Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT

Journal of Computer and System Sciences 68 (2004) 417–441

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

http://www.elsevier.com/locate/jcss

Clustering to minimize the sum of cluster diameters

Moses Charikar[a,1] and Rina Panigrahy[b]

[a] *Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08544, USA*
[b] *Cisco Systems, 170 West Tasman Drive, San Jose, CA 95134, USA*

Received 1 October 2001; revised 2 July 2003

Abstract

We study the problem of clustering points in a metric space so as to minimize the sum of cluster diameters or the sum of cluster radii. Significantly improving on previous results, we present a primal–dual based constant factor approximation algorithm for this problem. We present a simple greedy algorithm that achieves a logarithmic approximation. This also applies when the distance function is asymmetric and the objective is to minimize the sum of cluster radii. The previous best-known result obtained a logarithmic approximation with a constant factor blowup in the number of clusters. We also obtain an incremental clustering algorithm that maintains a solution whose cost is at most a constant factor times that of optimal with a constant factor blowup in the number of clusters.

1. Introduction

Clustering problems have been studied extensively in Computer Science, Operations Research and other fields. Several researchers have studied clustering problems as optimization problems with different objective functions where the goal is to minimize the objective function. The well-known p-center problem [10,15] is the clustering problem with the objective function being the maximum cluster diameter (or radius). Recently, there have been significant advances in our understanding of objective functions such as facility location [3,5,7,8,11,16,18,21] and k-median [3,4,16,17,19], both related to minimizing the sum of distances of demand points from their cluster

E-mail addresses: moses@cs.princeton.edu (M. Charikar), rinap@cisco.com (R. Panigrahy).

[1] Most of this work was done while the author was at the Computer Science Dept., Stanford University, Stanford, CA 94305. His research was supported by the Pierre and Christine Lamond Fellowship, ARO MURI Grant DAAH04-96-1-0007 and NSF Grant IIS-9811904. Research at Princeton supported by NSF ITR grant CCR-0205594 and DOE Early Career Principal Investigator award DE-FG02-02ER25540.

centers. Also, researchers have studied min-sum clustering, where the objective function is the sum of all intra-cluster distances [1,13].

In this paper, we consider the problem of clustering a set of points in a metric space into a specified number of clusters so as to minimize the sum of cluster diameters. Surprisingly little is known about this fairly natural objective function. This has been suggested in the literature as an alternative to the p-center objective in certain applications so as avoid the so-called *dissection effect*. Using the maximum diameter as the objective sometimes results in objects that should have been placed in the same cluster to be placed in different clusters. The sum of diameters objective is more useful as it reduces this dissection effect [14,20]. In a recent paper, Doddi et al. [9] considered this problem and designed approximation algorithms for it. They gave an algorithm that obtains a logarithmic approximation with a constant factor blowup in the number of clusters. For a constant number of clusters, they give a 2 approximation. On the negative side, they proved that it is NP-Hard to obtain an approximation factor $2 - \varepsilon$ for any $\varepsilon > 0$. In the absence of triangle inequality, they show that it is NP-Hard to obtain any approximation factor even with only three clusters.

In this paper, we present significant improvements on these results. We will consider the problem of minimizing the sum of cluster radii which is always within a factor of 2 of the sum of cluster diameters objective when the distances are symmetric. In Section 2, we present a greedy algorithm that obtains a logarithmic approximation using at most k clusters. This also applies to the case when the distances are asymmetric (for the objective of minimizing the sum of cluster radii). We also give primal–dual based algorithms which achieve a constant factor approximation using at most k clusters. Our algorithms draw numerous ideas from the recent work on the k-median problem by Jain and Vazirani [16] as well as Charikar and Guha [3]. It is interesting that our results parallel several of the results for the k-median problem even though the two objective functions are very different; one sums up contributions of demand points while the other sums of contributions per cluster. Finally, in Section 6, we present an incremental algorithm for clustering with this objective function. The algorithm maintains a solution with cost only a constant factor worse than the optimal solution with a constant factor blowup in the number of clusters. This is interesting since such a bicriteria guarantee was not known previously even in the offline case.

1.1. Preliminaries

Definition 1.1 (sum-diameters). Given n points in a metric space with distance function $d(i,j)$, the goal is to divide the points into k clusters so as to minimize the sum of cluster diameters. Here, the diameter of a cluster C is $\max_{i,j \in C} \{d(i,j)\}$, i.e. the maximum distance between any two points in C.

A closely related problem is to minimize the sum of cluster radii:

Definition 1.2 (sum-radii). Given n points in a metric space with distance function $d(i,j)$, the goal is to divide the points into k clusters and select one point in each cluster as the cluster center so as to minimize the sum of cluster radii. The radius of a cluster C with center $i \in C$ is $\max_{j\in C}\{d(i,j)\}$, i.e. the maximum distance of any point in the cluster from the cluster center.

An α approximation algorithm for one problem yields a 2α approximation for the other. It will be convenient to deal with the sum-radii formulation, so we will use that for the remainder of the paper. Also, we will assume that each cluster center is required to lie at one of the given n points. If arbitrary centers are allowed, our approximation guarantees apply with a multiplicative factor of 2.

2. A greedy algorithm

The greedy algorithm starts by placing each of the n points in singleton clusters. At each step, the algorithm operates on the set of centers of the current clusters. In a single step, some of the current clusters are merged and their centers replaced by the center of the new cluster. This process is continued till there are at most k clusters remaining. The algorithm maintains a set of points $\mathscr{C}$ and a set of clusters centered at points in $\mathscr{C}$ such that these clusters cover the original point set. We will use C_i to denote the cluster centered at $i \in \mathscr{C}$. Also, we will represent clusters by the set of original points covered by the cluster.

Algorithm Greedy-Cluster

```
1. Initialize $\mathscr{C}$ to be the set of $n$ points.
2. For each point $i \in \mathscr{C}$, $C_i = \{i\}$.
3. Repeat steps 3a to 3d, until $|\mathscr{C}| \leq k$.
   (a) For every point $i$ (in the original point set) and radius $r$, let $S_i(r)$ be the points in $\mathscr{C}$ within
       distance $r$ of $i$, and $n_i(r) = \min(|S_i(r)| - 1, |\mathscr{C}| - k)$.
   (b) Let $i^*, r^*$ be values of $i$ and $r$ respectively that minimize $\frac{r}{n_i(r)}$.
   (c) Set $C_{i^*} = \bigcup_{j \in S_{i^*}(r^*)} C_j$
   (d) $\mathscr{C} = \mathscr{C} \backslash S_{i^*}(r^*) \cup \{i^*\}$.
4. For each $i \in \mathscr{C}$, output cluster $C_i$.
```

Let $r^{(t)}$ be the radius of the cluster chosen (i.e. the value of r^*) in the tth iteration.

Lemma 2.1. *The sum of the radii of the clusters produced by the greedy algorithm at the end of the t'th iteration is at most $\sum_{s=1}^{t} r^{(s)}$.*

Proof. We will prove that the sum of cluster radii increases by at most $r^{(t)}$ in the tth iteration. Initially, this sum is 0, hence this will prove the claim.

Let i^*, r^* be the values selected by the algorithm in step 3b of the tth iteration. The clusters $C_j, j \in S_{i^*}(r^*)$ are merged to form a single cluster centered at i^*. Note that all the centers of the merged clusters are within a distance r^* of the new cluster center i^*. Hence the radius of the merged cluster C_{i^*} is at most

$$r^* + \max_{j \in S_{i^*}(r^*)} \text{radius}(C_j).$$

Since the new cluster is added to the collection and the clusters $C_j,\ j \in S_{i^*}(r^*)$ are removed, the sum of cluster radii increases by

$$r^* + \max_{j \in S_{i^*}(r^*)} \text{radius}(C_j) - \sum_{j \in S_{i^*}(r^*)} \text{radius}(C_j) \leqslant r^*.$$

Therefore, the sum of cluster radii increases by at most $r^* = r^{(t)}$. □

Let Opt be the cost of the optimal solution.

Lemma 2.2. *Suppose $\mathscr{C}$ is the set of points at the beginning of an iteration and i^*, r^* are the values picked in step* 3b *of the algorithm. Then,*

$$\frac{r^*}{n_{i^*}(r^*)} \leqslant \frac{\text{Opt}}{|\mathscr{C}| - k}.$$

Proof. Note that clusters centered at all the original points are considered in step 3a, so the clusters in the optimal solution are also considered in this step. Let i_j be the center of the jth cluster in the optimal solution and r_j be its radius. We use $S_i(r), n_i(r)$ to denote the values computed in step 3a (using $\mathscr{C}$ for the current iteration). Each point in $\mathscr{C}$ must be in at least one of the clusters in the optimal solution. Hence

$$\sum_{j=1}^{k} |S_{i_j}(r_j)| \geqslant |\mathscr{C}|,$$

$$\sum_{j=1}^{k} |S_{i_j}(r_j)| - 1 \geqslant |\mathscr{C}| - k,$$

$$\sum_{j=1}^{k} n_{i_j}(r_j) \geqslant |\mathscr{C}| - k.$$

The last inequality follows from the previous inequality and the fact that $n_{i_j}(r_j) = \min(|S_{i_j}(r_j)| - 1, |\mathscr{C}| - k)$. Now,

$$\frac{r^*}{n_{i^*}(r^*)} \leqslant \min_j \frac{r_j}{n_{i_j}(r_j)} \leqslant \frac{\sum_{j=1}^{k} r_j}{\sum_{j=1}^{k} n_{i_j}(r_j)} \leqslant \frac{\text{Opt}}{|\mathscr{C}| - k}. \qquad \square$$

Theorem 2.1. *Given an instance of the* sum-radii *problem, algorithm* Greedy-Cluster *produces a solution that is within $H(n-k)$ of the optimal solution and runs in time $O(n^3)$. Here $H(n) = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$.*

Proof. By Lemma 2.1, we need to bound the sum of the radii r^* chosen in step 3b. We will prove that this is at most $\text{Opt} \cdot H(n-k)$ by induction on n.

Suppose the statement holds for $|\mathscr{C}| < n$. We will prove this for $|\mathscr{C}| = n$. Consider the values of i^*, r^* selected by the algorithm in step 3b. When the clusters whose centers are in $S_{i^*}(r^*)$ are replaced by a single cluster centered at i^*, the number of cluster centers (i.e. $|\mathscr{C}|$) drops by $|S_{i^*}(r^*)| - 1$. We consider two cases:

Case 1: $|S_{i^*}(r^*)| - 1 \geqslant |\mathscr{C}| - k$.

In this case, $n_{i^*}(r^*) = |\mathscr{C}| - k$. By Lemma 2.2,

$$\frac{r^*}{n_{i^*}(r^*)} \leqslant \frac{\text{Opt}}{|\mathscr{C}| - k}.$$

Hence, $r^* \leqslant \text{Opt}$. Note that the algorithm stops after this, since the number of cluster centers is not more than k after this iteration. The cost of the greedy algorithm is bounded by Opt in this case.

Case 2: $|S_{i^*}(r^*)| - 1 < |\mathscr{C}| - k$.

In this case, $n_{i^*}(r^*) = |S_{i^*}(r^*)| - 1$. By Lemma 2.2,

$$\frac{r^*}{n_{i^*}(r^*)} \leqslant \frac{\text{Opt}}{|\mathscr{C}| - k}.$$

Hence,

$$\begin{aligned} r^* &\leqslant \text{Opt} \cdot \frac{|S_{i^*}(r^*)| - 1}{n-k} \\ &\leqslant \text{Opt}\left(\frac{1}{n-k} + \cdots + \frac{1}{n-k-|S_{i^*}(r^*)| + 2}\right) \\ &= \text{Opt}(H(n-k) - H(n-k-|S_{i^*}(r^*)| + 1). \end{aligned}$$

Let n' be the number of cluster centers after this step. Then $n' = n - |S_{i^*}(r^*)| + 1$. By the inductive hypothesis, the sum of cluster radii r^* selected by the greedy algorithm beyond this point is bounded by $\text{Opt} \cdot H(n-k-|S_{i^*}(r^*)| + 1)$. Hence the cost of the greedy algorithm is bounded by $\text{Opt} \cdot H(n-k)$.

The algorithm can be implemented to run in $O(n^3)$ time. For each point i, we maintain a list of the points in $\mathscr{C}$ sorted by distance from i. These n lists can be created initially in time $O(n^2 \log n)$. From these lists, each iteration can be performed in $O(n^2)$ time. As there are at most n iterations, the running time of the algorithm is $O(n^3)$. □

We note that the greedy algorithm applies in the asymmetric case as well, i.e. when distances satisfy the triangle inequality, but $d(i,j)$ is not necessarily equal to $d(j,i)$. The radius of a cluster C with center i is $\max_{j \in C} d(i,j)$. The analysis of the greedy algorithm goes through for this case, giving an $O(\log n)$ approximation.

3. Linear programming relaxation

Our algorithms will be primal–dual based. In this section we present the primal and dual LPs on the basis of which our algorithms are devised.

We start with the LP formulation for the sum-radii problem. For every center i and radius r, $\mathbf{y}_i^{(r)}$ is an indicator variable that indicates if there is a cluster of radius r centered at i. The primal LP is as follows:

$$\min \sum_i \sum_r r \cdot \mathbf{y}_i^{(r)}, \tag{1}$$

$$\forall j \sum_i \sum_{r\,:\,d(i,j) \leqslant r} \mathbf{y}_i^{(r)} \geqslant 1, \tag{2}$$

$$\sum_i \sum_r \mathbf{y}_i^{(r)} \leqslant k. \tag{3}$$

Constraint (2) says that every point is covered by some cluster. Constraint (3) says that the number of clusters is at most k.

The dual of the previous LP has a variable α_j corresponding to every demand point j. The dual is as follows:

$$\max \sum_j \alpha_j - k \cdot \mathbf{z}, \tag{4}$$

$$\forall i, r \sum_{j\,:\,d(i,j) \leqslant r} \alpha_j \leqslant r + \mathbf{z}. \tag{5}$$

We consider a *facility location* like version of the problem where the number of clusters is not fixed a priori, but every cluster centered at i incurs a fixed cost f_i, i.e. for a cluster of radius r centered at i, the charge to the objective function is $r + f_i$. Note that f_i is independent of the radius r. We refer to this problem as the fixed costs sum-radii problem. The primal LP for this is as follows:

$$\min \sum_i \sum_r \mathbf{y}_i^{(r)} \cdot (r + f_i), \tag{6}$$

$$\forall j \sum_i \sum_{r\,:\,d(i,j) \leqslant r} \mathbf{y}_i^{(r)} \geqslant 1. \tag{7}$$

The dual LP for this is the basis of our algorithms. The dual LP is as follows:

$$\max \sum_j \alpha_j, \tag{8}$$

$$\forall i, r \sum_{j\,:\,d(i,j) \leqslant r} \alpha_j \leqslant r + f_i. \tag{9}$$

4. Primal–dual algorithm

We describe primal–dual algorithms for sum-radii clustering using the linear programming formulations in the previous section.

We start with the fixed costs sum-radii problem. Let $C_i(r)$ be the cluster of radius r centered at i. Demand point j belongs to $C_i(r)$ iff $d(i,j) \leqslant r$. The primal–dual algorithm is the following:

Algorithm PRIMAL–DUAL FIXED COST SUM-RADII

```
1. For all demand nodes $j$, $\alpha_j \leftarrow 0$.
2. Repeat step 2a until every node $j$ belongs to a tight cluster $C(i,r)$. A tight cluster is one for
   which the dual constraint (9) is tight.
   (a) For all nodes $j$ that do not belong to any tight cluster, increase $\alpha_j$ arbitrarily subject to the
       constraint (9).
3. Perform procedure PRUNE-CLUSTER.
```

Note that in step 2a, we can choose to increase the variables in *any* fashion subject to the constraints (9). This will be important later on, when we present the incremental algorithm.

The *cluster pruning step* invoked in the last step of the algorithm proceeds as follows:

Procedure PRUNE-CLUSTER

```
1. Initially, $T$ is the set of tight clusters.
2. $F = \emptyset$.
3. Repeat steps 3a–3d until $T = \emptyset$.
   (a) Let $C_i(r)$ be the cluster of largest radius in $T$ (break ties arbitrarily).
   (b) Let $N$ be the set of all clusters in $T$ that intersect $C_i(r)$ (including $C_i(r)$).
   (c) $F = F \cup \{C_i(3r)\}$.
   (d) $T = T \setminus N$.
4. Output $F$.
```

Lemma 4.1. *Every demand point belongs to at least one cluster in the final set of clusters produced by the algorithm* PRIMAL–DUAL FIXED COST SUM-RADII.

Proof. Every demand point j belongs to at least one tight cluster. We will prove that every tight cluster is contained in one of the clusters in the final solution. Consider a tight cluster $C_i(r)$. We consider two cases:

Case 1: $C_i(r)$ was selected in step 3a.

Then the cluster $C_i(3r)$ belongs to the final solution and clearly $C_i(r)$ is completely contained in it.

Case 2: $C_i(r)$ was not selected in step 3a.

In this case, $C_i(r)$ must intersect some $C_{i'}(r')$ where $C_{i'}(r')$ was selected in step 3a. Since the largest radius set is picked in this step, $r' \geqslant r$. Hence $C_i(r)$ is completely contained in $C_{i'}(3r')$ which is one of the sets in the final solution. □

Let C be the sum of cluster radii and let F be the sum of the fixed costs f_i for the clusters in solution produced by algorithm PRIMAL–DUAL FIXED COST SUM-RADII. We prove the following theorem which is similar to a corresponding result in [16] (Theorem 7).

Theorem 4.1.

$$C + 3F \leqslant 3 \sum_j \alpha_j. \tag{10}$$

Proof. Consider a cluster $C_i(3r)$ in the final solution. This contributes $3r + 3f_i$ to the LHS of (10). Recall that f_i is independent of r. Note that this cluster $C_i(3r)$ was placed in the final solution because tight cluster $C_i(r)$ was picked in step 3a at some point. We will charge the cost of cluster $C_i(3r)$ to the dual value of the demand points j in $C_i(r)$. Since $C_i(r)$ is tight,

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j = r + f_i.$$

Hence the total contribution of the demand points within $C_i(r)$ to the RHS of (10) is $3r + 3f_i$. Every $\boldsymbol{\alpha}_j$ is counted at most once in this accounting process, since the clusters $C_i(r)$ picked in step 3a are disjoint. This proves the theorem. □

We can use this algorithm to get a bicriteria approximation for the sum-radii problem by setting the fixed costs appropriately.

Corollary 4.1. *For the* sum-radii *problem, for any* $\gamma > 0$, *we can produce a solution with cost* $3(1 + \frac{1}{\gamma})$Opt *using at most* $\lceil k(1+\gamma) \rceil$ *clusters for arbitrarily small* δ.

Proof. Let Opt be the optimal cost of a solution to the sum-radii problem using exactly k clusters. Set $f_i = \frac{\text{Opt}}{\gamma k}$. Then the optimal cost to the sum-radii with fixed costs problem is at most $(1 + \frac{1}{\gamma})$Opt. Hence $\sum_j \alpha_j \leqslant (1 + \frac{1}{\gamma})$Opt, since the dual value $\sum_j \alpha_j$ is a lower bound on the optimal cost. From Theorem 4.1, $C + 3F \leqslant 3\sum_j \alpha_j$. Hence C, the sum of cluster radii is bounded by $3(1 + \frac{1}{\gamma})$Opt. Let k' be the number of clusters in the final solution. Then $F = k' \dfrac{\text{Opt}}{\gamma k}$. Hence $k' \leqslant k(1+\gamma)$.

Now, we do not know the value of Opt a priori, but we can guess this to within a factor of $(1+\delta)$. This gives the claimed result. □

In the next section, we will need to use a slightly modified version of the algorithm and a correspondingly modified bound (10). After growing the dual variables, the algorithm selects its final clusters from set T consisting of tight clusters which satisfy

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j = r + f_i.$$

Suppose instead of this, the algorithm starts with set T being an arbitrary subset of clusters that satisfy

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \geqslant r + f_i - \varepsilon,$$

such that every demand point is contained in at least one cluster in T. Then, we can prove the following weaker guarantee:

$$C + 3F \leqslant 3\sum_j \alpha_j + 3k''\varepsilon,$$

where k'' is the number of clusters returned by the procedure PRUNE-CLUSTER.

Let C' be the sum of original radii of the clusters picked up in the solution returned by the primal–dual algorithm (i.e. before they were expanded to a factor of 3). In other words, $C = 3C'$. Then,

$$C' + F \leqslant \sum_j \alpha_j + k''\varepsilon. \tag{11}$$

5. Approximation using exactly k clusters

We can get a constant factor approximation using exactly k clusters. Our approach is the same as in [16], where a primal–dual algorithm for facility location was used to obtain an algorithm for the k-median problem. We exploit the similarity between the LP formulations of the sum-radii problem and the fixed costs sum-radii problem. Notice that by setting the fixed costs $f_i = z$, the constraints in the fixed costs sum-radii dual LP (8)–(9) are exactly the same as the constraints in the sum-radii dual LP (4)–(5). Thus a feasible solution to the former is a feasible solution to the latter (although with different objective function value).

The overall outline is similar to previous work on using Lagrangean relaxation and the primal–dual framework for k-median [16] and k-MST [6]. The basic idea is to invoke the primal–dual algorithm we described previously for appropriate settings of the fixed costs to obtain two solutions: one with $\leqslant k$ and the other with $\geqslant k$ clusters. Then we combine these two solutions to obtain a solution with at most k clusters. Our combination step is more complicated than the corresponding step in [16].

Firstly, we will need some bound on the largest cluster radii used by the optimal solution, in order to bound one of the terms in the cost of the combined solution. In order to obtain a crude bound, it suffices to know the largest cluster radius. However, we will guess the largest ℓ clusters used by the optimal. This is an optimization to reduce the approximation factor. ℓ will be finally set to $O(1/\varepsilon)$ and we will incur an additive factor of ε in the approximation ratio. Secondly, before we combine the two solutions, we add clusters from one solution to the other. This guarantees certain intersection properties for the clusters in the two solutions which are crucial for our combination step to proceed. A similar idea was used earlier in [3] for the k-median problem, but there the purpose was to obtain an improved approximation factor.

We now present the overall algorithm deferring some of the details for later discussion.

Algorithm PRIMAL–DUAL SUM-RADII

1. Guess the centers $i_1, \ldots, i_\ell$ and radii $r_1, \ldots, r_\ell$ of the largest ℓ clusters in the optimal solution. Let $k' = k - \ell$. The remaining steps find at most k' clusters to cover the points outside the guessed ℓ clusters.

2. Run algorithm PRIMAL–DUAL FIXED COST SUM-RADII with all fixed costs set to z, allowing clusters of radius at most r_ℓ with the following rule for increasing dual variables α_j: at any point of time, uniformly increase α_j for all j that are not contained in a tight cluster.
   Perform a binary search on z to find two values z_1 and z_2, $|z_1 - z_2| \leqslant \text{OPT}/(2^n \cdot \ell \cdot n)$, such that the algorithm produces $\leqslant k'$ clusters for z_1 and $\geqslant k'$ clusters for z_2.
3. Let T_1 (respectively T_2) be the set of tight clusters obtained by the algorithm for z_1 (resp. z_2). Let $\bar{F}_1 \subseteq T_1$ be the set of original clusters picked in the solution for z_1 (before expanding by a factor of 3) and let $\bar{F}_2 \subseteq T_2$ be the set of original clusters picked in the solution for z_2.[2]
   Identify clusters in $\bar{F}_2$ that are disjoint from all clusters in $\bar{F}_1$. Add these clusters, one at a time, to $\bar{F}_1$ (without deleting them from $\bar{F}_2$ until either $|\bar{F}_1| = k'$ or no such clusters exist. If $|\bar{F}_1| = k'$, return the clusters in $\bar{F}_1$ expanded by a factor of 3, else perform the remaining steps.
4. Let F_1 denote the final value of $\bar{F}_1$ at the end of the previous step and (for uniform notation) let $F_2 = \bar{F}_2$. Let the solutions represented by these be $\mathsf{soln}^{(1)}$ (respectively $\mathsf{soln}^{(2)}$), with $|F_1| = k_1 \leqslant k'$ (resp. $|F_2| = k_2 \geqslant k'$) clusters.
   Express k' as a convex combination of k_1 and k_2 with coefficients a and b, i.e.
   $$a + b = 1, \quad a \cdot k_1 + b \cdot k_2 = k'.$$
   In fact,
   $$a = \frac{k_2 - k'}{k_2 - k_1}, \quad b = \frac{k' - k_1}{k_2 - k_1}.$$
5. Perform procedure GROUP-CLUSTERS to group clusters, each group containing one cluster from F_1 and one or more clusters from F_2. All clusters in F_2 are grouped with some cluster in F_1, although some clusters in F_1 may not be in any group.
6. For group q, let $F_1^{(q)}$ denote the single cluster from F_1 and let $F_2^{(q)}$ denote the clusters from F_2. Further, let $F_2^{(q)} \times 3$ denote the set of clusters $C_i(3r)$ for every cluster $C_i(r)$ in $F_2^{(q)}$. Perform randomized procedure CONSTRUCT-DISTRIBUTION to assign each groups to be in configuration 1 or 2. For each group q in configuration 1, we pick the cluster $C_i(r) \in F_1^{(q)}$ and expand it to cover all the clusters in $F_2^{(q)} \times 3$. For each group in configuration 2, we pick the clusters $C_i(3r)$ for every cluster $C_i(r)$ in $F_2^{(q)}$.
7. Return the better of the two solutions $\mathsf{soln}^{(1)}$ and the solution produced as the output of the previous step.

5.1. Analysis

Note that there are at most $n^{2\ell}$ possibilities for the guesses made by the algorithm in step 1. If our guess is correct, then the ℓth largest radius $r_\ell \leqslant \text{OPT}/\ell$. We will describe the analysis assuming

[2] Note that the clusters in $\bar{F}_1$ are disjoint (because of the way in which they were picked) and similarly, the clusters in $\bar{F}_2$ are disjoint.

that the algorithm guesses correctly. Note that the optimal solution to the new instance excluding the points covered by the guessed clusters is

$$\text{Opt}' = \text{Opt} - (r_1 + \cdots + r_\ell).$$

We will show later that $|z_1 - z_2| \leqslant \text{Opt}/(2^n \cdot \ell \cdot n)$ can be achieved in polynomially many iterations.

Let $\alpha_j^{(1)}$ (resp. $\alpha_j^{(2)}$) be the dual variable corresponding to demand point j in the algorithm with fixed cost z_1 (resp. z_2). If indeed $|\bar{F}_1| = k'$ (in step 3), then we can show that the clusters in $\bar{F}_1$ (expanded by a factor of 3) give a $3 + \varepsilon$ approximation. For most of the following discussion, we assume that this is not the case, i.e. all clusters from $\bar{F}_2$ that could be added to $\bar{F}_1$ are indeed added.

Note that the clusters in F_1 are disjoint and the clusters in F_2 are disjoint. Further, every cluster in F_2 intersects some cluster in F_1. It is this property guaranteed by step 3, that is useful in the combining of the two solutions. Let the solutions represented by F_1 and F_2 be $\mathsf{soln}^{(1)}$ (respectively $\mathsf{soln}^{(2)}$), with $|F_1| = k_1 \leqslant k'$ (resp. $|F_2| = k_2 \geqslant k'$) clusters.

Let $z = \max(z_1, z_2)$ and let $\alpha_j = \min(\alpha_j^{(1)}, \alpha_j^{(2)})$. Note that for every cluster $C_i(r)$,

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \leqslant r + z.$$

Each cluster $C_i(r)$ in F_1 or F_2 must have been a tight cluster in either T_1 or T_2. In this case, we will show that (using continuity)

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \geqslant r + z - \frac{\text{Opt}}{\ell \cdot n}. \tag{12}$$

The proof of this is deferred to Section 5.2.

Note that the values α_j and z are a valid solution to the dual LP (4)–(5). Hence $\sum_j \alpha_j - k' \cdot z$ is a lower bound on the value of the optimal solution.

Let C_1' (respectively C_2') be the sum of *original* cluster radii (before they were expanded by a factor of 3) in $\mathsf{soln}^{(1)}$ (resp. $\mathsf{soln}^{(2)}$). The total fixed cost (using a fixed cost of z per cluster) is $k_1 \cdot z$ in $\mathsf{soln}^{(1)}$ and $k_2 \cdot z$ in $\mathsf{soln}^{(2)}$. We will use the modified primal–dual guarantee (11), for an appropriate value of ε.

$$C_1' + k_1 \cdot z \leqslant \sum_j \alpha_j^{(1)} + k_1 \frac{\text{Opt}}{\ell \cdot n}, \tag{13}$$

$$C_2' + k_2 \cdot z \leqslant \sum_j \alpha_j^{(2)} + k_2 \frac{\text{Opt}}{\ell \cdot n}, \tag{14}$$

$$a \cdot C_1' + b \cdot C_2' \leqslant \left(\sum_j \alpha_j - k' \cdot z \right) + k' \frac{\text{Opt}}{\ell \cdot n}, \tag{15}$$

$$a \cdot C_1' + b \cdot C_2' \leqslant \text{Opt}' + \frac{\text{Opt}}{\ell}. \tag{16}$$

We will bound the cost of the final solution in terms of $a \cdot C_1' + b \cdot C_2'$. From the above analysis, this is bounded by $\text{OPT}' + \text{OPT}/\ell$.

Grouping clusters from different solutions: We group each cluster in F_1 with clusters from F_2 as follows.

Procedure GROUP-CLUSTERS

```
1. $F_1' = F_1$, $F_2' = F_2$.
2. Repeat steps 2a–2e until $F_1' = \emptyset$
   (a) Let $C_i(r)$ be any cluster in $F_1'$.
   (b) Let $N$ be the set of clusters in $F_2'$ that intersect $C_i(r)$.
   (c) $F_1' = F_1' \backslash \{C_i(r)\}$.
   (d) $F_2' = F_2' \backslash N$.
   (e) If $N \neq \emptyset$, output the group consisting of $C_i(r)$ together with the clusters in $N$. (If $N = \emptyset$,
       output nothing).
```

Lemma 5.1. *Every cluster in F_2 is grouped with some cluster in F_1 by procedure* GROUP-CLUSTERS.

Proof. We prove that $F_2' = \emptyset$ at the end of procedure GROUP-CLUSTERS. Suppose F_2' is non-empty at the end of this procedure. Let $C_i(r)$ be a cluster in F_2 that remains in F_2' at the end. Note that $C_i(r)$ does not intersect any cluster in F_1. This is a contradiction. □

Probability distribution on the groups: Note that each group contains one cluster from F_1 and one or more clusters from F_2. For each group q, we either pick (suitably expanded versions of) the cluster $F_1^{(q)}$ or all the clusters in $F_2^{(q)}$ such that $F_2^{(q)} \times 3$ is completely covered. The goal is to do this in such a way that a total of at most k' clusters are picked. Further, for each group, $F_1^{(q)}$ is picked with probability roughly a and $F_2^{(q)}$ is picked with probability roughly b.

For group q, let $n_q = |F_2^{(q)}| - 1$. Let $k_1' \leqslant k_1$ be the number of groups produced. Then $\sum_q n_q = k_2 - k_1'$.

In Section 5.2, we describe procedure CONSTRUCT-DISTRIBUTION which we will use to construct the required probability distribution. CONSTRUCT-DISTRIBUTION takes as input the set of numbers $N = \{n_q : q = 1 \ldots t\}$ and a probability value $p \in [0, 1]$. It produces two sets S_1 and S_2 which are disjoint subsets of N. We state two lemmas about the properties of CONSTRUCT-DISTRIBUTION we will use in our analysis. The proofs appears in Section 5.2.

Lemma 5.2. *For subsets S_1, S_2 produced by* CONSTRUCT-DISTRIBUTION, *at most one $n_q \notin S_1 \cup S_2$ and*

$$\sum_{n_q \in S_2} n_q \leqslant p \sum_q n_q.$$

Lemma 5.3. CONSTRUCT-DISTRIBUTION constructs a probability distribution such that $\mathbf{Pr}[n_q \in S_2] \leqslant p$ and $\mathbf{Pr}[n_q \in S_1] \leqslant 1 - p$.

We apply procedure CONSTRUCT-DISTRIBUTION on the set of numbers n_q with probability p set to b. The allocation of clusters is determined by the disjoint sets S_1, S_2 produced by the procedure. For each $n_q \in S_2$, we say that group q is in configuration 2. For $n_q \in S_1$, we say that group q is in configuration 1. If $n_q \notin S_1 \cup S_2$, we say that group q is in configuration 1 and mark this group as a *special* group. Note that there can be at most one such group. We will need to treat this group separately in the analysis later on.

For each group in configuration 1, we pick the cluster $C_i(r) \in F_1^{(q)}$ and expand it to cover all the clusters in $F_2^{(q)} \times 3$. For each group in configuration 2, we pick the clusters $C_i(3r)$ for every cluster $C_i(r)$ in $F_2^{(q)}$.

The total number of clusters picked by this procedure is at most $k'_1 + b \cdot (\sum_q n_q) = k'_1 + b(k_2 - k'_1) = a \cdot k'_1 + b \cdot k_2 \leqslant a \cdot k_1 + b \cdot k_2 = k'$.

Lemma 5.4. *The expected cost of the solution produced is at most* $a \cdot C'_1 + (3 + a) \cdot C'_2 + 5\text{OPT}/\ell$.

Proof. For group q, let $C_1^{(q)}$ denote the radius of the cluster in $F_1^{(q)}$ and let $C_2^{(q)}$ denote the sum of radii of the clusters in $F_2^{(q)}$. Then $\sum_q C_1^{(q)} \leqslant C'_1$ and $\sum_q C_2^{(q)} = C'_2$. For a cluster in configuration 1, the radius of the cluster required to cover all clusters in $F_2^{(q)} \times 3$ is at most $C_1^{(q)} + 4C_2^{(q)}$. For a cluster in configuration 2, the sum of radii of the clusters chosen is $3C_2^{(q)}$. Note that the maximum radius of a single cluster required to cover all clusters in $F_2^{(q)} \times 3$ is at most $5r_\ell \leqslant 5\text{OPT}/\ell$. (Recall that r_ℓ was the guessed radius of the ℓth largest cluster).

We consider various possibilities for group q based on whether CONSTRUCT-DISTRIBUTION places n_q in S_1 or S_2 or neither. With probability at most b, $n_q \in S_2$, group q is in configuration 2 and the contribution to the cost of the solution is $3C_2^{(q)}$. With probability at most $1 - b = a$, $n_q \in S_1$, group q is in configuration 1 and the contribution to the cost of the solution is $C_1^{(q)} + 4C_2^{(q)}$. Further, at most one $n_q \notin S_1 \cup S_2$. The corresponding group q is in configuration 1 and the contribution to the cost is at most $5\text{OPT}/\ell$. Thus the expected cost of the solution is at most

$$\begin{aligned} & a \cdot \sum_q (C_1^{(q)} + 4C_2^{(q)}) + b \cdot \sum_q 3C_2^{(q)} + 5\text{OPT}/\ell \\ & = a \cdot \sum_q C_1^{(q)} + (3 + a) \sum_q C_2^{(q)} + 5\text{OPT}/\ell \\ & \leqslant a \cdot C'_1 + (3 + a) \cdot C'_2 + 5\text{OPT}/\ell. \qquad \square \end{aligned}$$

Recall that we also have solution $\mathsf{soln}^{(1)}$ with $k_1 \leqslant k'$ clusters and cost $3C'_1$. If $C'_1 \leqslant C'_2$, then $C'_1 \leqslant a \cdot C'_1 + b \cdot C'_2$; in this case, $\mathsf{soln}^{(1)}$ gives a $3 + 1/\ell$ approximation using (16). If $C'_2 \leqslant C'_1$, then the solution cost guaranteed by Lemma 5.4 is at most $a \cdot C'_1 + b \cdot C'_2 + (2 + 2a) \cdot C'_2 + 5\text{OPT}/\ell$, which gives a $5(1+1/\ell)+5/\ell$ approximation. Setting $\ell = 10/\varepsilon$, this gives a $5+\varepsilon$ approximation. Note that the algorithm runs in time $n^{O(1/\varepsilon)}$ as the largest ℓ cluster centers and cluster radii must be guessed by the algorithm.

We can improve the approximation factor by a slightly improved analysis.

Lemma 5.5.

$$\min(3C_1', a \cdot C_1' + (3+a) \cdot C_2') \leqslant \frac{3(3+a)}{(3-a+2a^2)}(a \cdot C_1' + b \cdot C_2').$$

Proof. We want to bound $\min(3C_1', a \cdot C_1' + (3+a) \cdot C_2')$. The minimum is less than any convex combination of the two terms. Consider the convex combination

$$\begin{aligned} &\frac{2a+2a^2}{3-a+2a^2} \times (3C_1') + \frac{3-3a}{3-a+2a^2} \times (aC_1' + (3+a)C_2') \\ &\quad = \frac{3(3+a)a}{3-a+2a^2} \times C_1' + \frac{3(3+a)(1-a)}{3-a+2a^2} \times C_2' \\ &\quad = \frac{3(3+a)}{3-a+2a^2} \times (aC_1' + (1-a)C_2'). \end{aligned}$$

Note that $1-a=b$. This proves the lemma. $\square$

Thus the approximation ratio obtained by taking the better of the two solutions, $\mathsf{soln}^{(1)}$ and the solution guaranteed by Lemma 5.4, yields a $\frac{3(3+a)}{3-a+2a^2}$ approximation (ignoring the $O(1/\ell)$ terms). The expression $\frac{3(3+a)}{3-a+2a^2}$ is maximized for $a = 2\sqrt{2}-3$. For this value of a, the value of the expression is $\frac{3}{8\sqrt{3}-13} \approx 3.503$.

This implies that the approximation ratio is bounded by $3.504 + 9/\ell$. Setting $\ell = 9/\varepsilon$, we get the following theorem:

Theorem 5.1. *There exists a polynomial time randomized algorithm that achieves a* $3.504+\varepsilon$ *approximation for the sum-radii problem using at most k clusters, in time* $n^{O(1/\varepsilon)}$.

The only randomization in the algorithm is in the procedure CONSTRUCT-DISTRIBUTION which decides which of two configurations each group is in. The algorithm can be easily derandomized using the method of conditional expectations. The details are straightforward and hence omitted.

5.2. *Technical proofs*

Continuity analysis: We establish inequality (12) used in the analysis using a continuity argument. Our goal is to show that if the primal–dual algorithm is run for two values of z that are very close, then the resulting values of α_j are also very close. Our proof mimics the continuity argument in [3]. Let $\alpha_j(z)$ denote the value of α_j produced by the primal–dual algorithm when the algorithm is run with all the fixed costs set to z.

Lemma 5.6. *Suppose we execute the algorithm for $z = z_1$ and $z = z_1 + \delta z$. Then, the values of α_j in the two executions differ by at most $2^n|\delta z|$. Further, if the constraint for cluster $C_i(r)$ is tight in one execution, then*

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \geqslant r + z - 2^n|\delta z|$$

in the other execution.

Proof. Suppose we execute separately and in parallel, the algorithm for z_1 and $z_2 = z_1 + \delta z$. An *event* is said to occur if, for some node j, α_j stops increasing in one of the two executions. Thus there are n events in all, one for each node j. Consider these n events in order of occurrence.

Suppose node j participates in the qth event in this sequence. We will prove that

$$|\alpha_j(z_1) - \alpha_j(z_1 + \delta z)| \leqslant 2^{q-1}|\delta z|.$$

We prove this by induction on q.

Suppose the claim is true for the first q events. Consider the $(q+1)$th event. Suppose node j participates in the $(q+1)$th event. It must be the case that in one of the two executions, for some cluster $C_i(r)$ that contains j, the constraint:

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \leqslant r + z$$

must have become tight. At this point of time, in the other execution, the difference between the LHS and the RHS for this constraint can be at most $(1 + \sum_{r=1}^{q} 2^{r-1})|\delta z| = 2^q|\delta z|$. Thus, α_j can increase by at most $2^q|\delta z|$ in the other execution.

A similar proof applies to the base case as well. Hence the claim follows by induction. This proves the first part of the lemma.

To prove the second part, consider a constraint for cluster $C_i(r)$ that becomes tight in one execution:

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \leqslant r + z.$$

At the time at which this constraint becomes tight, the difference between the LHS and the RHS in the other iteration can be at most $(1 + \sum_{q=1}^{n} 2^{q-1})|\delta z| = 2^n|\delta z|$. Hence at this point of time,

$$\sum_{j\,:\,j \in C_i(r)} \alpha_j \geqslant r + z - 2^n|\delta z|$$

in the other execution. Since the value of the LHS can only increase, this inequality remains true at the end of the execution. □

The term $2^n|\delta z|$ in Lemma 5.6 can be made equal to $\textsc{Opt}/n \cdot \ell$ by making $|\delta z| \leqslant \textsc{Opt}/(2^n \cdot n \cdot \ell)$. (This gives us inequality (12) used in the analysis). This in turn is achieved by running the binary search for polynomially many steps. The details are similar to the running time analysis in [3,16].

Lemma 5.7. *In $O(n \log n)$ steps of binary search, we can find z_1, z_2 such that*

- *for $z = z_1$, the primal–dual algorithm returns $\leqslant k'$ clusters and for $z = z_2$, the primal–dual algorithm returns $\geqslant k'$ clusters.*
- *$z_1 - z_2 \leqslant \text{OPT}/(2^n \cdot n \cdot \ell)$.*

Proof. For $z = 0$, the primal–dual algorithm will return a solution with many clusters (one for each point that needs to be covered). Note that we have guessed $r_\ell \leqslant \text{OPT}/\ell$ and the remaining solution uses clusters of radius at most r_ℓ. Thus $\text{OPT}' \leqslant k' r_\ell$. Now, for $z = 2k' r_\ell \geqslant \text{OPT}'$, we claim that the primal–dual algorithm returns at most k' clusters. Suppose the primal–dual algorithm returns a solution with k'' clusters, then

$$3k''z \leqslant 3\text{OPT}' + 3k'z,$$
$$(k'' - k')z \leqslant \text{OPT}'.$$

Since $z > \text{OPT}'$, this implies that $k'' \leqslant k'$.

We conduct a binary search on the interval $[0, 2k' \cdot r_\ell]$ in order to find the required z_1, z_2 with $z_1 - z_2 \leqslant r_\ell/(2^n \cdot n) \leqslant \text{OPT}/(2^n \cdot n \cdot \ell)$. It follows that this can be achieved in $O(n \log n)$ steps. □

5.3. Construction of probability distribution on clusters

Suppose we have a set of numbers $N = \{n_q : q = 1, \dots, t\}$ and probability p, $0 \leqslant p \leqslant 1$. We will choose disjoint subsets $S_1, S_2 \subseteq N$ such that at most one number $n_q \notin S_1 \cup S_2$. We describe a probability distribution on choices of disjoint subsets S_1, S_2 such that $\mathbf{Pr}[n_q \in S_2] \leqslant p$ and $\mathbf{Pr}[n_q \in S_1] \leqslant 1 - p$ and

$$\sum_{n_q \in S_2} n_q \leqslant p \sum_q n_q.$$

The basic idea behind the procedure is quite simple. We examine each value n_q in sequence and decide to either place it in S_1 or S_2 or neither. This choice is randomized. The goal is to (roughly) have $\sum_{n_q \in S_2} n_q = p \sum_q n_q$. We think of the LHS as the required value (reqd). As indices are added to S_2, this required value is correspondingly decreased. We also keep track of the sum of n_q for all q we have not examined so far (including the current one being examined). This is refered to as the remaining value (rem). Excluding boundary cases (i.e. when reqd is very small or very large), we place the current value n_q in S_2 with probability reqd/rem. Note that initially reqd/rem $= p$. The precise description of the procedure is as follows:

Procedure CONSTRUCT-DISTRIBUTION

```
1. $S_1 = \emptyset$, $S_2 = \emptyset$, $\text{rem}_1 = \sum_q n_q$, $\text{reqd}_1 = p \sum_q n_q$.
2. for $q = 1$ to $t$ do steps 3 to 4
```

```
3. if ($0 < \text{reqd}_q < n_q$) { /* reqd very small */
          with probability $n_q/\text{rem}_q$ {
                    $\text{reqd}_{q+1} = 0$
          }
          else {
                    $S_1 = S_1 \cup \{n_q\}$
                    $\text{reqd}_{q+1} = \text{reqd}_q$
          }
  } else if ($\text{rem}_q - n_q < \text{reqd}_q < \text{rem}_q$) { /* reqd very large */
          with probability $1 - n_q/\text{rem}_q$ {
                    $S_2 = S_2 \cup \{n_q\}$
                    $\text{reqd}_{q+1} = \text{reqd}_q - n_q$
          } else {
                    $\text{reqd}_{q+1} = \text{rem}_q - n_q$
          }
  } else {
          with probability $\text{reqd}_q/\text{rem}_q$ {
                    $S_2 = S_2 \cup \{n_q\}$.
                    $\text{reqd}_{q+1} = \text{reqd}_q - n_q$
          }
          else {
                    $S_1 = S_1 \cup \{n_q\}$
                    $\text{reqd}_{q+1} = \text{reqd}_q$
          }
  }
4. $\text{rem}_{q+1} = \text{rem}_q - n_q$.
```

We now prove Lemmas 5.2 and 5.3 about the properties of Construct-Distribution we used in our analysis.

Proof of Lemma 5.2. Note that if n_q is added to neither S_1 or S_2 in step 3, then subsequent n_q are either all put in S_1 or all put in S_2. (This is because either reqd_{q+1} is set to 0, in which $n_{q+1}, n_{q+2}, \ldots$ are placed in S_1, or reqd_{q+1} is set to $\text{rem}_q - n_q$ in which case $n_{q+1}, n_{q+2}, \ldots$ are all placed in S_2.) Thus at most one number n_q is excluded from both S_1 and S_2. Additionally, the procedure maintains the invariant that $\text{reqd} + \sum_{n_q \in S_2} n_q \leqslant p \sum_q n_q$. The procedure also ensures that $\text{reqd} \geqslant 0$. Hence, $\sum_{n_q \in S_2} n_q \leqslant p \sum_q n_q$. □

Proof of Lemma 5.3. We prove that the procedure Construct-Distribution constructs a probability distribution such that $\mathbf{Pr}[n_q \in S_2] \leqslant \text{reqd}_1/\text{rem}_1$ and $\mathbf{Pr}[n_q \in S_1] \leqslant 1 - \text{reqd}_1/\text{rem}_1$. Since $\text{reqd}_1/\text{rem}_1 = p$, this will establish the lemma. We prove this by induction on q.

For the base case, consider the first iteration. We consider several cases corresponding to the conditions checked in step 3 of the procedure.

Case 1: $0 < \text{reqd}_1 < n_1$.
$\mathbf{Pr}[n_1 \in S_2] = 0 < \text{reqd}_1/\text{rem}_1$, $\mathbf{Pr}[n_1 \in S_1] = 1 - n_1/\text{rem}_1 < 1 - \text{reqd}_1/\text{rem}_1$.
Case 2: $\text{rem}_1 - n_1 < \text{reqd}_1 < \text{rem}_1$.
$\mathbf{Pr}[n_1 \in S_2] = 1 - n_1/\text{rem}_1 < \text{reqd}_1/\text{rem}_1$, $\mathbf{Pr}[n_1 \in S_1] = 0 < 1 - \text{reqd}_1/\text{rem}_1$.
Case 3:
$\mathbf{Pr}[n_1 \in S_2] = \text{reqd}_1/\text{rem}_1$, $\mathbf{Pr}[n_1 \in S_1] = 1 - \text{reqd}_1/\text{rem}_1$.

Suppose the claim is true for q iterations. We now prove this for $q+1$ iterations. We need to prove the claim for n_{q+1}, since it is true for $n_1, \ldots, n_q$ by the inductive hypothesis. We consider the various possibilities considered by the procedure for n_1. This results in modified values rem_2 and reqd_2. For each of these possibilities, we can bound the probabilities of n_{q+1} being included in S_1 or S_2 in terms of rem_2 and reqd_2. To do this, we apply the inductive hypothesis to the execution of the q iterations after the first one. We only prove that $\mathbf{Pr}[n_{q+1} \in S_2]$ satisfies the claimed bound. The proof for $\mathbf{Pr}[n_{q+1} \in S_1]$ follows from similar calculations.

Again, we consider cases corresponding to the conditions checked in step 3 of the procedure.
Case 1: $0 < \text{reqd}_1 < n_1$.
With probability n_1/rem_1, $\text{reqd}_2 = 0$ and all subsequent n_q values, $q = 2, \ldots$ are included in S_1.
With probability $1 - n_1/\text{rem}_1$, $\text{reqd}_2 = \text{reqd}_1$. By the inductive hypothesis on iterations 2 through $q+1$, $\mathbf{Pr}[n_{q+1} \in S_2] \leqslant \text{reqd}_2/\text{rem}_2 = \text{reqd}_1/(\text{rem}_1 - n_1)$. Hence, $\mathbf{Pr}[n_{q+1} \in S_2]$ after iterations 1 through $q+1$ is at most

$$\left(\frac{\text{rem}_1 - n_1}{\text{rem}_1}\right) \cdot \left(\frac{\text{reqd}_1}{\text{rem}_1 - n_1}\right) = \frac{\text{reqd}_1}{\text{rem}_1}.$$

Case 2: $\text{rem}_1 - n_1 < \text{reqd}_1 < \text{rem}_1$.
With probability $1 - n_1/\text{rem}_1$, n_1 is added to S_1, $\text{reqd}_2 = \text{reqd}_1 - n_1$. By the inductive hypothesis on iterations 2 through $q+1$, $\mathbf{Pr}[n_{q+1} \in S_2] \leqslant \text{reqd}_2/\text{rem}_2 = (\text{reqd}_1 - n_1)/(\text{rem}_1 - n_1)$.
With probability n_1/rem_1, $\text{reqd}_2 = \text{rem}_1 - n_1$ and all subsequent n_q values are added to S_2. Hence, $\mathbf{Pr}[n_{q+1} \in S_2]$ after iterations 1 through $q+1$ is at most

$$\left(\frac{\text{rem}_1 - n_1}{\text{rem}_1}\right) \cdot \left(\frac{\text{reqd}_1 - n_1}{\text{rem}_1 - n_1}\right) + \left(\frac{n_1}{\text{rem}_1}\right) = \frac{\text{reqd}_1}{\text{rem}_1}.$$

Case 3: With probability $\text{reqd}_1/\text{rem}_1$, n_1 is added to S_2, $\text{reqd}_2 = \text{reqd}_1 - n_1$ By the inductive hypothesis on iterations 2 through $q+1$, $\mathbf{Pr}[n_{q+1} \in S_2] \leqslant (\text{reqd}_1 - n_1)/(\text{rem}_1 - n_1)$.
With probability $1 - \text{reqd}_1/\text{rem}_2$, n_1 is added to S_1, $\text{reqd}_2 = \text{reqd}_1$. By the inductive hypothesis on iterations 2 through $q+1$, $\mathbf{Pr}[n_{q+1} \in S_2] \leqslant \text{reqd}_2/\text{rem}_2 = (\text{reqd}_1)/(\text{rem}_1 - n_1)$.

$$\left(\frac{\text{reqd}_1}{\text{rem}_1}\right) \cdot \left(\frac{\text{reqd}_1 - n_1}{\text{rem}_2 - n_1}\right) + \left(1 - \frac{\text{reqd}_1}{\text{rem}_1}\right) \cdot \left(\frac{\text{reqd}_1}{\text{rem}_1 - n_1}\right) = \frac{\text{reqd}_1}{\text{rem}_1}.$$

By induction, the claim is true. □

6. Incremental algorithm

We obtain an incremental algorithm that clusters points so as to minimize the sum of cluster radii. The notion of an incremental algorithm is similar to that used in [2] where an incremental algorithm was presented for the k-center problem. Such an algorithm must make one pass over the data maintaining at most k clusters and incorporating newly encountered points using a set of very simple operations. When a new point is encountered, the operations available to update the current clustering are

1. The new point can be added to an existing cluster.
2. The new point can be placed in a singleton cluster.
3. At any point of time, the algorithm can also merge two of its existing clusters.

We compare the clustering maintained by the algorithm to the optimal clustering of the current set of points if the set of points was given in advance. The maximum possible ratio of the cost of the algorithm to the cost of the optimal is called its *performance ratio*.

The notion of an incremental clustering algorithm is similar to the recently studied notion of streaming algorithms for clustering problems [12]. We highlight some of the main differences in the two models. A clustering algorithm in the streaming model is not required to maintain an explicit clustering as the data stream is read. Indeed, such an algorithm will be able to output a clustering at any point of time based on the information it stores about the points it has seen so far, but there is no requirement that the clustering corresponding to prefixes of the data stream should evolve via the simple operations listed above. Thus in some sense, an incremental algorithm is more restricted than a streaming algorithm. On the other hand a streaming algorithm has a strict storage space constraint. This is not a restriction we impose on incremental clustering algorithms. Thus the two models are related though strictly speaking, incomparable.

However, the incremental algorithm we devise is also a valid streaming algorithm since it only needs to store a set of points whose size is linear in the number of clusters. Given that the small storage space constraint is met, the algorithm is indeed a streaming algorithm.

Actually, our algorithm does not maintain at most k clusters. We relax the requirement on the number of clusters slightly. Given a parameter k, the algorithm is guaranteed to maintain $c_1 \cdot k$ clusters with total cost $c_2 \cdot \textsc{Opt}$ where $\textsc{Opt}$ is the cost of the optimal solution (with at most k clusters). Here $c_1, c_2 > 0$ are constants.

We first outline the basic ideas and then describe the details of the algorithm. The idea is to incrementally maintain a solution to the fixed costs sum-radii problem. The fixed costs are set proportional to $\textsc{Opt}/k$. If we are able to maintain a constant factor approximation to the fixed costs sum-radii problem with these fixed costs, this will imply a bicriteria approximation for the original sum-radii problem.

The problem of course is that we do not know $\textsc{Opt}$ and that $\textsc{Opt}$ possibly increases as more points are added to the current point set. However, the algorithm maintains a lower bound on the optimal cost $\textsc{Opt}$ and updates this as new points are added. In particular, the algorithm operates in phases. In each phase, the algorithm maintains a lower bound L on the optimal cost and maintains a solution to the fixed costs sum-radii problem with fixed costs $2L/k$. The algorithm maintains a subset $\mathscr{W}$ of the point set seen so far (a *witness set*) together with a dual solution for this set of points.[3] This is a feasible solution to the dual LP for the fixed costs sum-radii problem (8)–(9). As new points are received by the algorithm, some of these are added to the witness set $\mathscr{W}$.

[3] The idea of using a lower bound witness set was also used in [2] for the k-center problem although there the structure of the lower bound witness set was very simple.

Also, the clusters in the solution maintained are updated to cover newly encountered points. When the dual solution is sufficiently large, the algorithm proceeds to the next phase and doubles its lower bound from L to $2L$.

6.1. Algorithm details

At any point of time, the algorithm maintains a lower bound L on the cost of the optimal solution. As stated before, the lower bound is obtained by maintaining a dual solution on an instance consisting of a subset of the points seen so far, i.e. the witness set $\mathscr{W}$. For ease of exposition, it will be convenient to assume that the optimal solution is forced to use clusters with centers at one of the points in $\mathscr{W}$. Of course, the optimal solution for the entire point set need not place its centers at points in $\mathscr{W}$. However, it is easy to see that a solution with arbitrary centers can be converted to a solution for the instance consisting of points in $\mathscr{W}$ with centers only at points in $\mathscr{W}$, by increasing the cost of the solution by a factor of 2. Thus we will assume for the purposes of maintaining the bound, that the optimal solution is forced to use centers in the witness set and analyze the cost of the solution with respect to this bound. However, this is with the implicit understanding that a lower bound of L maintained by the algorithm translates to a *true* lower bound of $L/2$ on the cost of the optimal solution.

Description of a phase: The input to a phase is a lower bound of L and a set of at most $4k$ clusters with sum of radii at most $40L$.

The algorithm treats the cluster centers of the previous phase as input points, processing them in the same way as points added incrementally during the phase. However we remember which points were cluster centers in the previous phase and the radius of the clusters they were centers of. The clusters maintained by the algorithm cover all the points seen during the phase, including the *cluster centers* of the previous phase. Now, in order to cover the *clusters* from the previous phase, the algorithm increases the radii of the clusters covering the *cluster centers*. For the analysis of the solution cost during a phase, we will separate these two contributions. For the most part, we ignore the fact that clusters need to be expanded to cover the clusters of the previous phase. Finally, we incorporate the contribution of this expansion on the cost of the solution maintained during the phase.

The algorithm maintains a set of points $\mathscr{W}$ called a witness set together with a dual solution on these. The dual value α_j for all points $j \in \mathscr{W}$ is L/k. The constraints on the dual solution are:

$$\forall i \in \mathscr{W}, r, \quad \sum_{j : j \in C_i(r)} \alpha_j \leqslant r + 2L/k.$$

A *near-tight* cluster is a cluster $C_i(r)$ for $i \in \mathscr{W}$ such that:

$$\sum_{j : j \in C_i(r)} \alpha_j \geqslant r/2 + L/k.$$

During the phase, the algorithm maintains a set of clusters that cover all *near-tight* clusters.

To do this, the algorithm maintains a subset $\mathscr{C}$ of disjoint *near-tight* clusters, called *core clusters*. For each core cluster $C_i(r) \in \mathscr{C}$, the algorithm uses the cluster $C_i(5r)$ in its solution.

The phase ends as soon as $|\mathscr{W}| = 4k$. At this point, the algorithm begins a new phase with lower bound $2L$.

Phase details: In every phase, we execute the following steps:

```
Procedure SINGLE-PHASE-SUM-RADII (sequence $S$ of points, lower bound $L$)
1. $\mathcal{W} \leftarrow \emptyset$, $\alpha_p = 0$ for all points $p$.
2. Repeat steps 2a–2d until $|\mathcal{W}| = 4k$.
   (a) Let $p$ be the next point in $S$ (remove $p$ from $S$).
   (b) If $p$ lies in a current near-tight cluster, goto step 2a.
   (c) $\mathcal{W} = \mathcal{W} \cup \{p\}$, $\alpha_p = L/k$.
   (d) Call procedure INCREMENTALLY-UPDATE-CLUSTERING to cover all new near-tight
       clusters.
3. Let $C$ be the set of cluster centers in the current solution. Call procedure SINGLE-PHASE-SUM-
   RADII($C \cdot S, 2L$). (Here $C \cdot S$ represents the sequence $C$ concatenated with the unprocessed
   points in the sequence $S$.)
```

Observe that in step 2d, we are guaranteed to cover p since p is contained in at least one *near-tight* cluster as $C_p(0)$ becomes a new *near-tight* cluster when p is added.

Procedure INCREMENTALLY-UPDATE-CLUSTERING

1. Order the set of new *near-tight* clusters in decreasing order of radius and examine each of these in order.
2. Suppose $C_i(r)$ is the current cluster being examined. We consider several cases depending on the *core clusters* in $\mathcal{C}$ that intersect $C_i(r)$.
   (a) *Case* 1: $C_i(r)$ does not intersect any clusters in $\mathcal{C}$.
   In this case, $C_i(r)$ is added as a *core cluster* to $\mathcal{C}$ and the cluster $C_i(5r)$ added to the solution maintained for covering all points seen in the current phase.
   (b) *Case* 2: $C_i(r)$ intersects some cluster $C_{i'}(r') \in \mathcal{C}$ with $r' \geqslant r/2$.
   In this case, $C_{i'}(5r')$ completely contains $C_i(r)$. Since $C_{i'}(5r')$ is in the current solution, $C_i(r)$ is completely covered by the current solution.
   (c) *Case* 3: For all clusters $C_{i'}(r') \in \mathcal{W}$ that intersect $C_i(r)$, $r' < r/2$.
   In this case, $C_i(r)$ is added to $\mathcal{C}$ as a *core cluster* and $C_i(5r)$ is added to the current solution. Further, all clusters $C_{i'}(r')$ that intersect $C_i(r)$ are removed from $\mathcal{C}$ and the corresponding clusters $C_{i'}(5r')$ are removed from the current solution. Effectively, all the removed clusters $C_{i'}(5r')$ are merged together to form the cluster $C_i(5r)$ in this step. Note that, for any removed cluster $C_{i'}(5r')$, since $r' < r/2$ and $C_{i'}(r')$ intersects $C_i(r)$, the old cluster $C_{i'}(5r')$ is completely contained in the newly added cluster $C_i(5r)$. This implies that all *near-tight* clusters previously covered by $C_{i'}(5r')$ are now covered by $C_i(r)$.

Observe that procedure INCREMENTALLY-UPDATE-CLUSTERING ensures that every new *near-tight* cluster $C_i(r)$ is covered by some cluster in the current solution. In particular this ensures that the newly added point p is also covered. Further, the algorithm maintains the invariant that the core clusters in $\mathcal{C}$ are disjoint.

6.2. Analysis

We first prove that the algorithm maintains a valid dual solution during a phase.

Lemma 6.1. *The dual solution maintained by the algorithm satisfies all the dual constraints.*

Proof. We need to verify that no dual constraints are violated when a new point p is added to $\mathcal{W}$. Assume for contradiction, that some dual constraint is violated by the addition of p to $\mathcal{W}$. We consider two cases:

Case 1: The dual constraint for $C_i(r)$ is violated for $i \in \mathcal{W}$, $i \neq p$. In this case, $p \in C_i(r)$ and

$$\sum_{j : j \in C_i(r)} \alpha_j \geqslant r + L/k.$$

(Note that the summation does not include the point p). However this means that $C_i(r)$ was a *near-tight* cluster before the addition of p. In this case, the algorithm would not have added p to $\mathcal{W}$.

Case 2: The dual constraint for $C_p(r)$ is violated. Hence,

$$\sum_{j : j \in C_p(r)} \alpha_j \geqslant r + L/k.$$

(Note that the summation does not include the point p). The cluster $C_p(r)$ must contain at least one point in $\mathcal{W}$ other than p. Consider any such point, say q. The cluster $C_q(2r)$ completely contains the cluster $C_p(r)$. Hence,

$$\sum_{j : j \in C_q(2r)} \geqslant \sum_{j : j \in C_p(r)} \alpha_j \geqslant r + L/k.$$

(Again, the summation does not include the point p). But this implies that the cluster $C_q(2r)$ was *near-tight* before the addition of point p. In this case, the algorithm would not have added p to $\mathcal{W}$.

The contradiction in both cases proves that no dual constraint is violated, proving the lemma. □

Lemma 6.2. *At the end of a phase with lower bound L, the cost of the optimal solution to the sum-radii problem for the points in $\mathcal{W}$, the lower bound witness set, is at least $2L$. (This is with the restriction that cluster centers are in $\mathcal{W}$.)*

Proof. At the end of the phase, $|\mathcal{W}| = 4k$. At this point, the value of the dual solution is $4L$. Suppose the optimal solution for points in $\mathcal{W}$ is strictly less than $2L$. This can be converted to a solution to the sum-radii problem with fixed costs $2L/k$; the cost of this solution is strictly less than $2L + k \cdot 2L/k = 4L$. However, this is strictly less than the lower bound of $4L$ given by the dual solution. The contradiction implies that the cost of the optimal solution to the sum-radii problem is indeed at least $2L$ at the end of the phase. □

Lemma 6.3. *In a phase with lower bound L, the algorithm maintains at most $4k$ clusters with sum of radii at most $40L$ which cover all input points seen during this phase including the cluster centers from the previous phase.*

Proof. For each core cluster $C_i(r) \in \mathscr{C}$, the solution maintained by the algorithm includes cluster $C_i(5r)$. Thus the cost of the solution maintained by the algorithm is

$$5 \sum_{C_i(r) \in \mathscr{C}} r.$$

Note that the core clusters $C_i(r) \in \mathscr{C}$ are disjoint. Further, each core cluster is *near-tight*. Thus,

$$\forall C_i(r) \in \mathscr{C}, \quad r/2 \leqslant \sum_{j \in C_i(r)} \alpha_j,$$

$$\sum_{C_i(r) \in \mathscr{C}} r/2 \leqslant \sum_{C_i(r) \in \mathscr{C}} \sum_{j \in C_i(r)} \alpha_j \leqslant \sum_{j \in \mathscr{W}} \alpha_j \leqslant 4L.$$

Hence the cost of the solution maintained by the algorithm is at most $5 \times 2 \times 4L = 40L$. The number of clusters in the solution is equal to the number of core clusters in $\mathscr{C}$. Since the core clusters are disjoint, each containing at least one point in $\mathscr{W}$, there are at most $|\mathscr{W}| = 4k$ clusters. $\square$

Lemma 6.4. *In a phase with lower bound L, the algorithm maintains at most $4k$ clusters with sum of radii at most $80L$ which cover all input points seen so far.*

Proof. Lemma 6.3 bounds the sum of the radii in the solution maintained by the algorithm in order to cover all the points seen in the current phase, including the cluster centers of the previous phase. For such a cluster center i, let r_i denote the radius of the cluster it was a center for in the previous phase. In order to cover all points seen so far, we need to expand the clusters so as to completely cover the clusters from the previous phase. To accomplish this, each cluster center i from the previous phase is assigned to one of the clusters in the current phase that contains it; further the radius of this cluster is increased by r_i.

Now, the sum of the radii of the clusters in the current solution are increased by (at most) the sum of the radii of the clusters in the previous phase. Thus the sum of radii of clusters in the current solution (to cover all points seen so far) is at most $40L + 40L = 80L$. The number of clusters is at most $4k$ as bounded by Lemma 6.3. $\square$

Lemma 6.5. *At the end of a phase with lower bound L, the algorithm satisfies the conditions required for the next phase with lower bound $2L$.*

Proof. By Lemma 6.2, the algorithm establishes a lower bound of $2L$ at the end of a phase with lower bound L, By Lemma 6.4, the sum of the radii of the clusters maintained by the algorithm is at most $80L = 40 \cdot 2L$ and there are at most $4k$ clusters in all. Thus the algorithm satisfies the conditions for the next phase with lower bound $2L$. $\square$

Note that in a phase with lower bound L, the cost of the optimal solution is at least $L/2$. Using Lemma 6.4, we get the following theorem.

Theorem 6.1. *The algorithm maintains a solution with at most* $4k$ *clusters with cost at most* 160 *times that of the optimal solution.*

For ease of exposition, we have not attempted to optimize the constants in the bicriteria guarantee; our goal rather was to demonstrate that it is possible to obtain an incremental algorithm with such a guarantee. It is possible to improve the constants by using techniques such as randomized doubling (see [2] for example) and better utilizing the lower bound provided by the dual solution.

It is interesting open problem to determine whether it is possible to obtain an incremental clustering algorithm for this problem without allowing an increased number of clusters. Further, it would be interesting to investigate general structural properties of clustering objective functions for which incremental (or streaming) clustering algorithms can be devised.

Acknowledgments

We thank the authors of [9] for sending us their paper, David S. Taylor and the two anonymous referees for their comments that helped vastly improved the presentation. The first author also thanks Deepa Ramakrishna for her support and encouragement.

References

[1] Y. Bartal, M. Charikar, D. Raz, Approximating min-sum k-clustering in metric spaces, Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 11–20, Crete, Greece.
[2] M. Charikar, C. Chekuri, T. Feder, R. Motwani, Incremental clustering and dynamic information retrieval, Proceedings of the 29th Annual ACM Symposium on Theory of Computing 1997, pp. 626–635, El Paso, Texas.
[3] M. Charikar, S. Guha, Improved combinatorial algorithms for the facility location and k-median problems, Proceedings of the 40th Annual Symposium on Foundations of Computer Science, 1999, pp. 378–388, New York, New York.
[4] M. Charikar, S. Guha, E. Tardos, D. Shmoys, A constant factor approximation algorithm for the k-median problem, Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 1–10, Atlanta, Georgia.
[5] F. Chudak, Improved approximation algorithms for uncapacitated facility location, Proceedings of the Sixth Conference on Integer Programming and Optimization, Lecture Notes in Computer Science, Vol. 1412, Springer, Berlin, 1998, pp. 180–194.
[6] F.A. Chudak, T. Roughgarden, D.P. Williamson, Approximate k-MSTs and k-Steiner trees via the primal–dual method and Lagrangean relaxation, Proceedings of the Eighth Conference on Integer Programming and Optimization, Lecture Notes in Computer Science, Vol. 2081, Springer, Berlin, 2001, pp. 60–70.
[7] F.A. Chudak, D. Shmoys, Improved approximation algorithms for the capacitated facility location problem, Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. S875–S876, Baltimore, Maryland.

[8] G. Cornuéjols, G.L. Nemhauser, L.A. Wolsey, The uncapacitated facility location problem, in: P.B. Mirchandani, R.L. Francis (Eds.), Discrete Location Theory, Wiley, New York, 1990.

[9] S.R. Doddi, M.V. Marathe, S.S. Ravi, D.S. Taylor, P. Widmayer, Approximation algorithms for clustering to minimize the sum of diameters, Nordic J. Comput. 7 (3) (2000) 185–203.

[10] M. Dyer, A.M. Frieze, A simple heuristic for the p-center problem, Oper. Res. Lett. 3 (1985) 285–288.

[11] S. Guha, S. Khuller, Greedy strikes back: improved facility location algorithms, Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 649–657.

[12] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams, Proceedings of the 21st Annual Symposium on Foundations of Computer Science, 2000, pp. 359–366, Redondo Beach, California.

[13] N. Guttman-Beck, R. Hassin, Approximation algorithms for min-sum p-clustering, Discrete Appl. Math. 89 (1998) 125–142.

[14] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, Math. Programming 79 (1997) 191–215.

[15] D.S. Hochbaum, D.B. Shmoys, A best possible approximation algorithm for the k-center problem, Math. Oper. Res. 10 (1985) 180–184.

[16] K. Jain, V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal–dual scheme and Lagrangian relaxation, Journal of the ACM 48 (2001) 274–296.

[17] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems, Part ii: p-medians, SIAM J. Appl. Math. 37 (1979) 539–560.

[18] M. Korupolu, C.G. Plaxton, R. Rajaraman, Analysis of a local search heuristic for facility location problems, Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 1–10, San Francisco, California.

[19] J.-H. Lin, J.S. Vitter, Approximation algorithms for geometric median problems, Inform. Process. Lett. 44 (1992) 245–249.

[20] C.L. Monma, S. Suri, Partitioning points and graphs to minimize the maximum or the sum of diameters, in: Alavi et al. (Eds.), Graph Theory, Combinatorics and Applications, Wiley, 1991, pp. 880–912.

[21] D.B. Shmoys, É. Tardos, K.I. Aardal, Approximation algorithms for facility location problems, Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 265–274, El Paso, Texas.