Online Sum-Radii Clustering [⋆]

Dimitris Fotakis[1] and Paraschos Koutris[2]

[1] School of Electrical and Computer Engineering,
National Technical University of Athens, 157 80 Athens, Greece.
fotakis@cs.ntua.gr
[2] Computer Science and Engineering, University of Washington, U.S.A.
pkoutris@cs.washington.edu

Abstract. In Online Sum-Radii Clustering, n demand points arrive online and must be irrevocably assigned to a cluster upon arrival. The cost of each cluster is the sum of a fixed opening cost and its radius, and the objective is to minimize the total cost of the clusters opened by the algorithm. We show that the deterministic competitive ratio of Online Sum-Radii Clustering for general metric spaces is $\Theta(\log n)$, where the upper bound follows from a primal-dual online algorithm, and the lower bound is valid for ternary Hierarchically Well-Separated Trees (HSTs) and for the Euclidean plane. Combined with the results of (Csirik et al., MFCS 2010), this result demonstrates that the deterministic competitive ratio of Online Sum-Radii Clustering changes abruptly, from constant to logarithmic, when we move from the line to the plane. We also show that Online Sum-Radii Clustering in HSTs is closely related to the Parking Permit problem introduced by (Meyerson, FOCS 2005). Exploiting the relation to Parking Permit, we obtain a lower bound of $\Omega(\log \log n)$ on the randomized competitive ratio of Online Sum-Radii Clustering in tree metrics. Moreover, we present a simple randomized $O(\log n)$-competitive algorithm, and a deterministic $O(\log \log n)$-competitive algorithm for the fractional version of the problem.

1 Introduction

In clustering problems, we seek a partitioning of n demand points into k groups, or *clusters*, so that a given objective function, that depends on the distance between points in the same cluster, is minimized. Typical examples are the k-Center problem, where we minimize the maximum cluster diameter, the Sum-k-Radii problem, where we minimize the sum of cluster radii, and the k-Median problem, where we minimize the total distance of points to the nearest cluster center. These are fundamental problems in Computer Science, with many important applications, and have been extensively studied from an algorithmic viewpoint (see e.g., [18] and the references therein).

In this work, we study an online clustering problem closely related to Sum-k-Radii. In the online setting, the demand points arrive one-by-one and must be irrevocably assigned to a cluster upon arrival. We require that once formed, clusters cannot be merged,

[⋆] This work was supported by the project AlgoNow, co-financed by the European Union (European Social Fund - ESF) and Greek national funds, through the Operational Program "Education and Lifelong Learning", under the research funding program Thales.

split, or have their center or radius changed. The goal is to open a few clusters with a small sum of radii. However, instead of requiring that at most k clusters open, which would lead to an unbounded competitive ratio, we follow [6, 7] and consider a Facility-Location-like relaxation of Sum-k-Radii, called *Sum-Radii Clustering*. In Sum-Radii Clustering, the cost of each cluster is the sum of a fixed opening cost and its radius, and we seek to minimize the total cost of the clusters opened by the algorithm.

In addition to clustering and data analysis, Sum-Radii Clustering has applications to location problems of wireless base stations, such as sensors [7, 8] or antennas [3, 15]. In such problems, we place some wireless base stations and setup their communication range so that some communication demands are satisfied and the total setup and operational cost is minimized. A standard assumption is that the setup cost is proportional to the number of stations installed, and the operational cost for each station is proportional to its range (or a low-degree polynomial of it).

Related Work. In the offline setting, Sum-k-Radii and the closely related problem of Sum-k-Diameters[3] have been thoroughly studied. Sum-k-Radii is $\mathbf{NP}$-hard even in metric spaces of constant doubling dimension [14]. Gibson et al. [13] proved that Sum-k-Radii in Euclidean spaces of constant dimension is polynomially solvable, and presented an $O(n^{\log \Delta \, \log n})$-time algorithm for Sum-$k$-Radii in general metric spaces, where Δ is the diameter [14]. As for approximation algorithms, Doddi et al. [9] proved that it is $\mathbf{NP}$-hard to approximate Sum-k-Diameters in general metric spaces within a factor less than 2, and gave a bicriteria algorithm that achieves a logarithmic approximation using $O(k)$ clusters. Subsequently, Charikar and Panigraphy [6] presented a primal-dual $(3.504 + \varepsilon)$-approximation algorithm for Sum-k-Radii in general metric spaces, which uses as a building block a primal-dual 3-approximation algorithm for Sum-Radii Clustering. Biló et al. [3] considered a generalization of Sum-k-Radii, where the cost is the sum of the α-th power of the clusters radii, for $\alpha \geq 1$, and presented a polynomial-time approximation scheme for Euclidean spaces of constant dimension.

Charikar and Panigraphy [6] also considered the incremental version of Sum-k-Radii, Similarly to the online setting, an incremental algorithm processes the demands one-by-one and assigns them to a cluster upon arrival. However, an incremental algorithm can also merge any of its clusters at any time. They presented an $O(1)$-competitive incremental algorithm for Sum-k-Radii that uses $O(k)$ clusters.

In the online setting, where cluster reconfiguration is not allowed, the Unit Covering and the Unit Clustering problems have received considerable attention. In both problems, the demand points arrive one-by-one and must be irrevocably assigned to unit-radius balls upon arrival, so that the number of balls used is minimized. The difference is that in Unit Covering, the center of each ball is fixed when the ball is first used, while in Unit Clustering, there is no fixed center and a ball may shift and cover more demands. Charikar et al. [5] proved an upper bound of $O(2^d d \log d)$ and a lower bound of $\Omega(\log d / \log\log\log d)$ on the deterministic competitive ratio of Unit Covering in d dimensions. The results of [5] imply a competitive ratio of 2 and 4 for Unit Covering on the line and the plane, respectively. The Unit Clustering problem was introduced by Chan and Zarrabi-Zadeh [4]. The deterministic competitive ratio of Unit Clustering on

[3] These problems are closely related in the sense that a c-competitive algorithm for Sum-k-Radii implies a $2c$-competitive algorithm for Sum-k-Diameters, and vice versa.

the line is at most $5/3$ [10] and no less than $8/5$ [11]. Unit Clustering has also been studied in d-dimensions with respect to the L_∞ norm, where the competitive ratio is at most $\frac{5}{6}2^d$, for any d, and no less than $13/6$, for $d \geq 2$ [10].

Departing from this line of work, Csirik at el. [7] studied online clustering to minimize the sum of the setup costs and the diameters of the clusters (CSDF). Motivated by the difference between Unit Covering and Unit Clustering, they considered three models, the strict, the intermediate, and the flexible one, depending on whether the center and the radius of a new cluster are fixed at its opening time. Csirik at el. only studied CSDF on the line and proved that its deterministic competitive ratio is $1 + \sqrt{2}$ for the strict and the intermediate model and $(1 + \sqrt{5})/2$ for the flexible model. Recently, Divéki and Imreh [8] studied online clustering in two dimensions to minimize the sum of the setup costs and the area of the clusters. They proved that the competitive ratio of this problem lies in $(2.22, 9]$ for the strict model and in $(1.56, 7]$ for the flexible model.

Contribution. Following [7], it is natural and interesting to study the online clustering problem of CSDF in metric spaces more general than the line. In this work, we consider the closely related problem of Online Sum-Radii Clustering (OnlSumRad), and give upper and lower bounds on its deterministic and randomized competitive ratio for general metric spaces and for the Euclidean plane. We restrict our attention to the strict model of [7], where the center and the radius of each new cluster are fixed at opening time. To justify our choice, we show that a c-competitive algorithm for the strict model implies an $O(c)$-competitive algorithm for the intermediate and the flexible model.

We show that the deterministic competitive ratio of OnlSumRad for general metric spaces is $\Theta(\log n)$, where the upper bound follows from a primal-dual algorithm, and the lower bound is valid for ternary Hierarchically Well-Separated Trees (HSTs) and for the Euclidean plane. This result is particularly interesting because it demonstrates that the deterministic competitive ratio of OnlSumRad (and of CSDF) changes abruptly, from constant to logarithmic, when we move from the line to the plane. Interestingly, this does not happen when the cost of each cluster is proportional to its area [8].

Another interesting finding is that OnlSumRad in metric spaces induced by HTSs is closely related to the Parking Permit problem introduced by Meyerson [17]. In Parking Permit, we cover a set of driving days by choosing among K permit types, each with a given cost and duration. The permit costs are concave, in the sense that the cost per day decreases with the duration. The algorithm is informed of the driving days in an online fashion, and irrevocably decides on the permits to purchase, so that all driving days are covered by a permit and the total cost is minimized. Meyerson [17] proved that the competitive ratio of Parking Permit is $\Theta(K)$ for deterministic and $\Theta(\log K)$ for randomized algorithms. We prove that OnlSumRad in HSTs is a generalization of Parking Permit. Combined with the randomized lower bound of [17], this implies a lower bound of $\Omega(\log \log n)$ on the randomized competitive ratio of OnlSumRad. Moreover, we show that, under some assumptions, a c-competitive algorithm for Parking Permit with K types implies a c-competitive algorithm for OnlSumRad in HSTs with K levels.

We conclude with a simple and memoryless randomized $O(\log n)$-competitive algorithm, and a deterministic $O(\log \log n)$-competitive algorithm for the fractional version of OnlSumRad. Both algorithms work for general metric spaces. The fractional algorithm is based on the primal-dual approach of [2, 1], and generalizes the fractional algorithm of [17] for Parking Permit. We leave as an open problem the existence of a randomized rounding procedure that converts the fractional solution to an (integral) clustering of cost within a constant factor of the original cost. This would imply a randomized $O(\log \log n)$-competitive algorithm for OnlSumRad in general metrics.

Other Related Work. OnlSumRad is a special case of Online Set Cover [2] with sets of different weight. In [2], it is presented a nearly optimal deterministic $O(\log m \log N)$-competitive algorithm, where N is the number of elements and m is the number of sets. Moreover, if all sets have the same weight and each element belongs to at most d sets, the competitive ratio can be improved to $O(\log d \log N)$. If we cast OnlSumRad as a special case of Online Set Cover, N is the number of points in the metric space, which can be much larger than the number of demands n, $m = \Omega(n)$, and $d = O(\log n)$. Hence, a direct application of the algorithm of [2] to OnlSumRad does not lead to an optimal deterministic competitive ratio. This holds even if one could possibly extend the improved ratio of $O(\log d \log N)$ to the weighted set structure of OnlSumRad.

At the conceptual level, OnlSumRad is related to the problem of Online Facility Location [16, 12]. However, the two problems exhibit a different behavior w.r.t. their competitive ratio, since the competitive ratio of Online Facility Location is $\Theta(\frac{\log n}{\log \log n})$, even on the line, for both deterministic and randomized algorithms [12].

2 Notation, Problem Definition, and Preliminaries

We consider a metric space (M, d), where M is the set of points and $d : M \times M \mapsto \mathbb{N}$ is the distance function, which is non-negative, symmetric and satisfies the triangle inequality. For a set of points $M' \subseteq M$, we let $\mathrm{diam}(M') \equiv \max_{u,v \in M'}\{d(u, v)\}$ be the diameter and $\mathrm{rad}(M') \equiv \min_{u \in M'} \max_{v \in M'}\{d(u, v)\}$ be the radius of M'. In a *tree metric*, the points correspond to the nodes of an edge-weighted tree and the distances are given by the tree's shortest path metric. For some $\alpha > 1$, a *Hierarchically α-Well-Separated Tree* (α-HST) is a complete rooted tree with lengths on its edges such that: (i) the distance of each leaf to its parent is 1, and (ii) on every path from a leaf to the root, the edge length increases by a factor of α on every level. Thus, the distance of any node v_k at level k to its children is α^{k-1} and the distance of v_k to the nearest leaf is $(\alpha^k - 1)/(\alpha - 1)$. We usually identify a tree with the metric space induced by it.

A *cluster* $C(p, r) \equiv \{v : d(p, v) \leq r\}$ is determined by its center p and its radius r, and consists of all points within a distance at most r to p. The cost of a cluster $C(p, r)$ is the sum of its opening cost f and its radius r.

Sum-Radii Clustering. In the offline version of Sum-Radii Clustering, we are given a metric space (M, d), a cluster opening cost f, and a set $D = \{u_1, \dots, u_n\}$ of demand points in M. The goal is to find a collection of clusters $C(p_1, r_1), \dots, C(p_k, r_k)$ that cover all demand points in D and minimize the total cost, which is $\sum_{i=1}^{k}(f + r_i)$.

Online Sum-Radii Clustering. In the online setting, the demand points arrive one-by-one, in an online fashion, and must be irrevocably assigned to an open cluster upon arrival. Formally, the input to Online Sum-Radii Clustering (OnlSumRad) consists of the cluster opening cost f and a sequence $u_1, \dots, u_n$ of (not necessarily distinct) demand points in an underlying metric space (M, d). The goal is to maintain a set of clusters of minimum total cost that cover all demand points revealed so far.

In this work, we focus on the so-called Fixed-Cluster version of OnlSumRad, where the center and the radius of each new cluster are irrevocably fixed when the cluster opens. Thus, the online algorithm maintains a collection of clusters, which is initially empty. Upon arrival of a new demand u_j, if u_j is not covered by an open cluster, the algorithm opens a new cluster $C(p, r)$ that includes u_j, and assigns u_j to it. The algorithm incurs an irrevocable cost of $f + r$ for the new cluster $C(p, r)$.

Competitive Ratio. We evaluate the performance of online algorithms using *competitive analysis*. A (randomized) algorithm is c-competitive if for any sequence of demand points, its (expected) cost is at most c times the cost of the optimal solution for the corresponding offline Sum-Radii instance. The (expected) cost of the algorithm is compared against the cost of an optimal offline algorithm that is aware of the entire demand sequence in advance and has no computational restrictions whatsoever.

Simplified Optimal. The following proposition simplifies the structure of the optimal solution in the competitive analysis.

Proposition 1. *Let S be a feasible solution of an instance $\mathcal{I}$ of OnlSumRad. Then, there is a feasible solution S' of $\mathcal{I}$ with a cost of at most twice the cost of S, where each cluster has a radius of $2^k f$, for some integer $k \geq 0$.*

Other Versions of OnlSumRad. For completeness, we discuss two seemingly less restricted versions of OnlSumRad, corresponding to the intermediate and the flexible model in [7]. In both versions, the demands are irrevocably assigned to a cluster upon arrival. In the Fixed-Radius version, only the radius of a new cluster is fixed when the cluster opens. The algorithms incurs an irrevocable cost of $f + r$ for each new cluster C of radius r. Then, new demands can be assigned to C, provided that $\mathrm{rad}(C) \leq r$. In the Flexible-Cluster version, a cluster C is a set of demands with neither a fixed center nor a fixed radius. The algorithm's cost for each cluster C is $f + \mathrm{rad}(C)$, where $\mathrm{rad}(C)$ may increase as new demands are added to C. The Fixed-Cluster version is a restriction of the Fixed-Radius version, which, in turn, is a restriction of the Flexible-Cluster version. The following proposition shows that from the viewpoint of competitive analysis, the three versions are essentially equivalent.

Proposition 2. *If there exists a c-competitive algorithm for the Fixed-Radius (resp. the Flexible-Cluster) version, then there exists a $2c$-competitive (resp. $10c$-competitive) algorithm for the Fixed-Cluster version.*

Parking Permit. In Parking Permit (ParkPermit), we are given a schedule of days, some of which are marked as driving days, and K types of permits, where a permit of each type k, $k = 1, \dots, K$, has cost c_k and duration d_k. The goal is to purchase a set of permits of minimum total cost that cover all driving days. In the online setting, the driving days are presented one-by-one, and the algorithm irrevocably decides on the permits to purchase based on the driving days revealed so far.

Meyerson [17] observed that by losing a constant factor in the competitive ratio, we can focus on the *interval version* of ParkPermit, where each permit is available over specific time intervals (e.g. a weakly permit is valid from Monday to Sunday). Moreover, every day is covered by a single permit of each type k, and each permit

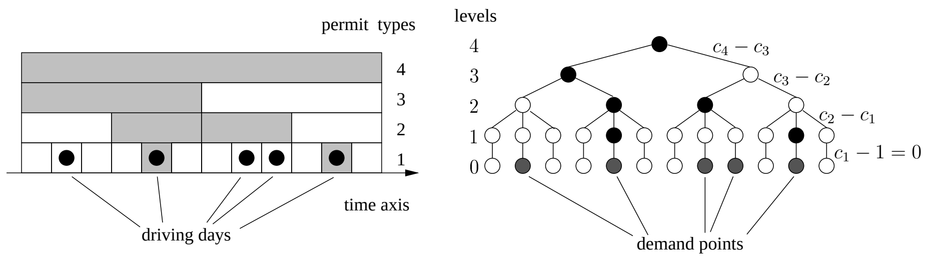

Fig. 1. An example of the reduction of Theorem 1. On the left, there is an instance of the interval version of ParkPermit. A feasible solution consists of the permits in grey. On the right, we depict the instance of OnlSumRad constructed in the proof of Theorem 1.

of type $k \geq 2$ has d_k/d_{k-1} permits of type $k-1$ embedded in it (see also Fig. 1). An interesting feature of the deterministic algorithm in [17] is that it is *time-sequence-independent*, in the sense that it applies, with the same competitive ratio of $O(K)$, even if the order in which the driving days are revealed may not be their time order (e.g. the adversary may mark Aug. 6 as a driving day, before marking May 25 as a driving day).

3 Online Sum-Radii Clustering and Parking Permit

In this section, we show that OnlSumRad in tree metrics and the interval version of ParkPermit are essentially equivalent problems. Our results either are directly based on this correspondence or exploit this correspondence so that they draw ideas from ParkPermit. We start with the following theorem, which shows that OnlSumRad in tree metrics is a generalization of the interval version of ParkPermit.

Theorem 1. *A c-competitive algorithm for Online Sum-Radii Clustering in HSTs with $K+1$ levels implies a c-competitive algorithm for the interval version of Parking Permit with K permit types.*

Proof. Given an instance $\mathcal{I}$ of the interval version of ParkPermit with K permit types, we construct an instance $\mathcal{I}'$ of OnlSumRad in an HST with $K+1$ levels such that any feasible solution of $\mathcal{I}$ is mapped, in an online fashion, to a feasible solution of $\mathcal{I}'$ of equal cost, and vice versa. Let $\mathcal{I}$ be an instance of the interval version of ParkPermit with K types of costs $c_1, \dots, c_K$ and durations $d_1, \dots, d_K$. Wlog., we assume that $c_1 = 1$ and that all days are covered by the permit of type K. Given the costs and the durations of the permits, we construct a tree T with appropriate edge lengths, which gives the metric space for $\mathcal{I}'$. The construction exploits the tree-like structure of the interval version (see also Fig. 1). Specifically, the tree T has $K+1$ levels, where the leaves correspond to the days of $\mathcal{I}$'s schedule, and each node at level k, $1 \leq k \leq K$, corresponds to a permit of type k.

Formally, the tree T has a leaf, at level 0, for each day in the schedule of $\mathcal{I}$. For each interval D_1 of d_1 days covered by a permit of type 1, there is a level-1 node v_1 in T whose children are the d_1 leaves corresponding to the days in D_1. The distance of each level-1 node to its children is $c_1 - 1 = 0$. Hence, opening a cluster $C(v_1, c_1 - 1)$ covers all leaves corresponding to the days in D_1. Similarly, for each interval D_k of d_k days covered by a permit of type k, $2 \leq k \leq K$, there is a node v_k at level k in T whose children are the d_k/d_{k-1} nodes at level $k-1$ corresponding to the permits of type $k-1$ embedded within the particular permit of type k. The distance of each level-k node to its children is $c_k - c_{k-1}$. Therefore, opening a cluster $C(v_k, c_k - 1)$ covers all leaves corresponding to the days in D_k. The cluster opening cost is $f = 1$. For each driving day t in $\mathcal{I}$, there is, in $\mathcal{I}'$, a demand located at the leaf of T corresponding to t.

Based on the correspondence between a type-k permit and a cluster $C(v_k, c_k - 1)$ rooted at a level-k node v_k, we can show that any solution of $\mathcal{I}$ is mapped, in an online fashion, to a solution of $\mathcal{I}'$ of equal cost, and vice versa. □

In the proof of Theorem 1, if the ParkPermit instance has $d_1 = 1$ and $c_k = 2^k$, for each type k, the tree T is essentially a 2-HST with K levels where all nodes at the same level k have d_k/d_{k-1} children. Thus, combined with Theorem 1, the following lemma shows that OnlSumRad in such tree metrics is similar to the interval version of ParkPermit. The proof of Lemma 1 applies the reverse reduction of Theorem 1.

Lemma 1. *A c-competitive time-sequence-independent algorithm for the interval version of ParkPermit with K permits implies a c-competitive algorithm for OnlSumRad in HSTs with K levels, where all nodes at the same level have the same number of children and all demands are located at the leaves.*

4 Lower Bounds on the Competitive Ratio

By Theorem 1, OnlSumRad in trees with $K+1$ levels is a generalization of ParkPermit with K permit types. Therefore, the results of [17] imply a lower bound of $\Omega(K)$ (resp. $\Omega(\log K)$) on the deterministic (resp. randomized) competitive ratio of OnlSumRad in trees with K levels. However, a lower bound on the competitive ratio of OnlSumRad would rather be expressed in terms of the number of demands n, because there is no simple and natural way of defining the number of "levels" of a general metric space, and because for online clustering problems, the competitive ratio, if not constant, is typically stated as a function of n.

Going through the proofs of Theorem 1 and of [17, Theorems 3.2 and 4.6], we can translate the lower bounds on the competitive ratio of ParkPermit, expressed as a function of K, into equivalent lower lower bounds for OnlSumRad, expressed as a function of n. In fact, the proofs of [17, Theorems 3.2 and 4.6] require that the ratio d_k/d_{k-1} of the number of days covered by permits of type k and $k-1$ is $2K$. Thus, in the proof of Theorem 1, the tree T has $(2K)^K$ leaves, and the number of demands n is at most $(2K)^K$. Combining this with the lower bound of $\Omega(\log K)$ on the randomized competitive ratio of ParkPermit [17, Theorem 4.6], we obtain the following corollary:

Corollary 1. *The competitive ratio of any randomized algorithm for Online Sum-Radii Clustering in tree metrics is $\Omega(\log\log n)$, where n is the number of demands.*

A Stronger Lower Bound on the Deterministic Competitive Ratio. This approach gives a lower bound of $\Omega(\frac{\log n}{\log\log n})$ on the deterministic competitive ratio of Online Sum-Radii Clustering. Using a ternary HST, we next obtain a stronger lower bound.

Theorem 2. *The competitive ratio of any deterministic online algorithm for Online Sum-Radii Clustering in tree metrics is $\Omega(\log n)$, where n is the number of demands.*

Proof. For simplicity, let us assume that n is an integral power of 3. For some constant $\alpha \in [2, 3)$, we consider an α-HST T of height $K = \log_3 n$ whose non-leaf nodes have 3 children each. The cluster opening cost is $f = 1$. Let A be any deterministic algorithm. We consider a sequence of demands located at the leaves of T. More precisely, starting from the leftmost leaf and advancing towards the rightmost leaf, the next demand in the sequence is located at the next leaf not covered by an open cluster of A. Since T has n leaves, A may cover all leaves of T before the arrival of n demands. Then, the demand sequence is completed in an arbitrary way that does not increase the optimal cost. We let C_{OPT} be the optimal cost, and let C_A be the cost of A on this demand sequence.

We let $c_k = 1 + \sum_{\ell=0}^{k-1} \alpha^\ell$ denote the cost of a cluster centered at a level-k node v_k with radius equal to the distance of v_k to the nearest leaf. We observe that for any $k \geq 1$ and any $\alpha \geq 2$, $c_k \leq \alpha c_{k-1}$. We classify the clusters opened by A according to their cost. Specifically, we let L_k, $0 \leq k \leq K$, be the set of A's clusters with cost in $[c_k, c_{k+1})$, and let $\ell_k = |L_k|$ be the number of such clusters. The key property is that a cluster in L_k can cover the demands of a subtree rooted at level at most k, but not higher. Therefore, we can assume that all A's clusters in L_k are centered at a level-k node and have cost equal to c_k, and obtain a lower bound of $C_A \geq \sum_{k=0}^{K} \ell_k c_k$ on the algorithm's cost.

To derive an upper bound on the optimal cost in terms of C_A, we distinguish between good and bad active subtrees, depending on the size of the largest radius cluster with which A covers the demand points in them. Formally, a subtree T_k rooted at level k is *active* if there is a demand point located at some leaf of it. For an active subtree T_k, we let $C_{T_k}^{\max}$ denote the largest radius cluster opened by A when a new demand point in T_k arrives. Let j, $0 \leq j \leq K$, be such that $C_{T_k}^{\max} \in L_j$. Namely, $C_{T_k}^{\max}$ is centered at a level-j node v_j and covers the entire subtree rooted at v_j. If $j \geq k$, i.e. if $C_{T_k}^{\max}$ covers T_k entirely, we say that T_k is a *good* (active) subtree (for the algorithm A). If $j < k$, i.e. if $C_{T_k}^{\max}$ does not cover T_k entirely, we say that T_k is a *bad* (active) subtree (for A).

For each $k = 0, \ldots, K$, we let g_k (resp. b_k) denote the number of good (resp. bad) active subtrees rooted at level k. To bound g_k from above, we observe that the last demand point of each good active subtree rooted at level k is covered by a new cluster of A rooted at a level $j \geq k$. Therefore, the number of good active subtrees rooted at level k is at most the number of clusters in $\bigcup_{j=k}^{K} L_j$. Formally, for each level $k \geq 0$, $g_k \leq \sum_{j=k}^{K} \ell_j$. To bound b_k from above, we first observe that each active leaf / demand point is a good active level-0 subtree, and thus $b_0 = 0$. For each level $k \geq 1$, we observe that if T_k is a bad subtree, then by the definition of the demand sequence, the 3 subtrees rooted at the children of T_k's root are all active. Moreover, each of these subtrees is either a bad subtree rooted at level $k - 1$, in which case it is counted in b_{k-1}, or a good subtree covered by a cluster in L_{k-1}, in which case it is counted in ℓ_{k-1}. Therefore, for each level $k \geq 1$, $3b_k \leq b_{k-1} + \ell_{k-1}$.

Using these bounds on g_k and b_k, we can bound from above the optimal cost in terms of C_A. To this end, the crucial observation is that we can obtain a feasible solution by opening a cluster of cost c_k centered at the root of every active subtree rooted at level k. Since the number of active subtrees rooted at level k is $b_k + g_k$, we obtain that for every $k \geq 0$, $C_{OPT} \leq c_k(b_k + g_k)$. Using the upper bound on g_k and summing up for $k = 0, \dots, K$, we have that $(K+1)C_{OPT} \leq \sum_{k=0}^{K} c_k b_k + \sum_{k=0}^{K} c_k \sum_{j=k}^{K} \ell_j$.

Using that $c_k \leq \alpha^k$ and that $c_k \leq \alpha c_{k-1}$, which hold for all $\alpha \geq 2$, we bound the second term by:

$$\sum_{k=0}^{K} c_k \sum_{j=k}^{K} \ell_j = \sum_{k=0}^{K} \ell_k \sum_{j=0}^{k} c_j \leq \sum_{k=0}^{K} \ell_k \sum_{j=0}^{k} \alpha^j \leq \sum_{k=0}^{K} \ell_k c_{k+1} \leq \alpha \sum_{k=0}^{K} \ell_k c_k \leq \alpha C_A$$

To bound the first term, we use that for every level $k \geq 1$, $3b_k \leq b_{k-1} + \ell_{k-1}$ and $c_k \leq \alpha c_{k-1}$. Therefore, $(3/\alpha) b_k c_k \leq (b_{k-1} + \ell_{k-1}) c_{k-1}$. Summing up for $k = 1, \dots, K$, we have that:

$$\frac{3}{\alpha} \sum_{k=1}^{K} b_k c_k \leq \sum_{k=1}^{K} b_{k-1} c_{k-1} + \sum_{k=1}^{K} \ell_{k-1} c_{k-1}$$

Using that $b_0 = 0$ and that $\alpha < 3$, we obtain that:

$$\frac{3}{\alpha} \sum_{k=0}^{K} b_k c_k \leq \sum_{k=0}^{K-1} b_k c_k + \sum_{k=0}^{K-1} \ell_k c_k \leq \sum_{k=0}^{K} b_k c_k + C_A \quad \Rightarrow \quad \sum_{k=0}^{K} b_k c_k \leq \frac{\alpha}{3-\alpha} C_A$$

Putting everything together, we conclude that for any $\alpha \in [2, 3)$, $(K+1)C_{OPT} \leq (\alpha + \frac{\alpha}{3-\alpha}) C_A$. Since $K = \log_3 n$, this implies the theorem. □

A Lower Bound for Deterministic OnlSumRad on the Plane. Motivated by the fact that the deterministic competitive ratio of OnlSumRad on the line is constant [7], we study OnlSumRad in the Euclidean plane. The following theorem uses a constant-distortion planar embedding of a ternary α-HST, and establishes a lower bound of $\Omega(\log n)$ on the deterministic competitive ratio of OnlSumRad on the Euclidean plane.

Theorem 3. *The competitive ratio of any deterministic algorithm for Online Sum-Radii Clustering on the Euclidean plane is $\Omega(\log n)$, where n is the number of demands.*

Proof sketch. Using a planar embedding of a ternary α-HST T with distortion $D_\alpha \leq \sqrt{2}\,\alpha/(\alpha - 2)$, we can show that a c-competitive algorithm for OnlSumRad on the plane implies a $2cD_\alpha$-competitive algorithm for HSTs. □

5 An Optimal Primal-Dual Online Algorithm

Next, we present a deterministic primal-dual algorithm for OnlSumRad in a general metric space (M, d). In the following, we assume that the optimal solution only consists of clusters with radius $2^k f$, where k is a non-negative integer (see also Proposition 1). For simplicity, we let $r_k = 2^k f$, if $k \geq 0$, and $r_k = 0$, if $k = -1$. Let $N = \mathbb{N} \cup \{-1\}$. Then, the following are a Linear Programming relaxation of OnlSumRad and its dual:

$$\begin{array}{lll} \min & \displaystyle\sum_{(z,k) \in M \times N} x_{zk}(f + r_k) & \\ \text{s.t} & \displaystyle\sum_{(z,k): d(u_j, z) \leq r_k} x_{zk} \geq 1 & \forall\, u_j \\ & x_{zk} \geq 0 & \forall\, (z,k) \end{array} \qquad \begin{array}{lll} \max & \displaystyle\sum_{j=1}^{n} a_j & \\ \text{s.t} & \displaystyle\sum_{j: d(u_j, z) \leq r_k} a_j \leq f + r_k & \forall\, (z,k) \\ & a_j \geq 0 & \forall\, u_j \end{array}$$

In the primal program, there is a variable x_{zk} for each point z and each $k \in N$ that indicates the extent to which cluster $C(z, r_k)$ is open. The constraints require that each demand u_j is fractionally covered. If we require that $x_{zk} \in \{0, 1\}$ for all z, k, we obtain an Integer Programming formulation of OnlSumRad. In the dual, there is a variable a_j for each demand u_j, and the constraints require that no potential cluster is "overpaid".

The primal-dual algorithm, or PD-SumRad in short, maintains a collection of clusters that cover all the demands processed so far. When a new demand u_j, $j = 1, \dots, n$, arrives, if u_j is covered by an already open cluster C, PD-SumRad assigns u_j to C and sets u_j's dual variable a_j to 0. Otherwise, PD-SumRad sets a_j to f. This makes the dual constraint corresponding to $(u_j, -1)$ and possibly some other dual constraints tight. PD-SumRad finds the maximum $k \in N$ such that for some point $z \in M$, the dual constraint corresponding to (z, k) becomes tight due to a_j. Then, PD-SumRad opens a new cluster $C(z, 3r_k)$ and assigns u_j to it. The main result of this section is that:

Theorem 4. *The competitive ratio of PD-SumRad is* $\Theta(\log n)$.

The lower bound on the competitive ratio of PD-SumRad follows from Theorem 2. To establish the upper bound, we first observe that the dual solution maintained by PD-SumRad is feasible. Thus the optimal cost for any demand sequence is at least the value of the dual solution maintained by PD-SumRad. Combining this observation with the following lemma, which shows that the total cost of PD-SumRad is at most $O(\log n)$ times the value of its dual solution, we obtain the claimed competitive ratio.

Lemma 2. *The cost of PD-SumRad is at most* $3(2 + \log_2 n) \sum_{j=1}^{n} a_j$.

Proof. We call a cluster $C(z, r_k)$ *tight* if the dual constraint corresponding to (z, k) is satisfied with equality. We observe that for any integer $k > \log_2 n$ and for all points z, $C(z, k)$ cannot become tight, because the lefthand-side of any dual constraint is at most nf. Therefore, we can restrict our attention to at most $2 + \log_2 n$ values of k.

Next, we show that for every $k = -1, 0, \dots, \lfloor \log_2 n \rfloor$, each demand u_j with $a_j > 0$ contributes to the opening cost of at most one cluster with radius $3r_k$. Namely, there is at most one cluster $C(z, 3r_k)$ for which u_j belongs to the tight cluster $C(z, r_k)$. For sake of contradiction, let us assume that for some value of k, PD-SumRad opens two clusters $C_1 = C(z_1, 3r_k)$ and $C_2 = C(z_2, 3r_k)$ for which there is some u_j with $a_j > 0$ that belongs to both $C(z_1, r_k)$ and $C(z_2, r_k)$. Since PD-SumRad opens at most one new cluster when a new demand is processed, one of the clusters C_1, C_2 opens before the other. So, let us assume that C_1 opens before C_2. This means that PD-SumRad opened C_1 in response to a demand $u_{j'}$, with $j' \leq j$, that was uncovered at its arrival time and made $C(z_1, r_k)$ tight. Then, any subsequent demand $u \in C(z_2, r_k)$ is covered by C_1, because:

$$d(u, z_1) \leq d(u, u_j) + d(u_j, z_1) \leq 2r_k + r_k = 3r_k$$

The second inequality above holds because both u and u_j belong to $C(z_2, r_k)$ and u_j also belongs to $C(z_1, r_k)$. Therefore, after C_1 opens, there are no uncovered demands in $C(z_2, r_k)$ that can force PD-SumRad to open C_2, a contradiction.

To conclude the proof of the lemma, we observe that when PD-SumRad opens a new cluster $C(z, 3r_k)$, the cluster $C(z, r_k)$ is tight. Hence, the total cost of $C(z, 3r_k)$ is at most $3 \sum_{u_j \in C(z, r_k)} a_j$. Therefore, the total cost of PD-SumRad is at most:

$$\sum_{(z,k):C(z,3r_k)\,\text{op.}} \sum_{u_j \in C(z,r_k)} 3a_j = 3\sum_{j=1}^{n} a_j \,|\{(z,k) : C(z,3r_k)\text{ opens} \wedge u_j \in C(z,r_k)\}|$$
$$\leq 3(2+\log_2 n)\sum_{j=1}^{n} a_j$$

The inequality holds because for each $k = -1, 0, \ldots, \lfloor \log_2 n \rfloor$ and each u_j with $a_j > 0$, there is at most one pair (z,k) such that $C(z,3r_k)$ opens and $u_j \in C(z,r_k)$. □

6 Randomized and Fractional Online Algorithms

Throughout this section, we assume that the optimal solution only consists of clusters of radius $2^k f$. For simplicity, we assume that n is an integral power of 2 and known in advance. Using standard techniques, we can remove these assumptions, by only losing a constant factor in the competitive ratio.

Randomized Algorithm. We first present Simple-SumRad, that is a simple randomized algorithm of logarithmic competitiveness. Simple-SumRad is *memoryless*, in the sense that it keeps in memory only its solution, namely the centers and the radii of its clusters. When a new demand u_j arrives, if u_j is covered by an already open cluster C, Simple-SumRad assigns u_j to C. Otherwise, for each $k = 0, \ldots, \log_2 n$, the algorithm opens a new cluster $C(u_j, 2^k f)$ with probability 2^{-k}, and assigns u_j to the cluster $C(u_j, f)$, which opens with probability 1. We can show that:

Lemma 3. *Simple-SumRad achieves a competitive ratio of at most* $2(4+\log_2 n)$.

Fractional Algorithm. We conclude with a deterministic $O(\log\log n)$-competitive algorithm for the fractional version of OnlSumRad in general metric spaces. The fractional algorithm is based on the primal-dual approach of [2, 1], and is a generalization of the online algorithm for the fractional version of ParkPermit in [17, Section 4.1].

A fractional algorithm maintains, in an online fashion, a feasible solution to the Linear Programming relaxation of OnlSumRad. In the notation of Section 5, for each point-type pair (z,k), the algorithm maintains a fraction x_{zk}, which denotes the extent to which the cluster $C(z,r_k)$ opens, and can only increase as new demands arrive. For each demand u_j, the fractions of the clusters covering u_j must sum up to at least 1, i.e. $\sum_{(z,k):u_j \in C(z,r_k)} x_{zk} \geq 1$. The total cost of the algorithm is $\sum_{(z,k)} x_{zk}(f + r_k)$.

The Algorithm. The fractional algorithm, or Frac-SumRad in short, considers $K+1$ different types of clusters, where $K = \log_2 n$. For each $k = 1, \ldots, K+1$, we let $c_k = f + r_k$ denote the cost of a cluster $C(p, r_k)$ of type k. The algorithm considers only the demand locations as potential cluster centers. For convenience, for each demand u_j and for each k, we let x_{jk} be the extent to which the cluster $C(u_j, r_k)$ is open, with the understanding that $x_{jk} = 0$ before u_j arrives. Similarly, we let $F_{jk} = \sum_{(i,k):u_j \in C(u_i,r_k)} x_{ik}$ be the extent to which demand u_j is covered by clusters of type k, and let $F_j = \sum_k F_{jk}$ be the extent to which u_j is covered.

When a new demand u_j, $j = 1, \ldots, n$, arrives, if $F_j \geq 1$, u_j is already covered. Otherwise, while $F_j < 1$, Frac-SumRad performs the following operation:

1. For every $k = 1, \dots K+1$, $x_{jk} \leftarrow x_{jk} + \frac{1}{c_k(K+1)}$
2. For every $k = 1, \dots, K+1$ and every demand $u_i \in C(u_j, r_k)$, $x_{ik} \leftarrow x_{ik}(1 + \frac{1}{c_k})$

Frac-SumRad maintains a (fractional) feasible solution in an online fashion. The proof of the following theorem extends the competitive analysis in [17, Section 4.1].

Theorem 5. *The competitive ratio of Frac-SumRad is $O(\log \log n)$.*

References

1. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A General Approach to Online Network Optimization Problems. *ACM Transactions on Algorithms*, 2(4):640-660, 2006.
2. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The Online Set Cover Problem. *SIAM J. on Computing*, 39(2):361-370, 2009.
3. V. Biló, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Geometric Clustering to Minimize the Sum of Cluster Sizes. *ESA '05*, LNCS 3669, pp. 460-471, 2005.
4. T.M. Chan and H. Zarrabi-Zadeh. A Randomized Algorithm for Online Unit Clustering. *Theory of Computing Systems*, 45(3):486-496, 2009.
5. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental Clustering and Dynamic Information Retrieval. *SIAM J. on Computing*, 33(6):1417-1440, 2004.
6. M. Charikar and R. Panigrahy. Clustering to Minimize the Sum of Cluster Diameters. *J. of Computer and System Sciences*, 68(2):417-441, 2004.
7. J. Csirik, L. Epstein, C. Imreh, and A. Levin. Online Clustering with Variable Sized Clusters. *MFCS '10*, LNCS 6281, pp. 282-293, 2010.
8. G. Divéki and C. Imreh. An Online 2-Dimensional Clustering Problem with Variable Sized Clusters. *Submitted for publication*, 2011.
9. S. Doddi, M.V. Marathe, S.S. Ravi, D.S. Taylor, and P. Widmayer. Approximation Algorithms for Clustering to Minimize the Sum of Diameters. *Nordic J. Computing*, 7(3):185-203, 2000.
10. M.R. Ehmsen and K.S. Larsen. Better Bounds on Online Unit Clustering. *SWAT '10*, LNCS 6139, pp. 371-382, 2010.
11. L. Epstein and R. van Stee. On the Online Unit Clustering Problem. *ACM Transactions on Algorithms*, 7(1):7, 2010.
12. D. Fotakis. On the Competitive Ratio for Online Facility Location. *Algorithmica*, 50(1):1-57, 2008.
13. M. Gibson, G. Kanade, E. Krohn, I.A. Pirwani, and K. Varadarajan. On Clustering to Minimize the Sum of Radii. *SODA '08*, pp. 819-815, 2008.
14. M. Gibson, G. Kanade, E. Krohn, I.A. Pirwani, and K. Varadarajan. On Metric Clustering to Minimize the Sum of Radii. *Algorithmica*, 57:484-498, 2010.
15. N. Lev-Tov and D. Peleg. Polynomial Time Approximation Schemes for Base Station Coverage with Minimum Total Radii. *Computer Networks*, 47(4):489-501, 2005.
16. A. Meyerson. Online Facility Location. *FOCS '01*, pp. 426-431, 2001.
17. A. Meyerson. The Parking Permit Problem. *FOCS '05*, pp. 274-284, 2005.
18. S.E. Schaeffer. Graph Clustering. *Computer Science Review*, 1:27-64, 2007.