A Tree Structure For Dynamic Facility Location

Gramoz Goranci
University of Vienna, Faculty of Computer Science, Vienna, Austria
gramoz.goranci@univie.ac.at

Monika Henzinger
University of Vienna, Faculty of Computer Science, Vienna, Austria
monika.henzinger@univie.ac.at

Dariusz Leniowski
University of Vienna, Faculty of Computer Science, Vienna, Austria
dariusz.leniowski@univie.ac.at

Abstract

We study the metric facility location problem with client insertions and deletions. This setting differs from the classic dynamic facility location problem, where the set of clients remains the same, but the metric space can change over time. We show a deterministic algorithm that maintains a constant factor approximation to the optimal solution in worst-case time $\tilde{O}(2^{O(\kappa^2)})$ per client insertion or deletion in metric spaces while answering queries about the cost in $O(1)$ time, where κ denotes the doubling dimension of the metric. For metric spaces with bounded doubling dimension, the update time is polylogarithmic in the parameters of the problem.

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases facility location, dynamic algorithm, approximation, doubling dimension

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.39

Funding The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.

1 Introduction

In the *metric facility location problem*, we are given a (possibly infinite) set V of *potential clients* or *points*, a finite set $\mathcal{C}$ of (*live clients*), a finite set $J \subseteq V$ of *facilities* with an *opening cost* f_j, for each facility j, and a metric d over V, such that $\mathrm{d}(i, j)$ is the cost of assigning client i to facility j. The goal is to determine a subset $J' \subseteq J$ of *open* facilities and to assign each client to an open facility such as to minimize the total cost. Obviously it is best to assign each live client to the closest *open* facility. Thus, the goal can be written as minimizing the objective function $\sum_{j \in J'} f_j + \sum_{i \in \mathcal{C}} \min_{j \in J'} \mathrm{d}(i, j)$.

The facility location problem is one of the central problems in combinatorial optimization and operations research [7], with many real-word applications. Typical examples include placements of servers in a network, location planning for medical centers, fire stations, restaurants, etc. From the computational perspective, this problem is NP-hard and it is even hard to approximate to a factor better than 1.463 [13, 20]. The best-known polynomial-time algorithm achieves a 1.488-approximation [17].

In many applications of facility location, problem data are continuously changing. This has lead to the study of this problem in different settings, e.g., online [18, 3, 10, 4, 11, 19, 1], streaming [15, 12, 16, 6] or dynamic [21, 5, 9, 8]. The focus of this paper is on the dynamic

setting, motivated by mobile network applications, where the set of clients may change over time, and we need to maintain the set of opened servers so as to obtain a solution of small cost after each change.

Formally, in the *dynamic facility location* problem, the set of clients $\mathcal{C}$ evolves over time and queries about both the cost as well as the set of opened facilities can be asked. Specifically, at each timestep t, either a new client is added to $\mathcal{C}$, a client is removed from $\mathcal{C}$, a query is made for the approximate cost of an optimal solution (*cost query*), or a query asks for the entire current solution (*solution query*). The goal is to maintain a set of open facilities that after each client update minimizes the above cost function. Thus, the cost f_j of each facility can be seen as a *maintenance cost* that has to be paid for each open facility between two client updates.

Our contribution. In this paper we present a *deterministic* data-structure that maintains a $O(1)$-factor approximation algorithm for the metric facility location problem, while supporting insertions and deletions of clients in $\tilde{O}(2^{O(\kappa^2)})$ update time, and answering cost queries in $O(1)$ time, where κ is the *doubling dimension*[1] of the metric space. As the running time per client update is bounded by $\tilde{O}(2^{O(\kappa^2)})$, the number of changes in the client-facility assignments is also bounded by this function. For metric spaces with bounded doubling dimension, such as the Euclidean space, the running time is $\tilde{O}(1)$. Formally, we have the following theorem.

▶ **Theorem 1.** *There exists a deterministic algorithm for the dynamic facility location problem where clients and facilities live in a metric space with doubling dimension κ, such that at every time step the solution has cost at most $O(1)$ times the cost of an optimal solution at that time. The worst-case update time for client insertion or deletion is $O(2^{O(\kappa^2)} \cdot \Delta^3 \cdot (\kappa^2 + \log \Delta))$, where Δ is logarithmic in the paramters of the problem. A cost query can be answered in constant time and a solution query in time linear in the size of the output.*

Comparison with prior work. The closest work related to our problem is the *streaming* algorithm for the *metric* facility location problem with *uniform* opening costs due to Lammersen and Sohler [16]. Specifically, given a sequence of insert and deletion operations of points (clients) from $\{1, \ldots, \Delta\}^d$, they devise a Monte-Carlo *randomized* algorithm that processes an insertion or deletion of a point in $\tilde{O}(2^{O(d)})$ time, using poly-logarithmic space and maintaining a $\tilde{O}(2^{O(d)})$-factor approximation. Since the d-dimensional Euclidean space has doubling dimension linear in d, we can also interpret the above result in terms of κ, i.e., the same bounds hold with d replaced by κ. An easy inspection of the algorithm in [16] shows that the queries can also be answered anytime in the update sequence in $O(1)$ time. Note that this algorithm heavily relies on randomization and the fact that facilities have uniform opening costs. In comparison our algorithm is (1) *deterministic*, (2) achieves a $O(1)$-factor approximation (independent of doubling dimension) (3) generalizes to any metric of bounded doubling dimension, and (4) supports non-uniform opening costs. Furthermore, for the Euclidean plane, i.e. $d = 2$, there is a randomized streaming algorithm that achieves a $(1 + \varepsilon)$-approximation with poly-logarithmic space [6]. However, it is not clear whether this algorithm supports fast queries.

Regarding the *dynamic facility location problem*, multiple variants can be found in the literature [21, 5, 9, 8]. All these variants are different from ours as they assume that the

[1] The doubling dimension of a metric space (V, d) is bounded by κ if for any $x \in V$ and any radius r, any ball with center x and radius r in (V, d) can be completely covered by 2^κ balls of radius $r/2$.

facilities and clients remain the same, and only *the distance metric* between clients and facilities can change. Additionally, every time a client switches to a different facility, a switching cost might be incurred and the goal is to minimize the above sum plus all the switching costs. For the version proposed in [8], there exists a polynomial time constant factor approximation algorithm [2].

There is also a large body of prior work on *online facility location* (see, e.g., [18, 3, 10, 4, 11, 19, 1]), where the clients arrive in an online fashion and have to be connected to a facility, potentially opening new facilities. If earlier decisions cannot be reversed, then no efficient constant factor approximation algorithm is possible [11]. If earlier decisions are, however, not permanent, specifically if facilities are only opened for a given amount of time, i.e. *leased*, and k different lease lengths are possible [4], the offline version of the problem has a polynomial time 3-approximation algorithm [19], but the online setting cannot have an approximation better than $\Omega(\log k)$, even for randomized algorithms [18]. All of these results are different from ours: (1) We have only one lease length, namely length 1, as each facility can be closed or opened after each timestep, (2) we allow client arrival *and* departure, and (3) our algorithm processes a client update in $O(2^{O(\kappa^2)} \cdot \Delta^3 \cdot (\kappa^2 + \log \Delta))$ worst-case time per operation, where Δ is logarithmic in the parameters of the problem, while the running time of the online algorithms is at least linear in the number of facilities [18, 3].

Technical contribution. From a technical point of view we modify and significantly extend a hierarchical partition of a subset of the facilities that was recently introduced for a related problem, called the *dynamic sum-of-radii clustering* problem [14]. In that work, a set of facilities J and a dynamically changing clients $\mathcal{C}$ are given and the goal is to output a set $J' \subseteq J$ together with a *radius* R_j, for each $j \in J'$, such that the J' covers $\mathcal{C}$ and the function $\sum_{j \in J'}(f_j + R_j)$ is minimized. In [14] a $O(2^{2\kappa})$-approximation algorithm with time $\tilde{O}(2^{6\kappa})$ per client insertion or deletion is presented for metrics with doubling dimension κ. Note that the function that is minimized is different from the function minimized in the facility location problem, as the term $\sum_{i \in \mathcal{C}} \min_{j \in J'} \mathrm{d}(i, j)$ is replaced by $\sum_{j \in J'} R_j$.

More specifically, the hierarchical decomposition of [14] picks a well-separated subset of J with "small" cost, assigns one or multiple radii to the selected facilities, and then hierarchically orders the pairs $\langle j, R \rangle$ in a tree structure, where j is a facility and R is a radius, such that the children of every pair have a smaller radius and the "ball" of the given radius of a child is fully contained in the "ball" of its parent with its radius. To achieve our result, in Sections 2 and 3 we extend this decomposition as follows:

1. *Abundance condition.* Instead of selecting facilities with "small" cost, we introduce the notion of an "abundance condition": Facilities that have "enough nearby" clients fulfill this condition, and we *only open such facilities*. This leads to following rough notion: The abundance condition is fulfilled for a facility j and a radius R if the number of clients within radius R of j is at least f_j/R. The fundamental idea is then as follows: (A) We assign a *payment* of R to each client within radius R of an open facility j, which implies that the sum of f_j plus the distances of these clients to j is upper bounded by twice the sum of the payments of these clients. (B) We then show that (i) each client pays for at most one facility and (ii) the sum of the client payments is linear in the cost of the optimal solution.
2. *Designated facilities.* To further reduce the cost of the open facilities, we designate to each facility j the "cheapest nearby" facility j^*, and if a facility j fulfills all conditions to be opened, we open the facility j^* instead. This allows us to modify the (rough) abundance condition so that it is fulfilled for a facility j and a radius R if the number of

clients within radius R of j is at least f_{j^*}/R. This modification increases the distance of the clients only by a constant factor, but might significantly decrease the cost of the open facilities. Note that our approach would not achieve a constant factor without this technique: The hierarchical decomposition is based on a well-separated subset of facilities whose construction ignores the facility costs. Thus it might happen that the chosen facilities have high cost, even though there are "cheap" facilities nearby. These cheap nearby facilities are now captured by the designated facilities.

3. *Enabled facilities.* The idea of not opening facilities that are "close" to an open facility can be further combined with the hierarchical decomposition. More specifically, if facility j with radius R and facility j' with radius R' are "close", only one of them is opened, namely the lowest-in-the-hierarchy facility that fulfills the abundance condition and has no nearby facility of smaller radius that is already open. The advantage of this scheme is that when a facility switches from open to closed or vice versa, we can bound the number of facilities that are affected by this change by a function that only depends on κ. If we had chosen to open the facility with larger radius, then the number of facilities that are affected by opening or closing one facility might have been large, i.e., not bounded by a function of κ alone.

4. *Coloring.* Recall that we need to guarantee that each client only pays for *one* open facility. To do so, we assign a *color* to each pair $\langle j, r\rangle$ in the hierarchy such that no two pairs $\langle j, r\rangle$ and $\langle j', r\rangle$, where the distance between j and j' is small, have the same color. As the metric has bounded doubling dimension, $2^{5\kappa}+1$ colors suffice for this coloring. Then we require that a facility is opened only if it fulfills the abundance condition, *and* no facility of either smaller radius or the same radius but with "smaller" color is already open. This requires to further relax the notion of "closeness" but reduces the number of open facilities enough so that each client can be assigned to pay for at most one "close" open facility and still every open facility has enough clients paying for its cost.

In Section 3, Theorem 24, we show that (a) for clients that are "close" to an open facility j in the *optimal solution* the sum of their payments (in our solution) is linear in f_j and (b) for clients that are "far" from an open facility in the *optimal solution* their payments (in our solution) are within a constant factor of their distance in the optimal solution. Additionally, we give a data structure that maintains this solution efficiently under insertions and deletion of clients (see Section 4). All missing proofs are deferred to the full version.

2 Preprocessing phase

Let W be the diameter of the metric space, i.e., $\mathrm{d}(i,j) \leq W$, for all $i, j \in V$, let $f_{\max} = \max_{j\in J}\{f_j\}$ be the maximum facility opening cost, and let $f_{\min} = \min\{f_j \mid j \in J, f_j > 0\} > 0$ be the minimum opening cost of any facility with non-zero opening cost. This is w.l.o.g. as all facilities with 0 opening cost are always kept open. Given the set of clients $\mathcal{C} \subseteq V$, let $\mathrm{OPT} = \mathrm{OPT}(\mathcal{C})$ denote the cost of an optimum solution for $\mathcal{C}$. In what follows, for the sake of exposition, we also let $\mathcal{C}$ to refer to the *current* set of clients.

The algorithm will maintain a number $n = 5^{\lfloor \log_5 |\mathcal{C}| \rfloor}$, i.e., the largest power of 5 smaller than $|\mathcal{C}|$: Initally we set n to 0 and use this value of n during preprocessing. Whenever the first client is inserted, we set $n = 1$. Afterwards, whenever the number of clients is a factor 5 larger, resp. smaller, than n, we update n by multiplying, resp. dividing it by 5.

Now let[2] $\rho_{\min} = \lceil \log_5 (f_{\min}/\max(|J|, n)) \rceil$ and let $\rho_{\max} = \lceil \log_5 (\max(W, f_{\max})) \rceil = O(\log W + \log f_{\max})$. Note that a change of $\rho_{\min}$ will require an update in our data structures, but this will only happen after $\Theta(n)$ many client insertions or deletions. As we will see later, the cost of this update will be charged against these client updates, and thus does not affect our running times. A *logradius* is an integer r such that $\rho_{\min} \leq 5^r \leq \rho_{\max}$. Let $\Delta = \rho_{\max} - \rho_{\min} + 1$ be the number of different logradii. Note that $\Delta = O(\log W + \log(f_{\max}/f_{\min}) + \log|J| + \log|\mathcal{C}|)$. Finally, let $c_1 = 20$, $c_2 = 35$, $c_X = 2c_2 + 2 = 72$, $c_3 = c_X + c_2 = 107$, $c_Y = 2c_3 + c_2 = 249$ and $c_4 = c_Y + c_2 = 284$.

A large part of our data structure is concerned with reducing the number of facilities that are potentially opened and finding an assignment of each client to at most one open facility. This is done in multiple ways, as described next. Based on the approach of [14], we construct a set of pairs $\Pi \subseteq (J \times [\rho_{\min}, \rho_{\max}])$, consisting of facility-logradius pairs and a laminar family of *areas*. Different from [14], we color pairs in Π, turning them into triplets, and introduce designated facilities, before defining open, closed and enabled triplets.

Maximal subsets of distant facilities. The first step is to filter out facilities that are close to other facilities. To achieve this, we greedily construct a set Π of *pairs* $\langle j, r\rangle$ where j is a facility and r is a logradius, satisfying the following properties:

1. (Covering) For every facility $j \in J$ and every logradius r, there exists a facility $j' \in J_r$ with $\mathrm{d}(j, j') \leq c_1 \cdot 5^r$.
2. (Separating) For all distinct $j, j' \in J_r$, $\mathrm{d}(j, j') > c_1 \cdot 5^r$.

We construct Π as follows: For each logradius $r \in [\rho_{\min}, \rho_{\max}]$, let J_r be a maximal subset of J such that any two facilities in J_r are at distance strictly larger than $c_1 \cdot 5^r$. Set $\Pi \leftarrow \bigcup_r \{\langle j, r\rangle \mid j \in J_r\}$. Note that for $r = \rho_{\max}$, the set J_r contains just one facility.

Hierarchical decomposition of Π. We now construct a hierarchical decomposition of Π and represent it by a tree $\mathcal{T}$, using the following algorithm. Set the root of $\mathcal{T}$ to be the unique pair $\langle j, \rho_{\max}\rangle$. For each $r < \rho_{\max}$ and $j \in J_r$: (1) Set $j' \in J_{r+1}$ be the facility closest to j. (2) Set $\mathrm{parent}(j, r) \leftarrow \langle j', r+1\rangle$.

By construction, $\mathcal{T}$ has height at most Δ and the parent of a pair $\langle j, r\rangle$ is a pair of the form $\langle j', r+1\rangle$. The following three lemmata describe the crucial properties of the tree $\mathcal{T}$.

▶ **Lemma 2** (Nesting of balls). *Let c^* be any constant such that $c^* \geq (5/4)c_1$. If $\mathrm{parent}(j, r) = \langle j', r+1\rangle$, then $\mathrm{d}(j, j') \leq c_1 \cdot 5^{r+1}$ and $B(j, c^* \cdot 5^r) \subseteq B(j', c^* \cdot 5^{r+1})$.*

▶ **Lemma 3** ([14]). *For any point p, radius r and some number $\alpha > 0$, the set of pairs $\Pi(p, r) = \{\langle j, r\rangle \in \Pi \mid \mathrm{d}(p, j) < 2^\alpha c_1 \cdot 5^r\}$ has at most $2^{(\alpha+1)\kappa}$ elements, where κ is the doubling dimension of the metric space.*

▶ **Lemma 4** ([14]). *A node $\langle j, r\rangle$ of $\mathcal{T}$ has at most $2^{4\kappa}$ children*

Hierarchical decomposition of V into a laminar family of areas. The balls $B(j, r)$ and $B(j', r)$ with $\langle j, r\rangle$ and $\langle j', r\rangle$ in Π might overlap, which is problematic for the mapping of clients to facilities. To rectify this problem, we partition V into a laminar family of *areas* such that no two same-logradius areas overlap, as follows: For each $\langle j, r\rangle \in \Pi$, initialize $A(j, r) \leftarrow \emptyset$. Next, for each point $p \in V$: (1) Let r^* be the smallest such that there exists pairs $\langle j, r^*\rangle \in \Pi$ with $p \in B(j, c_2 \cdot 5^{r^*})$. (2) Among all such pairs, let $\langle j^*, r^*\rangle$ denote the

[2] Note that $\rho_{\min}$ could be negative, but it is well-defined as $f_{\min} > 0$.

one minimizing $\mathrm{d}(p, j^*)$. (3) Add p to the set $A(j^*, r^*)$ and to every set $A(j', r')$ with (j', r') ancestor of (j^*, r^*) in $\mathcal{T}$.

The laminar family of areas fulfills the following lemmata.

▶ **Lemma 5** ([14]). *For each $\langle j, r\rangle \in \Pi$, $j \in A(j,r) \subseteq B(j, c_2 \cdot 5^r)$ and if $\mathrm{parent}(j,r) = \langle j', r+1\rangle$, then $A(j,r) \subseteq A(j', r+1)$.*

▶ **Lemma 6.** *Let $j \in J$, $r \in [\rho_{\min}, \rho_{\max}]$ and $p \in B(j, 5^r)$. Then there exists a pair $\langle j', r\rangle \in \Pi$ such that $p \in A(j', r)$ and $\mathrm{d}(j, j') \leq (c_2 + 1) \cdot 5^r$.*

Additionally an even stronger statement regarding the points covered by areas versus points covered by balls holds:

▶ **Lemma 7.** *For each logradius $r \in [\rho_{\min}, \rho_{\max}]$, $\bigcup_{\langle j,r\rangle \in J_r} A(j,r) = \bigcup_{\langle j,r\rangle \in J_r} B(j, c_2 \cdot 5^r)$.*

Covering balls using unions of areas. To select which facilities with a pair $\langle j, r\rangle$ in Π to open, we introduce below the abundance condition which measures how many clients are "close" to j. For measuring "closeness", we would like to say that a client i is close to a facility j if $i \in \mathrm{X}(j,r)$ for some definition of $\mathrm{X}(j,r)$ that fulfills the crucial property that for every $\langle j, r\rangle$ there exists a pair $\langle j', r\rangle \in \Pi$ such that $B(j, 5^r) \subseteq \mathrm{X}(j', r)$. Note that this might not hold if we use $\mathrm{X}(j,r) = A(j,r)$ and it does not follow from Lemma 6 as different points of $B(j, 5^r)$ might belong to different areas $A(j', r)$, whose facilities might be up to distance $(2c_2 + 2) \cdot 5^r$ apart. Thus, for any $\langle j, r\rangle \in \Pi$, we define $\mathrm{X}(j,r)$ as follows:

$$\mathrm{X}(j,r) = \bigcup \{A(j', r) \mid \mathrm{d}(j, j') \leq c_X \cdot 5^r\}, \quad \text{where } c_X = 2c_2 + 2,$$

and can now show the desired property for $\mathrm{X}(j,r)$:

▶ **Lemma 8.** *Let $j \in J$ with $\langle j, r\rangle \notin \Pi$. Then there exists $\langle j^*, r\rangle \in \Pi$ with $B(j, 5^r) \subseteq \mathrm{X}(j^*, r)$.*

Proof of Lemma 8. It suffices to show that there exists $\langle j^*, r\rangle \in \Pi$ such that for every area $A(j', r)$ that intersects with the ball $B(j, 5^r)$, we get that $A(j', r) \subseteq \mathrm{X}(j, r)$. First, by the Covering property, there exists a pair $\langle j^*, r\rangle \in \Pi$ such that $\mathrm{d}(j, j^*) \leq c_1 \cdot 5^r$. Next, let $A(j', r)$ be any area such that $A(j', r) \cap B(j, 5^r) \neq \emptyset$. By Lemma 6, we get that $\mathrm{d}(j', j) \leq (c_2 + 1) \cdot 5^r$. It follows that $\mathrm{d}(j', j^*) \leq \mathrm{d}(j', j) + \mathrm{d}(j, j^*) \leq (c_1 + c_2 + 1) \cdot 5^r \leq c_X \cdot 5^r$, which in tun implies that $A(j', r) \subseteq \mathrm{X}(j^*, r)$. ◀

It is crucial for the running time to get a bound on the number of areas used to construct $\mathrm{X}(j,r)$. We do this in the following lemma, which is a simple corollary of Lemma 3.

▶ **Lemma 9.** *For any $\langle j, r\rangle \in \Pi$, $\mathrm{X}(j,r)$ is a union of at most $2^{3\kappa}$ areas.*

We also need to bound how far any two points in $\mathrm{X}(j,r)$ can be apart:

▶ **Lemma 10.** *It holds that $\mathrm{X}(j,r) \subseteq B(j, c_3 \cdot 5^r)$.*

To further reduce the set of open facilities, it is necessary to introduce a notion of "closeness" between facilities that is more relaxed than the definition used for covering, where we required two facilities to be at least $c_1 \cdot 5^r$ apart. Now we guarantee that if a facility is open then no other facility within distance $c_Y \cdot 5^r$ is opened, resulting in the following definition of $\mathrm{Y}(j,r)$ for every $\langle j, r\rangle \in \Pi$:

$$\mathrm{Y}(j,r) = \bigcup \{A(j', r) \mid \mathrm{d}(j, j') \leq c_Y \cdot 5^r\}, \quad \text{where } c_Y = 2c_X + 3c_2.$$

The constant c_Y is chosen so that we can make sure that there are never two pairs $\langle j, r\rangle$ and $\langle j', r'\rangle$ such that both j and j' are open and $\mathrm{X}(j, r)$ and $\mathrm{X}(j', r')$ intersect. Thus, a client can always only belong to at most one set $\mathrm{X}(j, r)$, where $\langle j, r\rangle$ is open. The facility j will be the facility that the client is assigned to. The following lemmata follow as before:

▶ **Lemma 11.** *For any $\langle j, r\rangle \in \Pi$, $\mathrm{Y}(j, r)$ is a union of at most $2^{5\kappa}$ areas.*

▶ **Lemma 12.** *It holds that $\mathrm{Y}(j, r) \subseteq B(j, c_4 \cdot 5^r)$.*

▶ **Lemma 13.** *If $\langle j', r'\rangle$ is an ancestor of $\langle j, r\rangle$ in $\mathcal{T}$, then $\mathrm{Y}(j, r) \subseteq \mathrm{Y}(j', r')$.*

Coloring of pairs. We are now ready to define a tie-breaking rule based on colors. For any $\langle j, r\rangle \in \Pi$, consider the set $\Pi(j, r) = J_r \cap B(j, c_4 \cdot 5^r)$. By Lemma 3 and since $c_4 < 16c_1$, it follows that $\Pi(j, r)$ contains at most $2^{5\kappa}$ pairs. We need to guarantee that at most one of them will ever be opened. Thus we introduce a tie-breaking rule based on colors of pairs. This guarantees that out of all pairs in $\Pi(j, r)$ that fulfill the abundance condition only the one with the "smallest" color is opened.

More formally, we perform a preprocessing step using a greedy approach to color pairs of same log-radius in Π with $2^{5\kappa} + 1$ colors from 0 to $2^{5\kappa}$.

Specifically for each log-radius $r \in [\rho_{\min}, \rho_{\max}]$, we greedily color every pair $\langle j, r\rangle$ of J_r by one color s so that *no two pairs $\langle j, r\rangle, \langle j', r\rangle \in J_r$ with $\mathrm{d}(j, j') \leq c_4 \cdot 5^r$ are colored with the same color*, and we refer to $\langle j, r, s\rangle$ as *triplet*. Let $J_{r,s} := \{\langle j, r, s'\rangle \in J_r \mid s' = s\}$.

Designated facilities. Furthermore, even if a triplet $\langle j, r, s\rangle$ fulfills the condition to be opened (which is explained in the next section) it will not be opened, if there is a "cheaper" facility nearby. More formally, we precompute for each pair $\langle j, r\rangle$ in Π the following *designated facility*: Let $f^*_{\langle j,r\rangle}$ be the minimum opening cost of any facility in $\mathrm{X}(j, r)$, i.e.,

$$f^*_{\langle j,r\rangle} = \min\{f_{j'} \mid j' \in J \cap \mathrm{X}(j, r)\}.$$

The *designated facility* $j^*_{\langle j,r\rangle}$ of $\langle j, r\rangle$ is the facility with minimum cost $f^*_{\langle j,r\rangle}$ in $\mathrm{X}(j, r)$ with ties broken according to the minimum id-number, i.e., $j^*_{\langle j,r\rangle} = \min\{j' \mid j' \in J \cap \mathrm{X}(j, r), f_{j'} = f^*_{\langle j,r\rangle}\}$.

▶ **Observation 14.** *For any $\langle j, r\rangle \in \Pi$, $\mathrm{d}(j, j^*_{\langle j,r\rangle}) \leq c_3 \cdot 5^r$ and $f^*_{\langle j,r\rangle} \leq f_j$.*

If the triplet $\langle j, r, s\rangle$ fulfills the condition to be opened, we open $j^*_{\langle j,r\rangle}$ instead, or do nothing if $j^*_{\langle j,r\rangle}$ is already open. Whenever all triplets for which a facility is designated are closed, then the facility is *closed*.

3 Processing updates

After the preprocessing phase we are given a laminar family of triplets, where each triplet $\langle j, r, s\rangle$ is formed by an area $A(j, r)$ along with its pre-defined color s. Depending on the set of clients $\mathcal{C}$, a triplet can be either *disabled* or *enabled* and either *open* or *closed*, where each open triplet is also enabled. These properties of triplets are maintained dynamically as the set $\mathcal{C}$ of clients changes. Initially $\mathcal{C} = \emptyset$ and all triplets are closed and disabled. We now proceed to the formal definitions.

Open triplets. We *open* a triplet $\langle j, r, s\rangle$ if there are enough clients in the set $X(j, r)$ to pay the opening cost and it has no strictly smaller-radius or no same-radius and strictly smaller-color open triplet in its "neighborhood". A triplet that is not open is *closed*. Formally a triplet $\langle j, r, s\rangle$ is open if it belongs to the set $J^{\text{open}}_{r,s}(\mathcal{C})$, which is defined[3] recursively as follows.

$$J^{\text{open}}_{r,s}(\mathcal{C}) = \Big\{\langle j, r, s\rangle \in J_{r,s} \;\Big|\; 5^r \cdot |\mathcal{C} \cap \mathrm{X}(j, r)| \geq f^*_{\langle j,r\rangle} \wedge \forall \langle r', s'\rangle <_{\text{lex}} \langle r, s\rangle : \mathrm{Y}(j, r) \cap J^{\text{open}}_{r',s'}(\mathcal{C}) = \varnothing\Big\}.$$

We use $J^{\text{open}}_{\mathcal{C}} = \bigcup_{r,s} J^{\text{open}}_{r,s}(\mathcal{C})$ to denote the set of all open clients and $\mathcal{I}_{\mathcal{C}}$ to denote the set of all open facilities, i.e., $\mathcal{I}_{\mathcal{C}} = \{j^*_{\langle j,r\rangle} \in J \mid \langle j, r, s\rangle \in J^{\text{open}}_{\mathcal{C}}\}$.

We call the following condition for $\langle j, r, s\rangle$, used in the definition of $J^{\text{open}}_{r,s}$, the *abundance condition*,

$$5^r \cdot |\mathcal{C} \cap \mathrm{X}(j, r)| \geq f^*_{\langle j,r\rangle}. \tag{1}$$

▶ **Lemma 15.** *If $|\mathcal{C}| > 0$ then there exists at least one open triplet.*

When showing the bound on the approximation ratio, we need the property that for each point $i \in V$ there is *at most one* set $\mathrm{X}(j, r)$ associated with an open triplet such that i belongs to. This is necessary to make sure that each client "pays" for at most one open facility.

▶ **Lemma 16.** *Each client $i \in \mathcal{C}$ belongs to at most one $\mathrm{X}(j, r)$ with $\langle j, r, s\rangle \in J^{\text{open}}_{\mathcal{C}}$ for some color s.*

Note, however, that is not true that each client $i \in \mathcal{C}$ is contained in *at least one* $\mathrm{X}(j, r)$ associated with an open triplet for some r: even though there always exists a $\mathrm{X}(j, r)$ fulfilling the abundance condition and containing i (namely $\mathrm{X}(j^{\text{root}}, \rho_{\max})$), the corresponding triplet $\langle j^{\text{root}}, \rho_{\max}, s\rangle$ might not be open due to a "nearby" open triplet of smaller logradius. To deal with this issue we introduce enabled triplets and show that each client in $\mathcal{C}$ is contained in at least one $\mathrm{X}(j, r)$ of an enabled triplet.

Enabled triplets. A triplet $\langle j, r, s\rangle$ is *enabled* if it belongs to the set $J^{\text{enabled}}_{r,s}(\mathcal{C})$, which is defined[4] as follows:

$$J^{\text{enabled}}_{r,s}(\mathcal{C}) = \Big\{\langle j, r, s\rangle \in J_{r,s} \;\Big|\; \exists \langle r', s'\rangle \leq_{\text{lex}} \langle r, s\rangle : \mathrm{Y}(j, r) \cap J^{\text{open}}_{r',s'}(\mathcal{C}) \neq \varnothing\Big\}.$$

We use $J^{\text{enabled}}_{\mathcal{C}} = \bigcup_{r,s} J^{\text{enabled}}_{r,s}(\mathcal{C})$ to denote the set of all enabled facilities. The following observation follows from the definition.

▶ **Observation 17.** *If a triplet is open, then it is also enabled.*

Furthermore, as a corollary of Lemma 13 we have the following lemma.

▶ **Lemma 18.** *If a triplet is enabled, then all its ancestors in $\mathcal{T}$ are also enabled.*

---

[3] Remark that to make the formula a bit simpler we slightly abuse notation here – $\mathrm{Y}(j, r)$ is a set of areas (i.e., subsets of the metric space), while $J^{\text{open}}_{r',s'}(\mathcal{C})$ is a set of triples. Formally the intersection should be understood as $\mathrm{Y}(j, r) \cap \{j' \in J \mid \langle j', r', s'\rangle \in J^{\text{open}}_{r',s'}(\mathcal{C})\}$.

[4] Similarly to the definition of $J^{\text{open}}_{r,s}(\mathcal{C})$ we mean here $\mathrm{Y}(j, r) \cap \{j' \in J \mid \langle j', r', s'\rangle \in J^{\text{open}}_{r',s'}(\mathcal{C})\}$.

Since $A(j^{\text{root}}, \rho_{\max}) = V$, every point in V belongs to at least one $\mathrm{X}(j,r)$ associated with an enabled triplet. To guarantee that at least one triplet is enabled, recall that Lemma 15 showed that if $|\mathcal{C}| > 0$, then there exist an open, and, thus, also enabled triplet (see Observation 17). Lemma 18 implies that the root of $\mathcal{T}$ is enabled, implying the following lemma.

▶ **Lemma 19.** *If $|\mathcal{C}| > 0$ then every point in V belongs to at least one $\mathrm{X}(j,r)$ associated with an enabled triplet.*

The next lemma shows that the definition of enabled implies that any triplet that satisfies the abundance definition is either open or enabled. This is a crucial observation for the proof of the approximation ratio, as it will allow us to argue that for any facility that the optimal solution opens, an enabled triplet must be nearby.

▶ **Lemma 20.** *If $\langle j,r,s\rangle$ satisfies the abundance condition, then it is enabled.*

Assignment of clients. We next describe how to assign each client i to an enabled triplet $\langle j,r,s\rangle$ and an open facility. If $\langle j,r,s\rangle$ is not open, we show how to find a close open triplet of smallest radius. For this open triplet we know its designed facility that is open. This is the facility that the client is finally assigned to.

We start with the assignment of $i \in C$ to an enabled triplet. To this end, let r_i^{area} be the minimum logradius of any enabled triplet such that i belongs to the area associated with the triplet, i.e., $r_i^{\text{area}} = \min\{r \mid \langle j,r,s\rangle \in J_{\mathcal{C}}^{\text{enabled}}, i \in A(j,r)\}$. Note that $A(j,r) \subseteq \mathrm{X}(j,r)$, and let j_i^{area} be the center of the area with log-radius r_i^{area} and define $\langle j_i^{\text{area}}, r_i^{\text{area}}, s_i^{\text{area}}\rangle$ to be the corresponding triplet (recall that same-logradius areas are disjoint).

Once we determined the enabled triplet $\langle j_i^{\text{area}}, r_i^{\text{area}}, s_i^{\text{area}}\rangle$ of i, we assign i to the open triplet of minimum radius such that the corresponding facility belongs to $\mathrm{Y}(j_i^{\text{area}}, r_i^{\text{area}})$ and let j_i^{open} be the designated open facility of that open triplet. We assign i to it. Formally:

$$\langle r_i^{\text{aux}}, s_i^{\text{aux}}, j_i^{\text{aux}}\rangle = \min\Big\{\langle r',s',j'\rangle \;\Big|\; \langle j',r',s'\rangle \in J_{\mathcal{C}}^{\text{open}}, \langle r',s'\rangle \le_{\text{lex}} \langle r_i^{\text{area}}, s_i^{\text{area}}\rangle, \\ j' \in \mathrm{Y}(j_i^{\text{area}}, r_i^{\text{area}})\Big\},$$

$$j_i^{\text{open}} = j^*_{\langle j_i^{\text{aux}}, r_i^{\text{aux}}\rangle}.$$

Finally we denote the set of all clients assigned to facility j by $\mathcal{C}_j = \{i \in \mathcal{C} \mid j = j_i^{\text{open}}\}$. Note that j_i^{open} does not have to be the closest open facility, but as the next lemma shows it is not far away from an open facility.

▶ **Observation 21.** *Any $i \in \mathcal{C}$ is within $(c_2+c_3+c_4)\cdot 5^{r_i^{\text{area}}}$ of an open facility.*

The value r_i^{area} is crucial for the cost estimate. Thus it is important to characterize this value even further, as we do in the following lemma.

▶ **Lemma 22.** *For any $i \in \mathcal{C}$, if $i \in \mathrm{X}(j,r)$ and $\langle j,r,s\rangle \in J_{\mathcal{C}}^{\text{open}}$ for some color s, then $r_i^{\text{area}} = r$.*

Assume we open each facility in $\mathcal{I}_{\mathcal{C}}$ and each client is assigned to an open facility as described above or an even closer one, if one exists. As a consequence of the above lemma and the definition of open facilities we can now bound the total cost of the solution by $O(\sum_{i\in\mathcal{C}} 5^{r_i^{\text{area}}})$.

▶ **Lemma 23.** *It holds that $\sum_{i\in\mathcal{C}} \mathrm{d}(i,j_i) + \sum_{j\in\mathcal{I}_{\mathcal{C}}} f_j \le \sum_{i\in\mathcal{C}} (c_2+c_3+c_4+1)\cdot 5^{r_i^{\text{area}}}$.*

Now we are ready to prove the bound on the approximation ratio.

▶ **Theorem 24.** *For any subset $\mathcal{C} \subseteq V$ of clients, assign each client $i \in \mathcal{C}$ to the facility j_i^{open}. Then the cost of this solution is $O(1) \cdot \text{OPT}$, where OPT is the optimal solution for $\mathcal{C}$.*

Proof. Denote by $\mathcal{I}^*$ an arbitrary optimal solution. For each $i \in \mathcal{C}$ define j_i^* to be the facility i is connected to. Moreover, for each $j \in J$ let $\mathcal{C}_j^*$ be the set of clients connected to j. Formally, $j_i^* = \min\{j \in \mathcal{I}^* \mid \mathrm{d}(i,j) = \mathrm{d}(i,\mathcal{I}^*)\}$, $\mathcal{C}_j^* = \{i \in \mathcal{C} \mid j = j_i^*\}$. Consider some $j \in \mathcal{I}^*$ and let $r \in [\rho_{\min}, \rho_{\max}]$ be the logradius such that

$$5^{r-1} \cdot |\mathcal{C}_j^* \cap B(j, 5^{r-1})| < f_j \leq 5^r \cdot |\mathcal{C}_j^* \cap B(j, 5^r)|. \tag{2}$$

Note that r is well-defined as $5^{\rho_{\max}} \cdot |\mathcal{C}_j^* \cap B(j, 5^{\rho_{\max}})| \geq f_{\max} \geq f_j$ by the definition of $\rho_{\max}$ and $f_j \geq f_{\min} \geq 5^{\rho_{\min} + \log_5 n - 1} \geq 5^{\rho_{\min}-1} \cdot |\mathcal{C}_j^* \cap B(j, 5^{\rho_{\min}-1})|$ by the definition of $\rho_{\min}$.

To complete the proof we split $\mathcal{C}_j^*$ into two sets, $\mathcal{C}_j^{\text{lo}}$ and $\mathcal{C}_j^{\text{hi}}$, according to whether $i \in B(j, 5^{r-1})$ or not, i.e., $\mathrm{d}(i,j) \leq 5^{r-1}$ or $\mathrm{d}(i,j) > 5^{r-1}$ respectively, and show that

$$\sum_{i \in \mathcal{C}_j^{\text{lo}}} 5^{r_i^{\text{area}}} < 5 \cdot f_j, \text{ and} \tag{A}$$

$$\sum_{i \in \mathcal{C}_j^{\text{hi}}} 5^{r_i^{\text{area}}} \leq \sum_{i \in \mathcal{C}_j^{\text{hi}}} 5 \cdot \mathrm{d}(i,j). \tag{B}$$

Before showing the above inequalities we first argue that they prove the bound on the approximation ratio. Note that every client belongs to $\mathcal{C}_j^*$ for some $j \in \mathcal{I}^*$. Thus,

$$\sum_{i \in \mathcal{C}} 5^{r_i^{\text{area}}} \leq \sum_{j \in \mathcal{I}^*} 5 \cdot f_j + \sum_{i \in \mathcal{C}} 5 \cdot \mathrm{d}(i,j) \leq 5 \cdot \text{OPT}.$$

Using Lemma 23 we get that

$$\sum_{i \in \mathcal{C}} \mathrm{d}(i, j_i^{\text{open}}) + \sum_{j \in \mathcal{I}_{\mathcal{C}}} f_j \leq \sum_{i \in \mathcal{C}} (c_2 + c_3 + c_4 + 1) \cdot 5^{r_i^{\text{area}}} \leq 5(c_2 + c_3 + c_4 + 1) \cdot \text{OPT}.$$

We first show (A). Consider $i \in \mathcal{C}_j^{\text{lo}}$, or equivalently, $i \in B(j, 5^{r-1})$. We claim that $r_i^{\text{area}} \leq r$. Indeed, if $\langle j, r, s \rangle \in \Pi$ for some color s, then $\mathrm{X}(j,r) \supseteq A(j,r) \supseteq B(j, 5^r)$ along with (2) give

$$5^r \cdot |\mathcal{C} \cap \mathrm{X}(j,r)| \geq 5^r \cdot |\mathcal{C}_j^* \cap B(j, 5^r)| \geq f_j \geq f^*_{\langle j,r \rangle},$$

which in turn implies that $\langle j, r, s \rangle$ satisfies the abundance condition and so it is enabled (Observation 20). By definition of r_i^{area}, we get $r_i^{\text{area}} \leq r$. If $\langle j, r, s \rangle \notin \Pi$ for any s, then by Lemma 6, there exists a triplet $\langle j', r, s \rangle$ such that $i \in A(j', r)$. Because $\mathrm{d}(j', j) \leq \mathrm{d}(j', i) + \mathrm{d}(i, j) \leq (c_2 + 1) \cdot 5^r$ we have that $B(j, 5^r) \subseteq \mathrm{X}(j', r)$. This along with (2) imply that

$$5^r \cdot |\mathcal{C} \cap \mathrm{X}(j', r)| \geq 5^r \cdot |\mathcal{C}_j^* \cap B(j, 5^r)| \geq f_j \geq f^*_{\langle j', r \rangle},$$

where the last inequality follows by definition of $f^*_{\langle j', r \rangle}$. Similarly it follows that $\langle j', r, s \rangle$ is enabled and $r_i^{\text{area}} \leq r$. Recalling that $i \in B(j, 5^{r-1})$ we finally arrive at

$$\sum_{i \in \mathcal{C}_j^{\text{lo}}} 5^{r_i^{\text{area}}} \leq \sum_{i \in \mathcal{C}_j^{\text{lo}}} 5^r \leq 5 \cdot 5^{r-1} \cdot |\mathcal{C}_j^* \cap B(j, 5^{r-1})| \leq 5 \cdot f_j.$$

We next show (B). First, observe that for any ball $B(j, 5^{r''})$, $r'' \geq r$ with $i \in B(j, 5^{r''})$, one can prove similarly to the above that there exists an *enabled* triplet $\langle j'', r'', s \rangle$ such that

$i \in A(j'', r'')$. We have that $\mathrm{d}(j'', j) \leq \mathrm{d}(j'', i) + \mathrm{d}(i, j) \leq (c_2 + 1) \cdot 5^{r''}$, thus $X(j'', r'') \supseteq B(j, 5^{r''}) \supseteq B(j, 5^r)$. This, together with (2) and $f^*_{\langle j'', r'' \rangle} \leq f_j$ give the claim.

Now, consider $i \in \mathcal{C}^{\mathrm{hi}}_j$. Since $\mathrm{d}(i, j) > 5^{r-1}$, we get that $\lceil \log_5 \mathrm{d}(i, j) \rceil \geq r$ and $i \in B(j, 5^{\lceil \log_5 \mathrm{d}(i,j) \rceil})$. By the discussion above, there exists an enabled triplet $\langle j'', \lceil \log_5 \mathrm{d}(i, j) \rceil, s \rangle$ such that $\mathrm{X}(j'', \lceil \log_5 \mathrm{d}(i, j) \rceil) \supseteq B(j, 5^{\lceil \log_5 \mathrm{d}(i,j) \rceil})$, implying that $r^{\mathrm{area}}_i \leq \lceil \log_5 \mathrm{d}(i, j) \rceil$ and so

$$\sum_{i \in \mathcal{C}^{\mathrm{hi}}_j} 5^{r^{\mathrm{area}}_i} \leq \sum_{i \in \mathcal{C}^{\mathrm{hi}}_j} 5 \cdot \mathrm{d}(i, j). \qquad \blacktriangleleft$$

4 Data Structure

In this section we devise a data structure for the dynamic metric facility location problem that supports insertions and deletions of clients as well as returning (a) the approximate cost of the optimal solution or (b) a set of open facilities that achieves this approximate cost. We achieve this by maintaining the minimum cost solution restricted to pairs in Π. By Theorem 24 this is a $O(1)$ approximation to the cost of the optimal solution.

From the preprocessing phase the algorithm is given the set Π of facility-radius-color triplets, as well as the laminar family of areas $\mathcal{A}$ with its dependency tree $\mathcal{T}$ using the following representation. (1) A two-dimensional array of size $(\rho_{\max} - \rho_{\min} + 1) \times (2^{5\kappa} + 1)$, keeping for each logradius $r \in [\rho_{\min}, \rho_{\max}]$ and color s a list of all the facilities of J_r that share the color s, and (2) the dependency tree $\mathcal{T}$ in a tree data structure. Whenever we use the term *subtree*, *child*, or *descendant* in the following we refer to the dependency tree. (3) For each triplet $v = \langle j, r, s \rangle \in \Pi$, the list `neighbors_above`$(v)$ of all triplets $\langle j', r', s' \rangle$ such that (a) $\langle r', s' \rangle >_{\mathrm{lex}} \langle r, s \rangle$ and (b) $j \in \mathrm{Y}(j', r')$. (4) For each triplet $v = \langle j, r, s \rangle \in \Pi$, the value $f^*_{\langle j,r \rangle}$, which is the minimum opening cost among all facilities in $\mathrm{X}(j, r)$. Using the algorithm in Subsection 4.1 each list `neighbors_above` and each value $f^*_{\langle j,r \rangle}$ can be computed in time $O(2^{O(\kappa)}\Delta)$. Thus, the above data structure can be built in time $O(|J| \cdot 2^{O(\kappa)}\Delta)$.

The algorithm will maintain a dynamic data structure, which can be viewed as an *annotated dependency tree* that keeps for each node $v = \langle j, r, s \rangle$ of $\mathcal{T}$ the following information:

1. three bits $\mathrm{O_x}(v)$, $\mathrm{E_x}(v)$, $\mathrm{A_x}(v)$, which indicate whether the triplet $\langle j, r, s \rangle$ is open, enabled and fulfils the abundance condition, respectively,
2. the number $n_{\mathrm{area}}(v)$ of current clients that belong to the area $A(j, r)$, i.e., $n_{\mathrm{area}}(v) = |\mathcal{C} \cap A(j, r)|$,
3. the number $n_{\mathrm{x}}(v)$ of current clients that belong to $\mathrm{X}(j, r)$, i.e., $n_{\mathrm{x}}(v) = |\mathcal{C} \cap \mathrm{X}(j, r)|$,
4. the number `openbelow`(v) of all open triplets $\langle j', r', s' \rangle$ with $\langle r', s' \rangle <_{\mathrm{lex}} \langle r, s \rangle$ and their corresponding facilities falling within $\mathrm{Y}(j, r)$, i.e.,

$$\texttt{openbelow}(v) = \left| \left\{ \langle j', r', s' \rangle \in J^{\mathrm{open}}_{\mathcal{C}} \,\middle|\, \langle r', s' \rangle <_{\mathrm{lex}} \langle r, s \rangle, j' \in \mathrm{Y}(j, r) \right\} \right|,$$

5. the number $n_{\mathrm{enblbelow}}(v)$ of current clients that belong to areas below that are enabled, i.e.,

$$n_{\mathrm{enblbelow}}(v) = \left| \mathcal{C} \cap \bigcup \left\{ \langle j', r', s' \rangle \in J^{\mathrm{enbl}}_{\mathcal{C}} \,\middle|\, A(j', r') \subset A(j, r) \right\} \right|,$$

6. the value $\mathrm{c}(v) = \sum \{ 5^{r^{\mathrm{area}}_i} \mid i \in \mathcal{C} \cap A(j, r) \}$ (note that with the currently open facilities the cost accrued for the clients in $A(j, r)$ is $O(\mathrm{c}(v))$),
7. the value $y(v)$, which is the cost of the children of v, i.e., $y(v) = \sum_{u \text{ child of } v} \mathrm{c}(u)$.

We next describe the usefulness of the information we keep. Points 1-2 are self-explanatory. Point 3 provides information to test the abundance condition, and thus update the bits in Point 1. Point 4 is useful when deciding whether we should open an area or not. Points 5-7 allow us to efficiently update the cost of the solution.

4.1 Finding all balls containing a given point

In this section we describe a crucial subroutine that we use repeatedly when handling updates. It is given the hierarchy data structure for $\mathcal{T}$, an arbitrary point $p \in V$ and some constant c^* such that $c^* \geq (5/4)c_1$ and returns all balls $\langle j, r\rangle \in \Pi$ that are at distance at most $c^* \cdot 5^r$ from p, i.e., $p \in B(j, c^* \cdot 5^r)$. For $r \in [\rho_{\max}, \rho_{\min}]$, let $S(r)$ denote the set of such balls.

The algorithm FindBalls(p, c^*) performs a top-down traversal of the tree starting at its root $\langle j, \rho_{\max}\rangle$. Note that by the definition of $\rho_{\max}$, all points belong to $B(j, c^* \cdot 5^{\rho_{\max}})$ and $S(\rho_{\max}) = \{\langle j, \rho_{\max}\rangle\}$. For computing $S(r)$, $r = (\rho_{\max} - 1)$, it determines all children of the root to find the pairs $\langle j', r\rangle$ such that the distance of j' and p is at most $c^* \cdot 5^r$. This step is repeated to compute the set $S(\ell)$ for every level of the hierarchy, until we reach the bottom-most level. Finally, we let $S \coloneqq \bigcup\{S(r) : r \in [\rho_{\min}, \rho_{\max}]\}$. A detailed description of this procedure is deferred to the full version.

We next show that the algorithm correctly computes the set $S(r)$, for every log-radius r. Define $\text{children}(S(r)) = \bigcup_{\langle j,r\rangle \in S(r)} \text{children}(j, r)$.

▶ **Lemma 25.** *For each logradius $r \in [\rho_{\max}, \rho_{\min})$ assume $S(r)$ is computed correctly. Then it holds that $S(r-1) \subseteq \text{children}(S(r))$.*

Proof. Assume towards contradiction that there exists $\langle j, r-1\rangle \in S(r-1)$ such that $\mathrm{d}(j, p) \leq c^* \cdot 5^{r-1}$ but $\langle j, r-1\rangle \notin \text{children}(S(r))$. Let $\langle j', r\rangle$ be the parent of $\langle j, r-1\rangle$ in $\mathcal{T}$. By Lemma 2, $\mathrm{d}(j, j') \leq c_1 \cdot 5^r$.

Now, since $S(r)$ is correct, it follows that $\langle j', r\rangle \notin S(r)$, and thus $\mathrm{d}(p, j') > c^* \cdot 5^r$. However, by Lemma 2 we get that $p \in B(j, c^* \cdot 5^{r-1}) \subseteq B(j', c^* \cdot 5^r)$ and thus $\mathrm{d}(p, j') \leq c^* \cdot 5^r$, which is a contradiction. Thus the lemma follows. ◀

Since $c_X \geq (5/4)c_1$, let $c^* = c_X$. We now argue about the running time of FindBalls(p, c^*). Note that for each logradius r, if $p \in B(j, c^* \cdot 5^r)$, then $\mathrm{d}(p, j) \leq c^* \cdot 5^r$. By Lemma 3 and the fact that $c^* \leq 4c_1$ it follows that $|S(r)| \leq 2^{3\kappa}$. Additionally, by Lemma 4 each pair in $S(r)$ has at most $2^{4\kappa}$ children in $\mathcal{T}$ and there are at most Δ different radii. Thus the running time of the algorithm and the size of the output set S are both bounded by $2^{7\kappa} \cdot \Delta$.

▶ **Lemma 26.** *The running time of* FindBalls(p, c^*) *and the size of the output set S are both bounded by $2^{7\kappa} \cdot \Delta$.*

Repeatedly applying the FindBalls subroutine and updating the tree hierarchy in a bottom-up fashion, we can show that insertions and deletions of clients can be handled in $O(2^{O(\kappa^2)} \cdot \Delta^3 \cdot (\kappa^2 + \log \Delta))$ time. Additionally note that under client updates, the value of n will change, which in turn causes $\rho_{\min}$ to either increase or decrease by one. This forces us to either add or delete a bottom-level in the hierarchy, which can be implemented in $O((|J| + |\mathcal{C}|) \cdot 2^{O(\kappa^2)} \cdot \Delta^3 \cdot (\kappa^2 + \log \Delta))$ time. Since such an update is required only after $\Theta(n)$ operations, we get that the amortized time of our algorithm is still bounded by $O(2^{O(\kappa^2)} \cdot \Delta^3 \cdot (\kappa^2 + \log \Delta))$. By employing a standard global rebuilding technique we achieve a worst-case update time, thus proving our main result in Theorem 1. Details on implementing the above steps are deferred to the full version.

References

1. Sebastian Abshoff, Peter Kling, Christine Markarian, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. Towards the price of leasing online. *Journal of Combinatorial Optimization*, 4(32):1197–1216, 2015.
2. Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. In *Symposium on Discrete Algorithms (SODA)*, pages 708–721, 2015.
3. Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Information and Computation*, 194(2):175–202, 2004.
4. Barbara Anthony and Anupam Gupta. Infrastructure leasing problems. *International Conference on Integer Programming and Combinatorial Optimization*, pages 424–438, 2007.
5. Pierre Chardaire, Alain Sutter, and Marie-Christine Costa. Solving the dynamic facility location problem. *Networks*, 28(2):117–124, 1996.
6. Artur Czumaj, Christiane Lammersen, Morteza Monemizadeh, and Christian Sohler. $(1+\epsilon)$-approximation for facility location in data streams. In *Symposium on Discrete Algorithms (SODA)*, pages 1710–1728, 2013.
7. Zvi Drezner and Horst W Hamacher. *Facility location: applications and theory*. Springer Science & Business Media, 2001.
8. David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 459–470, 2014.
9. Reza Zanjirani Farahani, Maryam Abedian, and Sara Sharahi. Dynamic facility location problem. In *Facility Location*, pages 347–372. Springer, 2009.
10. Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007.
11. Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.
12. Dimitris Fotakis. Memoryless facility location in one pass. *ACM Trans. Algorithms*, 7(4):49:1–49:24, 2011.
13. Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.
14. Monika Henzinger, Dariusz Leniowski, and Claire Mathieu. Dynamic clustering to minimize the sum of radii. In *European Symposium on Algorithms (ESA)*, 2017.
15. Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Symposium on Theory of Computing (STOC)*, pages 373–380, 2004.
16. Christiane Lammersen and Christian Sohler. Facility location in dynamic geometric data streams. In *European Symposium on Algorithms (ESA)*, pages 660–671, 2008.
17. Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.
18. Adam Meyerson. Online facility location. In *Symposim on Foundations of Computer Science (FOCS)*, pages 426–431, 2001.
19. Chandrashekhar Nagarajan and David P Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013.
20. Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 240–257, 2002.
21. George O Wesolowsky. Dynamic facility location. *Management Science*, 19(11):1241–1248, 1973.