clusters are connected by an edge if the minimum distance between them is at most d, where the distance between two clusters is the minimum distances between points in them. Consider the connected components of G. Note that two clusters in different connected components have minimum distance strictly greater than d. We say that a cluster X intersects a connected component consisting of the optimal clusters $C_{i_1}, \ldots, C_{i_r}$ if X intersects $\cup_{j=1}^{r} C_{i_j}$.

We claim that at all times, any cluster produced by Center-Greedy intersects exactly one connected component of G. We prove this claim by induction over n. Suppose the claim is true before a new point p arrives. Initially, p is in a cluster of its own and Center-Greedy has $k+1$ clusters, each of which intersect exactly one connected component of G. Since there are $k+1$ cluster centers, two of them must be in the same optimal cluster. This implies that the distance between the two closest centers is at most d. If X_1 and X_2 are the clusters that Center-Greedy merges at this stage, the centers of X_1 and X_2 must be at most d apart. Hence, both clusters' centers must lie in the same connected component of G, say $\mathcal{C}$. By the inductive hypothesis, all points in X_1 and X_2 must be in $\mathcal{C}$. Hence, all points in the new cluster $X_1 \cup X_2$ must lie in $\mathcal{C}$, establishing the inductive hypothesis.

Since each cluster produced by Center-Greedy lies in exactly one connected component of G, the diameter is bounded by the maximum diameter of a connected component, which is at most $(2k-1)d$. ■

For Diameter-Greedy in general metric spaces, we only have the following weak upper bound; the proof is deferred to the final version.

Theorem 4 *For $k=2$, the Diameter-Greedy Algorithm has a performance ratio 3 in any metric space.*

In spite of the lower bounds for greedy algorithms, they may not be entirely useless since some variant may perform well in geometric spaces. We obtain some positive evidence in this regard via the following analysis for the line. The upper bounds given here should be contrasted with the lower bound of 2 for the line shown in Section 5. The following definitions underlie the analysis.

Definition 3 *Given a set S of n points in the line, a t-*partition *subdivides the interval between the first and last points of S into t subintervals whose endpoints are in S. The t-*diameter *of S is the minimum over all t-partitions of the maximum interval length in a t-partition of S. The 1-diameter is the diameter, while the 2-diameter is the radius of S where the center is constrained to be a point of S.*

We define the following family of algorithms based on the notion of the t-diameter.

Definition 4 *The t-Diameter Greedy Algorithm merges those two clusters which minimize the t-diameter of the merged cluster. Note that 1-Diameter Greedy is the same as Diameter-Greedy.*

While Diameter-Greedy has ratio 2 for $k=2$ and ratio 3 for $k=3$, we can show a lower bound of $\Omega(\log k)$ on its performance ratio on the line.

Theorem 5 *The Diameter-Greedy Algorithm for the line has performance ratio 2 for $k=2$ and performance ratio 3 for $k=3$.*

Unlike Diameter-Greedy, we can show that 3-Diameter Greedy has a bounded performance ratio on the line.

Theorem 6 *The 3-Diameter Greedy Algorithm has performance ratio 3 on the line.*

Proof: In fact, we show that it produces a clustering with 3-diameter at most the optimal diameter, and the factor of 3 follows. Assume this holds before the last two clusters are merged. Let $I_1, I_2, \ldots, I_k$ be the intervals in the optimal clustering, with maximum diameter d. Let $C_1, C_2, \ldots, C_{k+1}$ be the current clusters, each with 3-diameter at most d, of which two must be merged. If C_i starts in I_a and ends in I_b, let $x_i = b-a$; notice that $x_1 + \cdots + x_{k+1} \leq k-1$. We assume that if C_i ends in I_b then C_{i+1} starts in I_b; otherwise, we could replace the argument in the k intervals I_j by an argument either in the first b intervals $I_1, \ldots, I_b$, if there are at least $b+1$ clusters C_i in this region, or in the last $k-b$ intervals $I_{b+1}, \ldots, I_k$, if there are at least $k-b+1$ current clusters C_i in this region. Now, the bounds imply that for some i, we have $x_i + x_{i+1} < 2$. If $x_i = x_{i+1} = 0$, then the merging of C_i and C_{i+1} is contained in a single interval I_j and has diameter at most d. If say $x_i = 0$ and $x_{i+1} = 1$, then the gap G between the two consecutive intervals I_j and I_{j+1} involved is at most d, since C_{i+1} has 3-diameter at most d, so the merger of C_i and C_{i+1} has 3-diameter at most d given by the 3-partition I_j, G, I_{j+1}. This completes the proof. ■

We comment briefly on the running time of this algorithm. In the above proof, the 3-diameter of an interval may be replaced by an easily-computed upper bound: at the time of creation of interval $[a, b]$, let $[x, y]$ be the gap containing $(a+b)/2$, and let the upper bound be $\max(x-a, y-x, b-y)$. Maintaining the n points sorted in a balanced tree, the running time is $O(\log n)$ for each of the n points inserted.

3 The Doubling Algorithm

We now describe the Doubling Algorithm which has performance ratio 8 for incremental clustering in general metric spaces. The algorithm works in phases and uses two parameters α and β to be specified later, such that $\alpha/(\alpha-1) \leq \beta$. At the start of phase i, it has a collection of $k+1$ clusters $C_1, C_2, \ldots, C_{k+1}$ and a lower bound d_i on the optimal clustering's diameter (denoted by OPT). Each cluster C_i has a center c_i which is one of the points of the cluster. The following invariants are assumed at the start of phase i: **(a)** for each cluster C_j, the radius of C_j defined as $\max_{p \in C_j} d(c_j, p)$ is at most αd_i; **(b)** for each pair of clusters C_j and C_l, the inter-center distance $d(c_j, c_l) \geq d_i$; and, **(c)** $d_i \leq$ OPT.

Each phase consists of two stages: the first is a *merging stage* in which the algorithm reduces the number of clusters by merging certain pairs; the second is the *update stage* in which the algorithm accepts new updates and tries to maintain at most k clusters without increasing the radius of the clusters or violating the invariants (clearly, it can always do so by making the new points into separate clusters). A phase ends when the number of clusters again exceeds k.

Definition 5 *The* t-threshold graph *on a set of points* $P = \{p_1, p_2, \ldots, p_n\}$ *is the graph* $G = (P, E)$ *such that* $(p_i, p_j) \in E$ *if and only if* $d(p_i, p_j) \leq t$.

The merging stage works as follows. Define $d_{i+1} = \beta d_i$, and let G be the d_{i+1}-threshold graph on the $k+1$ cluster centers $c_1, c_2, \ldots, c_{k+1}$. The graph G is used to merge clusters by repeatedly performing the following steps while the graph is non-empty: pick an arbitrary cluster C_i in G and merge all its neighbors into it; make c_i the new cluster's center; and, remove C_i and its neighbors from G. Let $C'_1, C'_2, \ldots, C'_m$ be the new clusters at the end of the merging stage. Note that it is possible that $m = k + 1$ when the graph G has no edges, in which case the algorithm will be forced declare the end of phase i without going through the update stage.

Lemma 1 *The pairwise distance between cluster centers after the merging stage of phase* i *is at least* d_{i+1}.

Lemma 2 *The radius of the clusters after the merging stage of phase* i *is at most* $d_{i+1} + \alpha d_i \leq \alpha d_{i+1}$.

Proof: Prior to the merging, the distance between two clusters which are adjacent in the threshold graph is at most d_{i+1}, and their radius is at most αd_i. Therefore, the radius of the merged clusters is at most

$$d_{i+1} + \alpha d_i \leq (1 + \alpha/\beta) d_{i+1} \leq \alpha d_{i+1},$$

where the last inequality follows from the choice that $\alpha/(\alpha - 1) \leq \beta$. ■

The update stage continues while the number of clusters is at most k. When a new point arrives, the algorithm attempts to place it in one of the current clusters without exceeding the radius bound αd_{i+1}: otherwise, a new cluster is formed with the update as the cluster center. When the number of clusters reaches $k+1$, phase i ends and the current set of $k+1$ clusters along with d_{i+1} are used as the input for the $(i+1)$st phase.

All that remains to be specified about the algorithm is the initialization. The algorithm waits until $k+1$ points have arrived and then enters phase 1 with each point as the center of a cluster containing just itself, and with d_1 set to the distance between the closest pair of points. It is easily verified that the invariants hold at the start of phase 1. The following lemma shows that the clusters at the end of the ith phase satisfy the invariants for the $(i+1)$st phase.

Lemma 3 *The* $k+1$ *clusters at the end of the* i*th phase satisfy the following conditions:*

1. *The radius of the clusters is at most* αd_{i+1}.
2. *The pairwise distance between the cluster centers is at least* d_{i+1}.
3. $d_{i+1} \leq \text{OPT}$, *where* OPT *is the diameter of the optimal clustering for the current set of points.*

Proof: We have $k+1$ clusters at the end of the phase since that is the terminating condition. From Lemma 2, the radius of the clusters after the merging stage is at most αd_{i+1} and from the description of the update stage this bound is not violated by the insertion of new points. The distance between the clusters centers after the merging stage is d_{i+1}, and a new cluster is created only if a request point is at least d_{i+1} away from all current cluster centers. Therefore, the cluster centers have pairwise distance at least d_{i+1}. Since at the end of the phase we have $k+1$ cluster centers that are d_{i+1} apart, the optimal clustering is forced to put at least two of them in the same cluster. It follows that $\text{OPT} \geq d_{i+1}$. ■

Based on these lemmas, we make the following observations. The algorithm ensures the invariant that $d_i \leq \text{OPT}$ at the start of phase i. The radius of the clusters during phase i is at most αd_{i+1}. Therefore, the performance ratio at any time during the phase is at most $2\alpha d_{i+1}/\text{OPT} \leq 2\alpha\beta d_i/\text{OPT} \leq 2\alpha\beta$. We choose $\alpha = \beta = 2$; note, this choice satisfies the condition that $\alpha/(\alpha - 1) \leq \beta$. This leads to the following performance bound, which can be shown to be tight.

Theorem 7 *The Doubling Algorithm has performance ratio 8 in any metric space, and this is tight.*

An examination of the proof of the preceding theorem shows that the radius of the clusters is within factor 4 of the diameter of the optimal clustering, leading to the following result.

Corollary 1 *The Doubling Algorithm has performance ratio* 8 *for the radius measure.*

A simple modification of the Doubling Algorithm, in which we pick the new cluster centers by a simple left-to-right scan, improves the ratio to 6 for the case of the line.

While the obvious implementation of this algorithm appears to be inefficient, we can establish the following time bound, which is close to the best possible.

Theorem 8 *The Doubling Algorithm can be implemented to run in* $O(k \log k)$ *amortized time per update.*

Proof: First of all, we assume that there is a black-box for computing the distance between two points in the metric space in unit time. This is a reasonable assumption in most applications, and in any case even the static algorithms' analysis requires such an assumption. In the information retrieval application, the documents are represented as vectors and the black-box implementation will depend on the vector length as well as the exact definition of distance.

We now show how the Doubling Algorithm may be implemented so that the amortized time required for processing each new update is bounded by $O(k \log k)$. We maintain the edge lengths of the complete graph induced by the current cluster centers in a heap. Since there at most k clusters the space requirement is $O(k^2)$. When a new point arrives, we compute the distance of this point to the each of the current cluster centers, which requires $O(k)$ time. If the point is added to one of the current clusters, we are done. If, on the other hand the new point initiates a new cluster, we insert into the heap edges labeled with the distances between this new center and the existing cluster centers which takes $O(k \log k)$ time. For accounting purposes in the amortized analysis, we associate $\log k$ credits with each inserted edge. We will show that it is possible to charge the cost of implementing the merging stage of the algorithm to the credits associated with the edges. This implies the desired time bound.

We can assume, without loss of generality, that the merging stage merges at least two clusters. Let t be the threshold used during the phase. The algorithm extracts all the edges from the heap which have length less than t. Let m be the number of edges deleted from the heap. The deletion from the heap costs $O(m \log k)$ time. The t-threshhold graph on the cluster centers is exactly the graph induced by these m edges. It is easy to see that the procedure described to find the new cluster centers using the threshold graph takes time linear in the number of edges of the graph, assuming that the edges are given in the form of an adjacency list. Forming the adjacency list from the edges takes linear time. Therefore, the total cost of the merging phase is bounded by $O(m \log k + m) = O(m \log k)$ time. The credit of $\log k$ placed with each edge when it is inserted in to the heap accounts for this cost, completing the proof. ■

Finally, we describe a Randomized Doubling Algorithm with significantly better performance ratio. The algorithm is essentially the same as before, the main change being in the value of d_1 which is the lower bound for phase 1. In the deterministic case we chose d_1 to be the minimum pairwise distance of the first $k+1$ points, say x. We now choose a random value r from $[1/e, 1]$ according to the probability density function $1/r$, set d_1 to rx, and redefine $\beta = e$ and $\alpha = e/(e-1)$. Similar randomization of doubling algorithms was used earlier in scheduling [31], and later in other applications [7, 18].

Theorem 9 *The Randomized Doubling Algorithm has expected performance ratio $2e \approx 5.437$ in any metric space. The same bound is also achieved for the radius measure.*

Proof: Let σ be the sequence of updates and let the optimal cluster diameter for σ be γx, where x is the minimum pairwise distance of the first $k+1$ points. The optimal value is at least x, hence $\gamma \geq 1$. Suppose we choose $d_1 = rx$ for some $r \in (1/e, 1]$. Let ρ_r be the maximum radius of the clusters created for σ with this value of r. Using arguments similar to those in the proof of Theorem 7, we can show that ρ_r is at most $d_{i+1} + \alpha d_i = e^{i+1} d_1/(e-1)$, where i is the largest integer such that $d_i = e^{i-1} d_1 = e^{i-1} rx \leq \text{OPT} = \gamma x$. Let i^* be the integer such that $e^{i^*-1} \leq \gamma < e^{i^*}$ and $\delta = \gamma / e^{i^*}$. Then, $\rho_r \leq \frac{rex\gamma}{(e-1)\delta}$ when $r > \delta$, and $\rho_r \leq \frac{re^2 x\gamma}{(e-1)\delta}$ when $r \leq \delta$. Let X_r^- and X_r^+ be indicator variables for the the events $[r \leq \delta]$ and $[r > \delta]$, respectively. We claim that the expected value of ρ_r is bounded by

$$\begin{aligned} E[\rho_r] &\leq \int_{1/e}^{1} \frac{re\gamma x(eX_r^- + X_r^+)}{\delta r(e-1)} \mathrm{d}r \\ &= \frac{e\text{OPT}}{\delta(e-1)} \int_{1/e}^{1} (eX_r^- + X_r^+) \mathrm{d}r \\ &= \frac{e\text{OPT}\delta(e-1)}{\delta(e-1)} = e\text{OPT}. \end{aligned}$$

Therefore, the expected diameter is at most $2e\text{OPT}$. ■

4 The Clique Partition Algorithm

We now describe the Clique Algorithm which has performance ratio 6. This does not totally improve upon the Doubling Algorithm since the new algorithm involves solving the NP-hard clique partition problem, even though it is only on a graph with $k+1$ vertices. Finding a minimum clique partition is NP-hard even for graphs induced by points in the Euclidean plane [17], although it is in polynomial time for points on the line. Since the algorithm needs to solve the clique partition problem on graphs with $k+1$ vertices, this may not be too inefficient for small k.

Definition 6 *Given an undirected unweighted graph $G = (V, E)$, an l-clique partition is a partition of $V = V_1 \cup V_2 \cup \ldots \cup V_l$ such that the the induced graphs $G[V_i]$'s are cliques. A* minimum clique partition *is an l-clique partition with the minimum possible value of l.*

The Clique Algorithm is similar to the Doubling algorithm in that it also operates in phases which have a merging stage followed by an update stage. The invariants maintained by the algorithm are different though. At the start of the ith phase we have $k+1$ clusters $C_1, C_2, \ldots, C_{k+1}$ and a value d_i such that: **(a)** the radius of each cluster C_j is at most $2d_i$; the diameter of each cluster C_j is at most $3d_i$; and, **(c)** $d_i \leq \text{OPT}$.

The merging works as follows. Let $d_{i+1} = 2d_i$. We form the minimum clique partition of the d_{i+1}-threshold graph G of the cluster centers. The new clusters are then formed by merging the clusters in each clique of the clique partition. We arbitrarily choose one cluster from each clique and make its center the cluster center of the new merged cluster. Let $C'_1, C'_2, \ldots, C'_{l_i}$ be the resulting clusters. In the rest of the phase we also need to know which old clusters merged to form each of the new clusters.

Lemma 4 *The radius of the clusters after the merging stage is at most $2d_{i+1}$ and the diameter is at most $3d_{i+1}$.*

Proof: Let $C_{j_1}, C_{j_2}, \ldots, C_{j_{n_j}}$ be the clusters whose union is the new cluster C'_j and without loss of generality assume that the center of C_{j_1} was chosen to be the center of C'_j. Since the centers of $C_{j_1}, C_{j_2}, \ldots, C_{j_{n_j}}$ induce a clique in the d_{i+1}-threshold graph, the distance from the new center to each of the old cluster centers is at most d_{i+1}. The radius of each of $C_{j_1}, C_{j_2}, \ldots, C_{j_{n_j}}$ is at most $2d_i$. Therefore it follows that the new radius is at most $d_{i+1} + 2d_i \le 2d_{i+1}$ and the diameter is at most $2d_i + d_{i+1} + 2d_i \le 3d_{i+1}$. ■

During the update phase, a new point p is handled as follows. Let the current number of clusters be m, where $l_i \le m \le k$. Recall that $C'_1, C'_2, \ldots, C'_{l_i}$ are the clusters formed during the merging stage. If there exists j such that $j > l_i$ and $d(p, c'_j) \le d_{i+1}$, or if $j \le l_i$ and $d(p, c_{j_s}) \le d_{i+1}$ where C_{j_s} is a cluster which merged to form C'_j, add p to the cluster C'_j. If no such j exists, make a new cluster with p as the center. The phase ends when the number of clusters exceeds k, or if there are $k+1$ clusters at the end of the merging phase.

The intuition behind the new algorithm is the following. At the beginning of the phase we have $k+1$ clusters and a lower bound on the optimal. We use the lower bound to increase the radius of our existing clusters and merge some of them. To maintain the invariant for the lower bound in the next phase we need to ensure during this merging that the number of clusters we have after the merging is no more that what the optimal algorithm can achieve using the lower bound for the next phase. The doubling algorithm achieved this by picking an independent set as the new cluster centers in the distance threshold graph. The weakness of this approach is that we have a bound on the diameter, only as a function of the radius of the new cluster. We get the improvement by observing that a better bound on the number of clusters achievable by the optimal with diameter bounded by d_i is the size of the minimum clique partition of the distance threshold graph. We still need a condition on the radius in order to do the doubling, but now, since we use cliques, we can bound the diameter of the new clusters better than twice the radius.

The following lemmas show that the clusters at the end of phase i satisfy the invariants for phase $i+1$.

Lemma 5 *The radius of the clusters at the end of the phase i is at most $2d_{i+1}$ and the diameter of the clusters is at most $3d_{i+1}$.*

Lemma 6 *At the end of phase i, $d_{i+1} \le \text{OPT}$.*

Proof: Suppose $d_{i+1} > \text{OPT}$. Let $S = \{c_1, c_2, \ldots, c_{k+1}\}$ be the cluster centers at the beginning of the phase. Note that the centers $c'_1, \ldots, c'_{l_i}$ belong to S. Let $S' = \{c'_j \mid j > l_i\}$ be the set of cluster centers of the clusters which are formed in phase i after the merging stage. Since each of the centers c'_j in S' started a new cluster $d(c'_j, p) > d_{i+1}$ for all $p \in S \cup S' - \{c'_j\}$. Therefore in the optimal solution each center in S' is in a cluster which contains no center in S. This implies that the centers in S are contained in at most $l_i - 1$ clusters of diameter d_{i+1}. This is a contradiction since l_i was the size of the minimum clique partition of the d_{i+1}-threshold graph on S. ■

The diameter of the clusters during phase i is at most $3d_{i+1}$ and we maintain the invariant that $d_i \le \text{OPT}$ at the start of the phase. Therefore, the performance ratio of this algorithm is at most $3d_{i+1}/d_i \le 6$.

Theorem 10 *The Clique Algorithm has performance ratio 6 in any metric space, and this is tight.*

Since the radius of the clusters is within 4 of the optimal diameter, we obtain the following corollary.

Corollary 2 *The Clique Algorithm has performance ratio 8 in any metric space for the radius measure.*

As in the case of the Doubling Algorithm, we can use randomization to improve the bound. Let x be the minimum distance among the first $k+1$ points. The randomized algorithm sets $d_1 = rx$ in phase 1 of the deterministic algorithm, where r is chosen from $[1/2, 1]$ according to the probability density function $\frac{1}{r \ln 2}$. The analysis is similar to that of Theorem 9 and we omit the details.

Theorem 11 *The Randomized Clique Algorithm has performance ratio $\frac{3}{\ln 2} \approx 4.33$ in any metric space.*

Corollary 3 *The Randomized Clique Algorithm has performance ratio $\frac{4}{\ln 2} \approx 5.77$ for the radius measure in any metric space.*

The special structure of the clusters in the Clique Algorithm can be used to show that the performance ratio for the radius measure is better than 8 for the geometric case. This is based on the following result in geometry; we defer the proofs of the proposition and its consequence.

Proposition 12 *Any convex set in R^d of diameter at most 1 can be circumscribed by a sphere of radius r_d, where r_d satisfies the following recurrence with the base case $r_1 = 1/2$,*

$$r_d = \frac{1}{2\sqrt{1 - r_{d-1}^2}}.$$

The solution to this recurrence is $r_d = \sqrt{d/(2d+2)}$.

Theorem 13 *The Clique Algorithm has performance ratio $4(1+r_d)$ for the* radius *measure in R^d. This implies performance ratio 6 for $d=1$, 6.3 for $d=2$, and 6.83 asymptotically for large d.*

5 Lower Bounds

We present some lower bounds on the performance of incremental clustering. The lower bounds apply to both diameter and radius measures but our proofs are given for the diameter case. The following theorem shows that even for the simplest geometric space, we cannot expect a ratio better than 2; the proof is omitted.

Theorem 14 *For $k = 2$, there is a lower bound of 2 and $2 - 1/2^{k/2}$ on the performance ratio of deterministic and randomized algorithms, respectively, for incremental clustering on the line.*

In the case of general metric spaces, we can establish a stronger lower bound.

Theorem 15 *There is a lower bound of* $1+\sqrt{2} \approx 2.414$ *on the performance ratio of any deterministic incremental clustering algorithm for arbitrary metric spaces.*

Proof: Consider a metric space consisting of the points p_{ij}, $1 \le i,j \le 4$, $i \ne j$. The distances between the points are the shortest path distances in the graph with the following distances specified: $d(p_{ij}, p_{ji}) = 1$, and $d(p_{ij_1}, p_{ij_2}) = \sqrt{2}$. Let $B_i = \{p_{ij} \mid 1 \le j \le 4, i \ne j\}$. Note that the sets B_i, $1 \le i \le 4$, partition the metric space into 4 clusters of diameter $\sqrt{2}$. Let A be any deterministic algorithm for the incremental clustering problem. Let $k = 6$. Consider the clusters produced by A after it is given the 12 points p_{ij} described above.

Case 1: Suppose the maximum diameter of A's clusters is 1. Then A's clusters must be the 6 sets $\{p_{ij}, p_{ji}\}$. Now the adversary gives a point q such that $d(q, p_{ij}) = 10$ (any large number will do) for $1 \le i,j \le 4$. The optimal clustering is $\{q\}$ and the sets B_1, B_2, B_3, B_4. The optimal diameter is $\sqrt{2}$. We claim that the maximum diameter of A is at least $2+\sqrt{2}$. If the cluster that contains q contains any other point then our claim is clearly true. If on the other hand, the cluster that contains q does not contain any other point, A must have merged two of its existing clusters. Then the maximum diameter of A's resulting clusters is at least $2+\sqrt{2}$. Thus the performance ratio of A is at least $1+\sqrt{2}$.

Case 2: Suppose the maximum diameter of A's clusters is greater than 1. Then some cluster of A contains 2 points which are at least distance $\sqrt{2}$ apart. Let these points be p_{wx} and p_{yz}, $(w,x) \ne (z,y)$. Now the adversary gives 6 points r_{ij}, $1 \le i < j \le 4$ such that $d(r_{ij}, p_{ij}) = d(r_{ij}, p_{ji}) = 1$. The optimal clustering consists of the 6 sets $\{r_{ij}, p_{ij}, p_{ji}\}$. The optimal diameter is 1. We claim that the maximum diameter of A's clusters must be at least $1+\sqrt{2}$. Note that $d(r_{i_1j_1}, p_{i_2j_2}) \ge 1+\sqrt{2}$ for $(i_2,j_2) \ne (i_1,j_1)$, $(i_2,j_2) \ne (j_1,i_1)$. Also $d(r_{i_1j_1}, r_{i_2j_2}) \ge 2+\sqrt{2}$ for $(i_1,j_1) \ne (i_2,j_2)$. If A puts any two r_{ij} in the same cluster, our claim is clearly true. If all the r_{ij} are in separate clusters, each of the 6 clusters must contain one of them. Also one of the 6 clusters, say C must contain both the points p_{wx} and p_{yz}. Then C must have diameter at least $1+\sqrt{2}$, since the r_{ij} in C must be at distance at least $1+\sqrt{2}$ from one of p_{wx} and p_{yz}. Hence the performance ratio of A is at least $1+\sqrt{2}$. ∎

Finally, we can improve the randomized lower bound slightly for the case of arbitrary metric spaces.

Theorem 16 *For any* $\epsilon > 0$ *and* $k \ge 2$*, there is a lower bound of* $2-\epsilon$ *on the performance ratio of any randomized incremental algorithm.*

Proof: We use Yao's technique and show a lower bound on deterministic algorithms on an appropriately chosen distribution. Let A be a deterministic algorithm for incremental clustering. The distribution on inputs is as follows. Initially, the adversary provides n points $P_1, P_2 \ldots P_n$ such that the distance between any two of them is 1. Then the adversary partitions the n points into k disjoint sets $S_1, S_2 \ldots S_k$ at random, such that all partitions are equally likely. Finally the adversary provides k points $Q_1, Q_2, \ldots Q_k$, such that $d(Q_i, P_j) = 1$ if $P_j \in S_i$, $d(Q_i, P_j) = 2$ if $P_j \notin S_i$, $d(Q_i, Q_j) = 3$. Now, the diameter of the optimal solution for any input in the distribution is 1, obtained by constructing the k clusters $S_i \cup \{Q_i\}$. However, the incremental algorithm can produce a clustering with diameter 1 only if the clusters it produces after it sees points $P_1, P_2 \ldots P_n$ are precisely the sets S_i (selected at random by the adversary). Let $X_k(n)$ be the number of ways to partition the n points into k sets. Then the probability that the incremental algorithm produces a clustering of diameter 1 is at most $p = 1/X_k(n)$. With probability at least $1-p$, the incremental algorithm produces a clustering of diameter at least 2. Thus the expected value of the diameter of the clustering produced is at least $2-p$. Hence the expected value of the performance ratio is at least $2-p$. By chosing n suitably large, $X_k(n)$ can be made arbitrarily large, and hence p can be made arbitrarily small, in particular smaller than ϵ for any fixed $\epsilon > 0$. ∎

6 Dual Clustering

We now consider the dual clustering problem: for a sequence of points $p_1, p_2, \ldots, p_n \in \Re^d$, cover each point with a unit ball in $\Re^d$ as it arrives, so as to minimize the total number of balls used. In the static case this problem is *NP-Complete* and there is a PTAS for any fixed dimension [22]. We note that in general metric spaces, it is not possible to achieve any bounded ratio.

Our algorithm's analysis is based on a theorem from combinatorial geometry called Roger's theorem [36] (see also Theorem 7.17 [33]), which says that R^d can be covered by any convex shape with covering density $O(d \log d)$. Since the volume of a radius 2 ball is 2^d times the volume of a unit-radius ball, the number of balls needed to cover a ball of radius 2 using balls of unit radius is $f(d) = O(2^d d \log d)$. We first describe an incremental algorithm which has performance ratio $f(d)$. We also establish an asymptotic lower bound of $\Omega(\frac{\log d}{\log\log\log d})$; for $d = 1$ and 2, our proof yields lower bounds of 2 and 4, respectively.

Theorem 17 *For the dual clustering problem in* $\Re^d$*, there is an incremental algorithm with performance ratio* $O(2^d d \log d)$.

Proof: Our algorithm maintains a set $\mathcal{C}$ of *centers* which is a subset of the points that have arrived so far; initially, $\mathcal{C} = \emptyset$. Define the *range* $R(p)$ of a center p to be the sphere of radius 2 about p. For any two centers p_1 and p_2, we ensure that $d(p_1, p_2) > 2$. Associated with each center p is a set of points $\Gamma(p)$ called the *neighbors* of p. For convenience, we assume that $p \in \Gamma(p)$. We ensure that all neighbors of p lie in $R(p)$. When a new point x is received, if $x \in R(p)$ for some center p, we add x to $\Gamma(p)$, breaking ties arbitrarily. If no such center exists, x must be at a distance greater than 2 from all the existing centers. In this case, we make x a new center and set $\Gamma(x) = \{x\}$.

From Roger's theorem on packing density a sphere of radius 2 in $\Re^d$ can be covered by $f(d) = O(2^d d \log d)$ spheres of radius 1. When a new center p is created, we fix a set of spheres $S_1(p), S_2(p), \ldots, S_{f(d)}(p)$ which cover $R(p)$. Whenever a point x is added to $\Gamma(p)$, if it is not already covered by some previously placed sphere, we add the sphere $S_r(p)$ where r is any value such that $x \in S_r(p)$. Note that such a sphere must exist as $x \in R(p)$ and the spheres $S_1(p), S_2(p), \ldots, S_{f(d)}(p)$ cover $R(p)$ completely.

Since no two centers can be covered by a sphere of unit radius, any solution must use a separate sphere to cover each center. Hence, the number of centers is a lower bound for the number of spheres used by the optimal offline algorithm. For each center p, the incremental algorithm uses at most $f(d)$ spheres to cover the points in $\Gamma(p)$. Hence, the performance ratio of the incremental algorithm is bounded by $f(d) = O(2^d d \log d)$. ■

The following theorem gives a lower bound for the dual clustering problem.

Theorem 18 *For the dual clustering problem in $\Re^d$, any incremental algorithm must have performance ratio $\Omega(\frac{\log d}{\log \log \log d})$.*

Proof: The idea is as follows. At time t, when t points have been given by the adversary, it will be the case that the points $p_1, \ldots, p_t$ can be covered by a ball of radius $R_t < 1$. Then, the adversary will find a point p_{t+1} lying outside the t unit balls laid down by the algorithm so as to minimize the radius R_{t+1} of the ball required to cover all $t+1$ points and present that as a request. The game terminates when at some time $k+1$, we have for the first time that $R_{k+1} > 1$. Clearly, k is a lower bound on the performance ratio since the points $p_1, \ldots, p_k$ can be covered by a ball of radius $R_k \leq 1$, and the algorithm has used k balls up to that point. It remains to analyze the worst-case growth rate of R_t as a function of t. Note that $R_1 = 0$ and $R_2 = 1/2$.

Let α_d denote the volume of a unit ball in $\Re^d$. At time t, let D_t be any ball of radius (at most) R_t that covers the points $p_1, \ldots, p_t$. For some δ_t to be specified later, define the ball D_t^* as a ball with the same center as D_t and with radius $R_t + \delta_t$. We will choose δ_t such that the volume of D_t^* is at least $t\alpha_d$, implying that the current t unit balls placed by the algorithm cannot cover the entire volume of D_t^*. This would imply that there is a choice of a point p_{t+1} inside D_t^* which is not covered by the current t balls. It is also clear that the new set of $t+1$ points now can be covered by a ball of radius at most $R_t + \delta_t/2$. implying that

$$R_{t+1} = R_t + \frac{\delta_t}{2}.$$

To determine the value of δ_t is easy, since we have the inequality that

$$\alpha_d (R_t + \delta_t)^d > t\alpha_d$$

from the requirement that the ball D_t have volume equal to that of t unit balls. Now let $R_t = 1 - \epsilon_t$. Substituting in the above equations we obtain that:

$$\begin{array}{rll} & \delta_t & = 2(\epsilon_t - \epsilon_{t+1}) \\ \Rightarrow & R_t + \delta_t & = 1 + \epsilon_t - 2\epsilon_{t+1} \\ \Rightarrow & \alpha_d(1 + \epsilon_t - 2\epsilon_{t+1})^d & > t\alpha_d \\ \Rightarrow & \ln(1 + \epsilon_t - 2\epsilon_{t+1}) & > \dfrac{\ln t}{d} \end{array}$$

Note that $\epsilon_t - 2\epsilon_{t+1} < 1$. Using the fact that $\ln(1+x) > x/2$ for $x < 1$, we see that choosing ϵ_i such that

$$\frac{\epsilon_t - 2\epsilon_{t+1}}{2} = \frac{\ln t}{d}$$

will satisfy our requirements. Unfolding the recurrence,

$$\frac{\epsilon_1}{2^t} - \epsilon_{t+1} = \sum_{i=1}^{t} \frac{\ln i}{d 2^{t-i}} = \sum_{i=1}^{t} \frac{\ln i}{d 2^{t-i}} \leq \sum_{i=1}^{t} \frac{\ln t}{d 2^{t-i}} \leq \frac{2 \ln t}{d}$$

Noting that $\epsilon_1 = 1$, we obtain that $\epsilon_{t+1} \geq 2^{-t} - 2d^{-1} \ln t$. The lower bound is the smallest value of t for which ϵ_{t+1} is negative. Let t_{max} be the largest value of t for which

$$\frac{1}{2^t} - \frac{2 \ln t}{d} \geq 0 \Rightarrow 2^{t+1} \ln t \leq d.$$

$$\Rightarrow t_{max} = \Omega \left(\frac{\log d}{\log \log \log d} \right).$$

This gives the desired lower bound. ■

Acknowledgements

We thank Pankaj Agarwal and Leonidas Guibas for helpful discussions, and for suggesting that we consider the dual clustering problem.

References

[1] M.S. Aldenderfer and R.K. Blashfield. *Cluster Analysis.* Sage, Beverly Hills, 1984.

[2] M. Bern and D. Eppstein. Approximation Algorithms for Geometric Problems. In: D.S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1996.

[3] P. Brucker. On the complexity of clustering problems. In: R. Henn, B. Korte, and W. Oletti, editors, *Optimization and Operations Research,* Heidelberg, New York, NY, 1977, pp. 45–54.

[4] F. Can. Incremental Clustering for Dynamic Information Processing. *ACM Transactions on Information Processing Systems*, 11 (1993), pp. 143–164.

[5] F. Can and E.A. Ozkarahan. A Dynamic Cluster Maintenance System for Information Retrieval. In *Proceedings of the Tenth Annual International ACM SIGIR Conference,* 1987, pp. 123-131.

[6] F. Can and N.D. Drochak II. Incremental Clustering for Dynamic Document Databases. In *Proceedings of the 1990 Symposium on Applied Computing,* 1990, pp. 61–67.

[7] S. Chakrabarti, C. Phillips, A. Schulz, D.B. Shmoys, C. Stein, and J. Wein. Improved Scheduling Algorithms for Minsum Criteria. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, Springer, 1996.

[8] B.B. Chaudhri. Dynamic clustering for time incremental data. *Pattern Recognition Letters*, 13 (1994), pp. 27-34.

[9] D.R. Cutting, D.R. Karger, J.O. Pederson, and J.W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference*, 1992, pp. 318–329.

[10] D.R. Cutting, D.R. Karger, and J.O. Pederson. Constant interaction-time Scatter/Gather Browsing of Very Large Document Collections. In *Proceedings of the 16th Annual International ACM SIGIR Conference*, 1993, pp. 126–135.

[11] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, NY, 1973.

[12] B. Everitt. *Cluster Analysis.* Heinemann Educational, London, 1974.

[13] C. Faloutsos and D.W. Oard. A Survey of Information Retrieval and Filtering Methods. Technical Report CS-TR-3514, Department of Computer Science, University of Maryland, College Park, 1995.

[14] T. Feder and D.H. Greene. Optimal Algorithms for Approximate Clustering. *In Proceedings of the Twentieth Annual Symposium on Theory of Computing*, 1988, pp. 434–444.

[15] R.J. Fowler, M.S. Paterson, and S.L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters,* 12 (1981), pp. 133–137.

[16] W. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[17] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[18] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *In Proceedings of the Seventh ACM-SIAM Symposium on Discrete Algorithms*, 1996, pp. 152–157.

[19] T.E. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science,* 38 (1985), pp. 293–306.

[20] J.A. Hartigan. *Clustering Algorithms.* Wiley, New York, NY, 1975.

[21] D. Hochbaum. Various Notions of Approximations: Good, Better, Best, and More. In: D.S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1996.

[22] D.S. Hochbaum and W. Maas. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *Journal of the ACM*, 32 (1985), pp. 130–135.

[23] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research,* 10 (1985), pp. 180–184.

[24] D.S. Hochbaum and D.B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM,* 33 (1986), pp. 533–550.

[25] S. Irani and A. Karlin. Online Computation. In: D.S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1996.

[26] N. Jardine and C.J. van Rijsbergen. The Use of Hierarchical Clustering in Information Retrieval. *Information Storage and Retrieval*, 7 (1971), pp. 217–240.

[27] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, NJ, 1988.

[28] O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems, part I: the p-centers problem. *SIAM Journal of Applied Mathematics,* 37 (1979), pp. 513–538.

[29] N. Megiddo and K.J. Supowit. On the complexity of some common geometric problems. *SIAM Journal on Computing,* 13 (1984), pp. 182–196.

[30] S.G. Mentzer. Lower bounds on metric k-center problems. Manuscript, 1988.

[31] R. Motwani, S. Phillips and E. Torng. Non-Clairvoyant Scheduling, In *Proceedings of the Fourth ACM-SIAM Symposium on Discrete Algorithms*, 1993, pp. 422–431. See also: *Theoretical Computer Science,* 130 (1994), pp. 17–47.

[32] E. Omiecinski and P. Scheuermann. A Global Approach to Record Clustering and File Organization. In *Proceedings of the Third BCS-ACM Symposium on Research and Development in Information Retrieval,* 1984, pp. 201–219.

[33] J. Pach and P.K. Agarwal. *Combinatorial Geometry.* John Wiley & Sons, New York, NY, 1995.

[34] E. Rasmussen. Clustering Algorithms. Chapter 16 in: W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms,* Prentice-Hall, Englewood Cliffs, NJ, 1992.

[35] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[36] C. Rogers. A note on coverings. *Mathematika,* 4 (1957), pp. 11–6.

[37] G. Salton. *Automatic Text Processing.* Addison-Wesley, Reading, MA, 1989.

[38] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, New York, NY, 1983.

[39] P. Willett. Recent Trends in Hierarchical Document Clustering: A Critical Review. *Information Processing & Management,* 24 (1988), pp. 577–597.

[40] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, New York, NY, 1994.