Facility Location in the Sublinear Geometric Model

Morteza Monemizadeh ✉
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

In the *sublinear geometric model*, we are provided with an oracle access to a point set P of n points in a bounded discrete space $[\Delta]^2$, where $\Delta = n^{O(1)}$ is a polynomially bounded number in n. That is, we do not have direct access to the points, but we can make certain types of queries and there is an oracle that responds to our queries. The type of queries that we assume we can make in this paper, are range counting queries where ranges are axis-aligned rectangles (that are basic primitives in database [36, 11, 17], computational geometry [1, 2, 6, 5], and machine learning [35, 31, 29, 28]). The oracle then answers these queries by returning the number of points that are in queried ranges. Let Alg be an algorithm that (exactly or approximately) solves a problem $\mathcal{P}$ in the sublinear geometric model. The query complexity of Alg is measured in terms of the number of queries that Alg makes to solve $\mathcal{P}$. In this paper, we study the complexity of the (uniform) Euclidean facility location problem in the sublinear geometric model. We develop a randomized sublinear algorithm that with high probability, $(1+\epsilon)$-approximates the cost of the Euclidean facility location problem of the point set P in the sublinear geometric model using $\tilde{O}(\sqrt{n})$ range counting queries. We complement this result by showing that approximating the cost of the Euclidean facility location problem within $o(\log(n))$-factor in the sublinear geometric model using the sampling strategy that we propose for our sublinear algorithm needs $\tilde{\Omega}(n^{1/4})$ RangeCount queries. We leave it as an open problem whether such a polynomial lower bound on the number of RangeCount queries exists for any randomized sublinear algorithm that approximates the cost of the facility location problem within a constant factor.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Facility Location, Sublinear Geometric Model, Range Counting Queries

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.6

Category APPROX

1 Introduction

In the *sublinear query model*, we consider scenarios in which we do not have direct access to the underlying data, but there is an oracle or a prophet who provides us *metadata* about the data that we are interested in. An interesting real-world application of such scenarios is for retail or ride hailing companies that have millions of customers. Often these giant companies do outsourcing by hiring a party outside a company to perform temporary services to reduce costs or optimize customers buying experience. Due to privacy concerns[1], these companies do not want to provide accurate and detailed information of their customers to the outsourced companies. However, they can provide some sort of metadata (such as how many customers are in an area or a neighborhood or how often customers in a town purchase particular goods or use a service) that can help the outsourced company to perform its service.

[1] See "Threat to Security and Confidentiality" in `https://www.thebalancesmb.com/top-outsourcing-disadvantages-2533777` and "Security and Privacy Concerns" in `https://www.truppglobal.com/blog/top-10-outsourcing-problems-and-how-to-solve-them#5-security-and-privacy-concerns`

As a concrete example, let us consider a retail company that plans to open new stores or lockers to provide fast and responsive support for its customers. To this end, the company hires an (experienced) outsourced company to estimate the total cost of opening new stores or lockers. Due to privacy concerns, the company does not want to provide information about its customers, but it can provide aggregate data such as how many customers are in an area or a neighborhood. An interesting question is how many times these aggregate data should be provided so that the outsourced company can estimate a fairly accurate cost of opening new stores or lockers.

Motivated by these applications, we study the (uniform) Euclidean facility location in the *sublinear geometric model* which is a sublinear model suitable when the underlying data is a point set in a d-dimensional Euclidean space. In particular, we seek to design sublinear algorithms that approximate the cost of the Euclidean facility location problem when we do not have access to the underlying point set, but instead, we can make queries and there is an oracle who will respond and provide solutions to our queries. Next, we define the facility location problem formally.

▶ **Definition 1** ((uniform) Euclidean facility location)**.** *In the* (uniform) Euclidean facility location *problem, we are given a point set P of size n in a bounded discrete space $[\Delta]^2$, where $\Delta = n^{O(1)}$ is a polynomially bounded number in n and an opening cost $f > 0$, and the goal is to compute a set $F^* \subset [\Delta]^2$ of facilities that minimizes the cost function $cost_{FL}(P, F^*) = \sum_{p \in P} \text{dist}(p, F^*) + f \cdot |F|$, where $dist(p, F^*) = \min_{q \in F^*} dist(p, q)$ is the Euclidean distance of p to its nearest facility in F^*. We denote the optimal facility location cost of the point set P by $OPT_{FL}(P, f)$.*

1.1 Sublinear geometric model

Let $P \subseteq [\Delta]^2$ be a point set of size n in a 2-dimensional discrete space $[\Delta]^2 = \{1, 2, 3, \cdots, \Delta\}^2$, where $\Delta = n^{O(1)}$ is a polynomially bounded number in n. For the simplicity of exposition, we assume that $\Delta = n$. This means that the space $[\Delta]^2$ is indeed, the space $[n]^2$. We also assume that every unit square (of side length one), can store at most one (weighted) point [2].

In the *sublinear geometric model*, we assume we do not have access to points of P directly, but instead, we have *query access* to points of P. That is, we can make certain queries with respect to point set P and there is an oracle that returns the solutions to our queries. The type of queries that we assume are provided in the sublinear geometric model are *range counting queries* where ranges are *axis-aligned rectangles* (that are basic primitives in database [36, 11, 17], computational geometry [1, 2, 6, 5], and machine learning [35, 31, 29, 28]). The oracle then answers these queries by returning the number of points in queried ranges.

Range counting queries. We assume that we have a query access to P where we query an axis-aligned rectangle $c \subseteq [n]^2$ and the oracle returns the number of points $n_c = |P \cap c|$ that are in rectangle c. We denote such a query by RangeCount(c). Let Alg be an algorithm that (exactly or approximately) solves a problem $\mathcal{P}$ in the sublinear geometric model. The query complexity of Alg is measured in terms of the number of RangeCount(c) queries that Alg makes to solve $\mathcal{P}$. Assume that the input of problem $\mathcal{P}$ is a point set P of size n. If the number of RangeCount(c) queries that we ask is $o(n)$, we say the query complexity of problem $\mathcal{P}$ is sublinear in terms of its input set.

[2] We make these explicit assumptions to simplify the notations in this paper.

1.2 Our Contribution

Here, we state our main result.

▶ **Theorem 2** (Facility location in sublinear geometric model). *Let P be a point set of size n in a discrete space $[n]^2$. Let $f > 0$ be the opening cost for the facility location problem and $0 < \epsilon \leq 1$ be the error parameter. Then, there exists a randomized sublinear algorithm that (w.h.p.) returns an $(1+\epsilon)$-estimator for the facility location cost of P in the sublinear geometric model using $\tilde{O}(\sqrt{n})$ RangeCount queries.*

We complement this result by showing that approximating the cost of the Euclidean facility location problem within $o(\log(n))$-factor in the sublinear geometric model using the sampling strategy that we propose for Theorem 2 needs $\tilde{\Omega}(n^{1/4})$ RangeCount queries. See Figure 2 in Section 4 for the illustration of the hard instance that we explain there. We leave it as an open problem whether such a polynomial lower bound on the number of RangeCount queries exists for any randomized sublinear algorithm that approximates the cost of the facility location problem within a constant factor.

▶ **Lemma 3** (Lower bound). *There exists a facility location instance for which the sublinear algorithm of Theorem 2 needs $\Omega(\frac{\sqrt[4]{n}}{\log n})$ RangeCount queries to estimate the cost of facility location within $o(\log n)$-factor in the sublinear geometric model.*

Outline of the proof of Theorem 2. Our starting point to prove this theorem is to choose a known polynomial-time approximation scheme (PTAS) for the Euclidean facility location problem in the plane. Our goal would be to simulate such a PTAS in sublinear time to obtain an estimator for the optimal cost of the facility location problem.

Known PTAS algorithms [7, 30, 15] are often based on partitioning the underlying space of a point set. Two such PTAS algorithms are known. The first one was proposed by Arora, Raghavan, and Rao [7] and later improved by Kolliopoulos and Rao [30] that partitions the space into regions and then combine solutions of small regions using the dynamic programming to obtain solutions for the bigger regions. The issue with extracting an estimator based on this PTAS is that simulating the dynamic programming in sublinear time seems to be hard.

The second PTAS for this problem was proposed by Czumaj, Lammersen, Monemizadeh, and Sohler [15] where they partition the underlying space $[n]^2$ of a point set P into a set Λ of cells in a way that they can solve the facility location instance inside each cell independent of the facility location instances of the other cells. The independency that is provided by this algorithm is a good choice for us. However, the problem with using this approach (in order to develop an estimator) in the geometric sublinear model is we do not have access to Λ.

Overall Idea: We develop a sublinear algorithm that randomly samples cells of a number of grids (of exponentially increasing granularity) that we impose on the discrete space $[n]^2$ and determines whether the sampled cells could be cells in the set Λ or not. For those sampled cells that are in fact, cells in Λ, we solve the facility location instance independently and compute a $(1+\epsilon)$-approximation of their facility location costs. We then multiply their costs with proper weights to $(1+\epsilon)$-estimate the optimal facility location cost $OPT_{FL}(P, f)$.

Next we briefly describe the space partitioning and the PTAS that are developed in [15] and then, we explain our sublinear algorithm in detail.

PTAS of [15]. Suppose we are given a point set $P \subset [n]^2$ of size n. We impose a (nested) grid set $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ on the space $[n]^2$ where the grid $\mathcal{G}_i$ consists of cells of side length 2^i. We then randomly shift the border lines of the grids $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ as follows. We choose two random real numbers $v_1, v_2 \in [0, n]$. Then, we shift every border line ℓ of every grid $\mathcal{G}_i$ using the random vector $v = (v_1, v_2)$. Observe that after this random shifting process, the point set P is enclosed in a rectangle of side length $[2n]^2$. In the sublinear geometric model, whenever we make range queries for ranges that are axis-aligned rectangles, we first shift these rectangles using v and then query them.

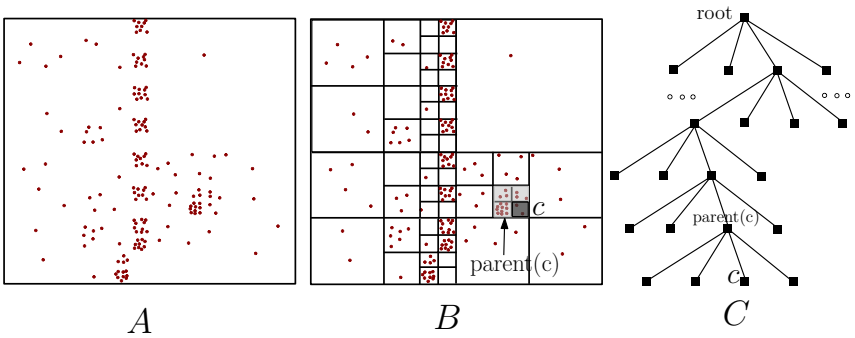

Figure 1 In the sub-figure A, a point set $P \subseteq [\Delta]^2$ is given. The sub-figure B illustrates the space partitioning Λ that is computed by the PTAS of [15]. A light cell c (dark gray square) that is in the partition set Λ and its parent *parent*(c) (gray square) are shown. In the sub-figure C, the nodes that correspond to c and *parent*(c) are also shown in the corresponding quadtree partitioning.

The PTAS [15] is based on partitioning heavy cells and detecting light cells, two notions that we define next. Let c be an arbitrary cell in a grid $\mathcal{G}_i$. We use the 3-approximation algorithm due to Jain and Vazirani [26] (who developed it for facility location) to determine if c is heavy or light. Let us call this algorithm $\textsc{Alg}^3_{FL}$. We say an arbitrary c in a grid $\mathcal{G}_i$ is a *heavy cell* if $\textsc{Alg}^3_{FL}(P \cap c)$ outputs that the facility location cost of $P \cap c$ is at least $\delta_{FL} \cdot f$ where $\delta_{FL} = O(\epsilon^{-2} \log^2 n)$. If the reported cost is less than $\delta_{FL} \cdot f$, c is a *light* cell.

Starting with the square $[2n]^2$, if a cell $c \in \mathcal{G}_i$ is heavy, we then split it into its 4 children c_1, c_2, c_3, c_4 (that are in the grid $\mathcal{G}_{i-1}$) of equal side length and recurse for those children that are heavy. At the end of this recursive algorithm, we let Λ be the set of all light cells that are constructed in this way. Interestingly, for every cell c, we can solve the facility location instance of the point subset $P \cap c$ independently. Later, we combine the solutions of these independent instances to obtain a $(1+\epsilon)$-approximate solution of the optimal facility location of P. See Figure 1 for an illustration of Λ.

Sublinear algorithm. Now, we outline our sublinear algorithm. This algorithm samples cells of the grids $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ randomly and determine whether the sampled cells can be in set Λ or not. If we sample enough cells from Λ uniformly at random, compute a $(1+\epsilon)$-approximation of their costs and multiply their costs with proper weights, we can approximate the optimal facility location cost $OPT_{FL}(P, f)$ within $(1+\epsilon)$-factor. However, this is not an easy task in the sublinear geometric model as we have the following challenges:

- **Query Access:** Given a cell c in a grid $\mathcal{G}_i$:
  - We need to determine if c is heavy or light using $\textsc{Alg}^3_{FL}(P \cap c)$.
  - If we know that we have sampled a cell $c \in \Lambda$, we need to compute a $(1+\epsilon)$-approximation of its cost using a PTAS algorithm, say [15].

  However, we do not have direct access to the points in $P \cap c$. We can only make range counting queries.
- **Many noisy empty cells:** In any grid $\mathcal{G}_i$ for $i \leq \frac{1}{2} \cdot \log(n)$, we have $\Omega(n)$ cells and plenty of them could be empty. Recall that $|P| = n$. Thus, a uniform random sampling may need to sample many cells to hit a cell from Λ.

We first develop a tester algorithm (so-called HeavyTester) that using $\text{poly}(\epsilon^{-1} \cdot \log(n))$ RangeCount queries determines if a cell c in a grid $\mathcal{G}_i$ is a heavy or a light cell. In case that c is a light cell, the tester returns the facilities that $(1+\epsilon)$-approximates the optimal facility location cost of the cell c. This already resolves the first challenge. Next, we deal with the second challenge. We first develop a sublinear algorithm (so-called $\sqrt{n}$-Estimator) that using $\tilde{O}(\sqrt{n})$ RangeCount queries distinguishes between the following two cases:

- **Low cost instances:** The first case is if the optimal cost of P is upper-bounded by $O(\sqrt{n}f)$. If this is indeed the case, the algorithm $\sqrt{n}$-Estimator solves the facility location instance of P and returns $(1+\epsilon)$-approximate solution (not the cost) of P.
- **High cost instances:** The second case is if the optimal cost of P is $\Omega(\sqrt{n}f)$. For this case, the algorithm $\sqrt{n}$-Estimator outputs that the cost of P is $\Omega(\sqrt{n}f)$.

From now on, we assume that the optimal facility location cost of P is $\Omega(\sqrt{n}f)$.

We develop a **telescoping sampling** that samples any arbitrary cell c in a grid $\mathcal{G}_i$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, where $parent(c)$ is the parent of c in the grid $\mathcal{G}_{i+1}$ and $n_{parent(c)} = |parent(c) \cap P|$ is the number of points in $parent(c)$ (See Figure 1 for the illustration of a light cell and its parent.) The reason that we do the telescoping sampling based on $n_{parent(c)}$ not $n_c = |c \cap P|$ is that the cells in Λ are light, but their parents (and ancestors) are heavy, so heavy parents have a minimum number of points that helps us to use known concentration bounds [3, 32] to analyze the telescoping sampling.

For a grid $\mathcal{G}_i$, we implement the telescoping sampling as follows. We start with the unique cell c' in grid $\mathcal{G}_{\log(n)+1}$ and split it into 4 sub-cells c_1, c_2, c_3, c_4 and query the number of points inside each of them. Then, we sample the sub-cell c_j for $j \in [4]$ with probability $\frac{n_{c_j}}{n_{c'}}$. Suppose we sample c_1. We let $c' = c_1$ and recursively repeat the same process till we end up with the parent of a cell $c \in \mathcal{G}_i$. By the telescoping argument, a cell $c \in \mathcal{G}_i$ is sampled with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$.

Once we sample a cell c in a grid $\mathcal{G}_i$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, we test if c is a heavy or a light cell using algorithm HeavyTester(c). If c is heavy or both c and $parent(c)$ are light, we do not do anything. If c is a light cell and its parent is a heavy cell, the cell c must be in Λ. In this case, the tester returns set F_c of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of the instance $c \cap P$. Let $\text{cost}^{\epsilon}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \text{dist}(p, F_c) + |F_c| \cdot f$ be the facility location cost of c with respect to facility set F_c.

Next, we would like to assign a weight to the sampled cell c that is in Λ. For that, one option would be to multiply $\text{cost}^{\epsilon}_{FL}(c, F_c, f)$ by the term $\frac{n}{n_{parent(c)}}$. However, this may not be possible. The problem is $n_{parent(c)}$ might be very small and so, the weight $\frac{n}{n_{parent(c)}}$ may be $\Omega(n)$ which essentially means if we want to use known concentration bounds such as the Hoeffding bound [22], we need $\Omega(n)$ samples to approximate the optimal cost $OPT_{FL}(P, c)$.

We develop a novel sampling technique that samples (almost uniformly at random) those light cells in a grid $\mathcal{G}_i$ whose parents have roughly the same number of points. To this end, we first observe that from set Λ, we only need to consider those cells whose cost is at least

τf where $\tau = O(\frac{\epsilon}{\log(n)})$. We call these cells *significant light cells* and we let Ψ be the set of significant light cells of Λ. We show that in order to develop a $(1+\epsilon)$-estimator for facility location, we can safely ignore insignificant light cells (those with cost less than τf) of Λ.

We then partition Ψ into likelihood classes Ψ_i^j, where Ψ_i^j is the set of those significant light cells that are in the grid $\mathcal{G}_i$ and their parents have at least $(1+\epsilon)^j$ points and less than $(1+\epsilon)^{j+1}$. We consider only those likelihood classes Ψ_i^j whose size is $|\Psi_i^j| \geq \beta\sqrt{n}$ where $\beta = (\frac{\epsilon}{\log(n)})^{O(1)}$. We call these classes *contributing likelihood classes*. We show that non-contributing likelihood classes can be safely ignored.

We follow the steps below to show that if a likelihood class Ψ_i^j is contributing, we can approximate the facility location cost of it within $(1+\epsilon)$-factor.

- **Step 1:** We first show that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, cell c is sampled almost uniformly at random from the set Ψ_i^j.
- **Step 2:** Next, we prove that with high probability, we sample at least $\text{poly}(\epsilon^{-1}\log(n))$ significant light cells from Ψ_i^j.
- **Step 3:** We finally develop an estimator that $(1+\epsilon)$-approximates the size of Ψ_i^j.

Putting everything together, we obtain an estimator that $(1+\epsilon)$-approximates the optimal cost $OPT_{FL}(P, f)$ in the sublinear geometric model using $\tilde{O}(\sqrt{n})$ range counting queries.

1.3 Related Work

Sublinear algorithms have been studied extensively for problems in metric spaces. Czumaj and Sohler [16] showed that we can $(1+\epsilon)$-approximate the weight of minimum spanning tree in metric space in time $\tilde{O}(n) = O(n\text{poly}(\epsilon^{-1}\log n))$. Badoiu, Czumaj, Indyk, and Sohler [8] showed that we can approximate the cost of facility location in metric space within constant factor in time $\tilde{O}(n)$, and Indyk [23] showed that we can return a constant factor approximation of the k-median problem in metric spaces in time $\tilde{O}(n)$. Interestingly, all sublinear algorithms that we are aware of in metric spaces have $\tilde{O}(n)$ running times. One may ask if there exists any sublinear algorithm that is truly sublinear in n?

For sparse graphs this is indeed possible. Chazelle, Rubinfeld, and Trevisan [12] in a seminal work showed that given a connected graph G (that is represented by adjacency lists) of average degree $\overline{d}$ with edge weights in the set $\{1, ..., w\}$, we can $(1+\epsilon)$-approximate the weight of the minimum spanning tree (MST) of G in time $O(\overline{d}w\epsilon^{-2}\log(\frac{\overline{d}w}{\epsilon}))$. Observe that a graph with average degree $\overline{d}$ is a sparse graph as it can have at most $O(n\overline{d})$ edges. Later, Nguyen and Onak [33] in another seminal work initiated the study of sublinear algorithms for Vertex Cover, Maximum Matching, Maximum Weight Matching, and Set Cover problems in bounded-degree graphs. As an example, for the problem of estimating the size of maximum matching, they showed how to approximate the size of the maximum matching up to an additive error ϵn in time $2^{d^{O(1/\epsilon)}}$, where d is the maximum degree of a vertex in the graph G. They obtained similar bounds for the aforementioned problems that approximate their size or cost up to an (ϵn)-additive term. Yoshida, Yamamoto, and Ito[38] and then Behnezhad [10] improve the running time (in fact, the query complexity) of Nguyen and Onak's algorithm for estimating the size of maximum matching and minimum vertex cover. As an example, Behnezhad showed that we can approximate the size of maximal matching (i.e., 2-approximate maximum matching) to within (ϵn)-additive error in time $\tilde{O}((\overline{d}+1)/\epsilon^2)$.

What do we know for Euclidean spaces? Interestingly, very few sublinear algorithms have been developed for optimization problems in Euclidean spaces. Indeed, we are aware of two sublinear algorithms for problems in Euclidean spaces. The first work is due to Czumaj, Ergün, Fortnow, Magen, Newman, Rubinfeld, and Sohler [14] who studied the problem of

approximating the weight of Euclidean minimum spanning trees (EMST) in a sublinear time. The authors show that if we are provided with oracle access to basic data structures in the Euclidean space, we can estimate the weight of a EMST within $(1+\epsilon)$-factor.

In particular, they assume that we do not have access to a point set, but we can make queries and there is an oracle that answers our queries. The type of queries that they are allowed to make are *minimum bounding box* queries, *range* queries and *approximate nearest neighbor* queries. They show that in order to $(1+\epsilon)$-approximate the weight of the Euclidean minimum spanning tree in the plane (i.e., $\mathbb{R}^2$), we need to make $\tilde{O}(\sqrt{n})$ such queries.

The second sublinear algorithm that we are aware is for the minimum Euclidean bichromatic matching due to Indyk [25]. This algorithm is more of a linear-time constant factor approximation algorithm than a sublinear algorithm. Indyk assumed that we have access to points, but the main issue is solving the minimum Euclidean bichromatic matching problem using classical algorithms known for the maximum matching problem, say *Hungarian method* takes $O(n^3)$ time. (See [37]). He showed we can approximate the cost of minimum bichromatic matching within constant factor in near linear time and he showed that in time $\tilde{O}(n)$, we are able to do that. Later, Raghvendra and Agarwal [34] showed that in time $\tilde{O}(n)$ we can in fact, compute $(1+\epsilon)$-approximate Euclidean bichromatic matching.

The sublinear geometric model that we study in this paper is closely related to the dynamic geometric streaming model [24, 19] and the Massively Parallel Computations (MPC) model [27, 20]. In the the dynamic geometric streaming model [24, 19], Frahling and Sohler [19] in a groundbreaking work showed that given a (polynomially in n bounded) stream of insertions and deletions of points from an underlying space $[\Delta]^2$ (where $\Delta = n^{O(1)}$), we can compute a $(1+\epsilon)$-approximate solution for the k-median, k-means, MaxCut, maximum travelling salesperson, maximum spanning tree and average distance problems using $\text{poly}(k\epsilon^{-2}\log(n))$ space. Frahling, Indyk and Sohler [18] studied the problem of approximating the weight of the Euclidean minimum spanning tree within $(1+\epsilon)$ in the dynamic geometric streaming model. Later, Czumaj, Lammersen, Monemizadeh, and Sohler [15] developed a $(1+\epsilon)$-estimator for the cost of the Euclidean facility location using $\text{poly}(\epsilon^{-1}\log(n))$ space.

The Massively Parallel Computations (MPC) model was first introduced by Karloff, Suri, and Vassilvitskii [27] and later extended by [9, 20]. Andoni, Nikolov, Onak, Yaroslavtsev [4] studied the Euclidean minimum spanning tree and the minimum Euclidean bichromatic matching problems in the MPC model. They showed $O(1)$-round MPC $(1+\epsilon)$-algorithm for the Euclidean minimum spanning tree that uses $\tilde{O}(\sqrt{n})$ machines each one having local space $\tilde{O}(\sqrt{n})$. For the minimum Euclidean bichromatic matching problem, they showed that we can approximate the cost of minimum bichromatic matching within $(1+\epsilon)$-factor using similar number of communication rounds, number of machines, and the space per machine.

2 Preliminaries

Rounding notations. For the sake of simplicity, we assume that the logarithm of a number is always rounded down or up in an appropriate manner. As an example, if we want to use a range $[\lfloor\log(a)\rfloor, \cdots]$ or a range $[\cdots, \lceil\log(a)\rceil]$ where $a \in \mathbb{R}$, we write $[\log(a), \cdots]$ and $[\cdots, \log(a)]$ to simplify the notation, respectively.

Randomly shifted grids. Let $P \subseteq [n]^2$ be a point set of size n. We consider $\log(n)+1$ grids $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ where the grid $\mathcal{G}_i$ consists of cells of side length 2^i that we impose on the space $[n]^2$. We randomly shift the border lines of the grids as follows. We choose two random real numbers $v_1, v_2 \in [0, n]$ and we let $v = (v_1, v_2)$ be the random vector. Then, we shift every border line ℓ of every grid $\mathcal{G}_i$ using v. Observe that after this

shift process, the point set P is enclosed in a rectangle of side length $[2n]^2$. In the sublinear geometric model, whenever we make range queries for ranges that are axis-aligned rectangles, we first shift these rectangles using v and then query them.

We let $\mathcal{G}_{\geq i} = \{\mathcal{G}_i, \mathcal{G}_{i+1}, \cdots, \mathcal{G}_{\log(n)}\}$ and $\mathcal{G}_{\leq i} = \{\mathcal{G}_i, \mathcal{G}_{i-1}, \cdots, \mathcal{G}_0\}$. Let $c \in \mathcal{G}_i$ be a cell in a grid $\mathcal{G}_i$. We denote by $c \cap P$ the subset of points that are inside the cell c or on the border lines of c. We denote the number of points in c by $n_c = |c \cap P|$. We let ℓ_c be the side length of c which is 2^i in the grid $\mathcal{G}_i$. We denote the parent of c which is in the grid $\mathcal{G}_{i+1}$ by *parent*(c). Let $c' \in \mathcal{G}_{\geq i}$ be a cell that contains c (i.e., $c \subseteq c'$). We say c' is an *ancestor* of c. We denote the ancestor of c at a grid $\mathcal{G}_{j \geq i}$ by ancestor(c, j). If $c' \in \mathcal{G}_{>i}$, we say that c' is a *proper ancestor* of c. Similarly, let $c' \in \mathcal{G}_{<i}$ be a cell that is contained in the cell c. We say c' is a *descendant* of c. We denote the descendant of c at a grid $\mathcal{G}_{j<i}$ by *descendant*(c, j).

2.1 Space partitioning and PTAS of [15]

Let $P \subseteq [2n]^2$ be a point set of size n and $f > 0$ be a positive real number. Let $0 < \epsilon \leq 1$ be the error parameter. As we explained in the introduction of this paper, Czumaj, Lammersen, Monemizadeh, and Sohler [15] show a construction for the facility location problem that partitions the space $[2n]^2$ into a set Λ of disjoint cells for which we can solve the facility location problem independently. We denote this algorithm by $\textsc{Alg}^{\epsilon}_{FL}(P)$ whose pseudocode is given below.

Algorithm 1 $\textsc{Alg}^{\epsilon}_{FL}$ - Czumaj, Lammersen, Monemizadeh, and Sohler [15].

Data: A point set $P \subseteq [n]^2$, an opening cost $f > 0$, and a parameter $0 < \epsilon \leq 1$.
Let c be the square $[2n]^2$. Let $\Gamma = \{c\}$ and $\Lambda = \emptyset$;
while Γ *has a heavy cell* c **do**
  Let $\Gamma = \Gamma \backslash \{c\}$;
  Let c_1, c_2, c_3, c_4 be children of c ; `/* If` $c \in \mathcal{G}_i$`, its children are in` $\mathcal{G}_{i-1}$ `*/`
  for $c_i \in \{c_1, c_2, c_3, c_4\}$ **do**
    if c_i *is a heavy cell* **then**
      $\Gamma = \Gamma \cup \{c_i\}$;
    else
      $\Lambda = \Lambda \cup \{c_i\}$; `/*` c_i `is light */`
for $c \in \Lambda$ **do**
  Let F_c be the set of facilities returned by the PTAS [30] for the point set $P \cap c$;
  Let $\text{cost}^{\epsilon}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \text{dist}(p, F_c) + |F_c| \cdot f$ be the cost of $c \cap P$;
return Set Λ and for every cell $c \in \Lambda$, set F_c of opened facilities and $\text{cost}^{\epsilon}_{FL}(c, F_c, f)$;

The PTAS [15] is based on partitioning heavy cells and detecting light cells, two notions that we define them next. Let c be an arbitrary cell in a grid $\mathcal{G}_i$. We use the 3-approximation algorithm due to Jain and Vazirani [26] to determine if c is heavy or light.

Heavy and light cells. Let c be a cell in a grid $\mathcal{G}_i$. Let $\delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2$. Let F_c be the set of facilities returned by $\textsc{Alg}^{3}_{FL}(P \cap c)$. We say c is a *heavy cell* if $\text{cost}^{3}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \text{dist}(p, F_c) + |F_c| \cdot f \geq \delta_{FL} \cdot f$; otherwise it is a *light* cell.

▶ **Lemma 4** ([15]). *Let $P \subseteq [2n]^2$ be a point set of size n and $f \in \mathbb{R}^+$ be a positive real number. Let $0 < \epsilon \leq 1$ be the error parameter. For the output of $\textsc{Alg}^{\epsilon}_{FL}(P)$ we have*

- *For every cell $c \in \Lambda$, we have $cost^{\epsilon}_{FL}(c, F_c, f) \leq (1 + \epsilon) \cdot OPT_{FL}(P \cap c, f)$.*
- *$(1 - \epsilon) \cdot OPT_{FL}(P, f) \leq cost^{\epsilon}_{FL}(\Lambda, f) = \sum_{c \in \Lambda} cost^{\epsilon}_{FL}(c, F_c, f) \leq (1 + \epsilon) \cdot OPT_{FL}(P, f)$.*

3 Sublinear algorithm for facility location problem

In this section, we develop our sublinear algorithm for the facility location problem and prove Theorem 2. We break this algorithm into four steps as follows:

1. **Heavy tester:** We first develop a tester algorithm for heavy cells.
2. **Telescoping sampling:** Then, we explain the telescoping sampling.
3. **$O(\sqrt{n})$-estimator:** In the third step, we explain a $\sqrt{n}$-estimator for the Euclidean facility location problem in the sublinear geometric model.
4. **$(1+\epsilon)$-estimator:** In the final step, we describe our $(1+\epsilon)$-estimator in the sublinear geometric model and analyze it.

3.1 A tester algorithm for heavy cells

Here, we develop a tester algorithm in the sublinear geometric model that given an arbitrary cell c in a grid $\mathcal{G}_i$, makes $\text{poly}(\epsilon^{-1}\cdot\log(n))$ RangeCount queries to distinguish between the case that c is a heavy cell or a light one. The proof of the following lemma is in Appendix A.

▶ **Lemma 5** (Heavy Tester). *Let c be a cell in a grid $\mathcal{G}_i$. Then, there exists a deterministic tester algorithm that we call HeavyTester(c) so that*

- ***Testing heaviness:*** *It makes $O(\epsilon^{-8}\cdot\log^6(n))$ RangeCount queries to determine if the cell c is heavy or light.*
- ***Approximating a solution:*** *If c is a light cell in set Λ, then HeavyTester(c) returns a set F_c of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of c.*

3.2 $O(\sqrt{n})$-approximate sublinear algorithm

Now, we explain an algorithm that distinguishes between the case that the facility location cost of P is at most $\sqrt{n}\cdot f$ or it is greater than $\sqrt{n}\cdot f$. For the former case, we return $(1+\epsilon)$-approximate solution and for the latter, we obtain a lower bound $\sqrt{n}\cdot f$ for the cost.

Algorithm 2 $\sqrt{n}$-Estimator.

Data: The discrete space $[2n]^2$, an opening cost $f>0$, and an error parameter $0<\epsilon\leq 1$.
Result: A $\sqrt{n}$-estimator $\mathcal{Z}$ for the facility location cost of a point set $P\subseteq[2n]^2$.

```
Let c be the square [2n]^2 and let Γ be an empty set. Let H = {c};
At any time, let cost^ε_FL(Γ, f) = Σ_{c'∈Γ} cost^ε_FL(c, F_c, f), where F_c is the set of
  facilities that the randomized algorithm [15] Alg^ε_FL returns for each cell c ∈ Γ;
while cost^ε_FL(Γ, f) ≤ √n · f and H ≠ ∅ do
    Take an arbitrary cell c ∈ H and delete it from H;
    if HeavyTester(c) returns that c is a heavy cell then
        Let C_c = {c_1, c_2, c_3, c_4} be the four children of c in the grid G_{i-1};
        Let H = H ∪ C_c;
    else
        Add c to the set Γ;
Return Γ;
```

Lemma 6 is similar to Lemma 22 and we explain Lemma 22 in detail later.

▶ **Lemma 6.** *Let $P \subseteq [\Delta]^2$ be a point set of size n. Let $f > 0$ be the opening cost and $0 < \epsilon \leq 1$ be the error parameter. Then, the sublinear algorithm $\sqrt{n}$-*Estimator *uses $O(\epsilon^{-9} \log^7(n) \cdot \sqrt{n})$ RangeCount queries and distinguishes between the following cases:*

- *If the optimal facility location cost of P is $O(\sqrt{n} \cdot f)$, then this sublinear algorithm returns a set of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of P.*
- *If the optimal facility location cost of P is $\Omega(\sqrt{n} \cdot f)$, then this sublinear algorithm $O(\sqrt{n})$-approximates the optimal facility location cost $OPT(P, f)$ of P. Moreover, it finds an $\Omega(\sqrt{n} \cdot f)$ lower-bound for the optimal facility location cost $OPT_{FL}(P, f)$ of P.*

3.3 Telescoping sampling

In this section we develop a sampling mechanism that samples a cell c with probability $p_c = \frac{n_c}{n}$. This will be used as a basic primitive that we later use for our $(1+\epsilon)$-estimator.

Algorithm 3 TelescopingSampling.

Data: A grid $\mathcal{G}_i$
Result: A sampled cell $c' \in \mathcal{G}_i$ such that the probability that any cell $c'' \in \mathcal{G}_i$ is c' is $\mathbf{Pr}[c' = c''] = \frac{n_{c''}}{n}$.

1 Let c be the cell in the grid $\mathcal{G}_{\log(n)}$ and let $j = \log(n)$;
2 **while** $j > i$ **do**
3 Split c into its four sub-cells c_1, c_2, c_3, c_4 that are in the grid $\mathcal{G}_{j-1}$ and query the number of points inside each of them using the subroutine RangeCount;
4 Sample the sub-cell c_i for $i \in [4]$ with probability $\frac{n_{c_i}}{n_c}$;
5 Let c' be the cell that is sampled in Step 4; `// c' is one of` c_1, c_2, c_3, c_4.
6 Let $c = c'$ and $j = j - 1$;
7 Return c;

The proofs of the following lemmas are given in Appendix B.

▶ **Lemma 7.** *Assume that we invoke Subroutine* TelescopingSampling$(\mathcal{G}_i)$. *Then, a cell c that is returned by this subroutine is sampled with the probability that $\mathbf{Pr}[c] = \frac{n_c}{n}$.*

▶ **Lemma 8.** *Given a grid $\mathcal{G}_i$, the number of RangeCount queries that we make in Subroutine* TelescopingSampling$(\mathcal{G}_i)$ *is $O(\log n)$.*

3.4 $(1+\epsilon)$-approximate sublinear algorithm

In this section, we prove Theorem 2. We first give the pseudocode of our main sublinear algorithm. Next, we prove the correctness of our algorithm and analyze its query complexity.

3.5 Analysis

Let P be a point set of n points in discrete space $[2n]^2$. Now, for the sake of the analysis, assume that we run the following two processes.

- In the first process, for point set P, we run the quadtree construction (for the facility location problem) that we presented in Lemma 4 and it returns a set Λ of light cells.
- In the second process, given RangeCount query access to point set P, we invoke Algorithm 4 $(1+\epsilon)$-Estimator and it returns the estimator $\mathcal{Z} = \mathcal{A} + \mathcal{B}$.

Algorithm 4 $(1+\epsilon)$-ESTIMATOR.

Data: The discrete space $[2n]^2$, an opening cost $f > 0$, and an error parameter $0 < \epsilon$.
Result: A $(1+\epsilon)$-estimator $\mathcal{Z}$ for the facility location cost of a point set $P \subseteq [2n]^2$.

Let $\mathcal{Z} = 0$, $\mathcal{A} = 0$, and $\mathcal{B} = 0$;
for $i = \log(n) + 1$ *down to* $\frac{3}{4} \cdot \log(n)$ **do**
  for *each cell* $c \in \mathcal{G}_i$ **do**
    if HEAVYTESTER(c) *returns that* c *is a heavy cell* **then**
      Let $C_c = \{c_1, c_2, c_3, c_4\}$ be the four children of c in the grid $\mathcal{G}_{i-1}$;
      Invoke HEAVYTESTER for the sub-cells C_c to find out which ones are heavy;
      for *each light sub-cell* $c' \in C_c$ *if there exists any light cell* **do**
        Let $\text{cost}^{\epsilon}_{FL}(c', F_{c'}, f)$ be the cost that the algorithm $\text{ALG}^{\epsilon}_{FL}(c')$ returns for c. Let $\mathcal{A} = \mathcal{A} + \text{cost}^{\epsilon}_{FL}(c, F_{c'}, f)$;
for $i = \frac{3}{4} \cdot \log(n)$ *down to* 1 **do**
  Let $\mathcal{W}_{i-1} = 0$;
  Let $\mathcal{X}_{i-1}$ and $\mathcal{Y}_{i-1}$ be two vectors of length $\log_{1+\epsilon}(n)$ initialized to zero;
  for $\ell = 1$ *to* $z = 2^{14}\delta^2_{FL} \log^5(n)\epsilon^{-6} \cdot \sqrt{n}$ **do**
    Let c be a cell sampled using Subroutine TELESCOPINGSAMPLING$(\mathcal{G}_i)$;
    if HEAVYTESTER(c) *returns that* c *is a heavy cell* **then**
      Let $C_c = \{c_1, c_2, c_3, c_4\}$ be the four children of c in the grid $\mathcal{G}_{i-1}$;
      Invoke HEAVYTESTER for the sub-cells C_c to find out which ones are heavy;
      if *at least one of the sub-cells* C_c *is a significant light cell* **then**
        Let j be the power of $(1+\epsilon)$ where $(1+\epsilon)^j \le n_c < (1+\epsilon)^{j+1}$;
        Let C'_c be the sub-cells in C_c that are significant light cells;
        Let $\mathcal{Y}_{i-1}[j] = \mathcal{Y}_{i-1}[j] + |C'_c|$;
        for *each significant light sub-cell* $c' \in C'_c$ **do**
          Let $\text{cost}^{\epsilon}_{FL}(c', , F_{c'}, f)$ be the cost that Algorithm $\text{ALG}^{\epsilon}_{FL}(c')$ returns for c'. Let $\mathcal{X}_{i-1}[j] = \mathcal{X}_{i-1}[j] + \text{cost}^{\epsilon}_{FL}(c', F_{c'}, f)$;
  for $j = 1$ *to* $\log_{1+\epsilon}(n)$ **do**
    Let $w^j_i = \frac{n}{z(1+\epsilon)^j} \cdot \mathcal{Y}_{i-1}[j]$. Let $\mathcal{W}_{i-1} = \mathcal{W}_{i-1} + w^j_i \cdot \mathcal{X}_{i-1}[j]$;
  Let $\mathcal{B} = \mathcal{B} + \mathcal{W}_{i-1}$;
Return $\mathcal{Z} = \mathcal{A} + \mathcal{B}$;

Recall that for the quadtree construction in Lemma 4, we randomly shift the border lines of cells of the quadtree using a random vector v. In the sublinear geometric model, we randomly shift the border lines of axis-aligned rectangles (using v) that we use in RANGECOUNT queries. Lemma 4 shows that $(1-\epsilon) \cdot OPT_{FL}(P, f) \le \sum_{c \in \Lambda} \text{cost}^{\epsilon}_{FL}(c, F_c, f) \le (1+\epsilon) \cdot OPT_{FL}(P, f)$.

We further split the set Λ into two subsets $\Lambda_{\mathcal{A}}$ and $\Lambda_{\mathcal{B}}$. In particular, let $\Lambda_{\mathcal{A}}$ be the subset of light cells in Λ that are in the grids $\mathcal{G}_{\frac{3}{4}\cdot\log(n)}, \cdots, \mathcal{G}_{\log(n)+1}$. Let $\Lambda_{\mathcal{B}} = \Lambda \backslash \Lambda_{\mathcal{A}}$ be the subset of light cells in Λ that are in $\mathcal{G}_1, \cdots, \mathcal{G}_{\frac{3}{4}\cdot\log(n)-1}$. Our goal in this section is to show that with a probability greater than $1 - 1/n^5$, the following three claims are correct:

- The number of queries that Algorithm 4 $(1+\epsilon)$-ESTIMATOR makes is $\tilde{O}(\sqrt{n})$.
- The term $\mathcal{A}$ is the cost of light cells of the set $\Lambda_{\mathcal{A}}$. That is, $\mathcal{A} = \sum_{c \in \Lambda_{\mathcal{A}}} \text{cost}^{\epsilon}_{FL}(c, F_c, f)$.

- The term $\mathcal{B}$ is an $(1+\epsilon)$-estimator for the cost of light cells of the set $\Lambda_{\mathcal{B}}$. That is, $(1-\epsilon)\cdot\sum_{c\in\Lambda_{\mathcal{B}}} \text{cost}^{\epsilon}_{FL}(c,F_c,f) \leq \mathcal{B} \leq (1+\epsilon)\cdot\sum_{c\in\Lambda_{\mathcal{B}}} \text{cost}^{\epsilon}_{FL}(c,F_c,f)$.

Once we prove these three claims, we have shown that $\mathcal{Z}$ is an $(1+\epsilon)$-estimator of the optimal facility location cost $OPT_{FL}(P,f)$ what proves Theorem 2.

3.5.1 The estimator term $\mathcal{A}$

The following two lemmas (whose proofs are given in Appendix C) show that (1) the number of RangeCount queries that we make to compute estimator $\mathcal{A}$ is $\tilde{O}(\sqrt{n})$, and (2) the term $\mathcal{A}$ approximates the facility location cost of light cells of set $\Lambda_{\mathcal{A}}$.

▶ **Lemma 9.** *Let $\Lambda_{\mathcal{A}}$ be the subset of light cells in Λ that are in the grids $\mathcal{G}_{\geq \frac{3}{4}\cdot\log(n)}$. Then,*

- *All light cells of $\Lambda_{\mathcal{A}}$ are detected by Algorithm 4 $(1+\epsilon)$-Estimator.*
- *To compute estimator $\mathcal{A}$, we make $O(\epsilon^{-8}\cdot\log^6(n)\cdot\sqrt{n})$ RangeCount queries.*

▶ **Lemma 10.** $\mathcal{A} = \sum_{c\in\Lambda_{\mathcal{A}}} cost^{\epsilon}_{FL}(c,F_c,f)$.

3.5.2 The estimator term $\mathcal{B}$

In this section, we prove that estimator $\mathcal{B}$ approximates the cost of light cells of $\Lambda_{\mathcal{B}}$ within $(1+\epsilon)$-factor. Let $c\in\Lambda$ be an arbitrary light cell in the set Λ. Suppose that c is in a grid $\mathcal{G}_i$. Then, the parent of c that we denote it by *parent*(c) is a heavy cell in grid $\mathcal{G}_{i+1}$.

▶ **Definition 11** (Significant Light Cells). *Let $\tau = \frac{\epsilon}{9\log(n)}$. We say a light cell $c\in\Lambda$ is a significant light cell if $cost^{\epsilon}_{FL}(c,F_c,f)\geq\tau f$; otherwise, we say c' is an insignificant light cell. We let $\Psi = \{c\in\Lambda : cost^{\epsilon}_{FL}(c,F_c,f)\geq\tau f\}$ be the set of light cells in Λ that are significant.*

We denote by $\Psi_i = \{c\in\Psi : c\in\mathcal{G}_i\}$ the set of significant light cells in the grid $\mathcal{G}_i$.

▶ **Definition 12** (Heavy Parents of Significant Light Cells). *We denote the set of heavy parents of significant light cells Ψ_i by $\Gamma(\Psi_i) = \{c'\in\mathcal{G}_{i+1} : \exists c\in\Psi_i$ such that c' is parent$(c)\}$.*

Note that up to 4 cells in Ψ_i can have the same heavy parent.

▶ **Definition 13** (Likelihood Classes). *We partition Ψ_i into $\epsilon^{-1}\log(n)$ likelihood classes Ψ_i^j where a cell $c\in\Psi_i^j$ if the number of points in its parent(c) is in the range $(1+\epsilon)^j \leq n_{parent(c)} < (1+\epsilon)^{j+1}$.*

We denote the set of heavy parents of significant light cells that are in the class Ψ_i^j by $\Gamma(\Psi_i^j) = \{c'\in\mathcal{G}_{i+1} : \exists c\in\Psi_i^j$ such that $c' = parent(c)\}$.

▶ **Definition 14** (Contributing Likelihood Classes). *Let $\beta = \frac{\epsilon^2}{2\delta_{FL}\cdot\log^2(n)}$. We say a class Ψ_i^j is a contributing likelihood class if $|\Psi_i^j|\geq\beta\sqrt{n}$; otherwise, it is a non-contributing class.*

Roadmap of the proof of Theorem 2. We first prove (in Lemma 22) that the cost of significant light cells of $\Psi = \{c\in\Lambda : \text{cost}^{\epsilon}_{FL}(c,F_c,f)\geq\tau f\}$ is an $(1+\epsilon)$-approximation of $OPT_{FL}(P,f)$. Therefore, when we develop a $(1+\epsilon)$-estimator for the facility location problem, we can ignore insignificant light cells of Λ.

On the other hand, all significant light cells of Ψ are partitioned into likelihood classes Ψ_i^j. Now imagine we have the lower bound $OPT_{FL}(P,f)\geq\sqrt{n}\cdot f$ for $OPT_{FL}(P,f)$. Otherwise, the estimator $\sqrt{n}$-Estimator of Lemma 6 makes $O(\epsilon^{-9}\log^7(n)\cdot\sqrt{n}))$ RangeCount queries and returns a set of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of P. Lemma 27 shows that we can safely ignore the contribution of the non-contributing likelihood classes Ψ_i^j. Let us fix a contributing likelihood classes Ψ_i^j.

- In Lemma 15 we show that with probability at least $1 - 2/n^5$, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is a $(1+\epsilon)$-approximation of the size of Ψ_i^j.
- Lemma 16 shows that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, the cell c is sampled almost uniformly at random from the set Ψ_i^j.
- Lemma 17 proves that with probability at least $1-1/n^{20}$, Algorithm 4 $(1+\epsilon)$-ESTIMATOR samples at least $\text{poly}(\epsilon^{-1} \cdot \log(n))$ significant light cells from Ψ_i^j.

In Lemma 18, we use these three tools to show that if a likelihood class Ψ_i^j is contributing, we can approximate the facility location cost $\text{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$ to within $(1+\epsilon)$-factor. Putting everything together, the estimator $\mathcal{Z}$ that is computed in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is an $(1+\epsilon)$-approximation of $OPT_{FL}(P, f)$ what proves Theorem 2.

The statements of Lemmas 22 and 27, and their proofs are given in Appendix D.

Approximating the number of cells of a contributing class. In this section, we show that with high probability, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is a $(1+\epsilon)$-approximation of the size of the contributing likelihood class Ψ_i^j.

▶ **Lemma 15.** *Let Ψ_i^j be a contributing class. Then, with probability at least $1 - 2/n^5$, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is a $(1+\epsilon)$-approximation of $|\Psi_i^j|$. That is,* $\mathbf{Pr}\left[(1-\epsilon) \cdot |\Psi_i^j| \leq w_i^j \leq (1+\epsilon) \cdot |\Psi_i^j|\right] \geq 1 - 1/n^5$.

Proof. We define z runs $\mathcal{R}_1, \cdots, \mathcal{R}_z$ where during a run $\mathcal{R}_\ell$ we sample a cell $c \in \mathcal{G}_{i+1}$. We define a random variable Y_ℓ corresponding to the number of significant light sub-cells of a heavy cell $c \in \Gamma(\Psi_i^j)$ that is sampled in the run $\mathcal{R}_\ell$. Note that a heavy cell can have $0, 1, 2, 3, 4$ significant light sub-cells. If a heavy cell c from the set $\Gamma(\Psi_i^j)$ is sampled using Subroutine TELESCOPINGSAMPLING($\mathcal{G}_{i+1}$), the random variable Y_ℓ will be the number of significant light sub-cells of c; otherwise Y_ℓ is zero. Then, we have $\mathbf{E}[Y_\ell] = \sum_{c \in \Gamma(\Psi_i^j)} \mathbf{Pr}[c] \cdot |C_c'|$, where C_c' is the set of significant light sub-cells of a heavy cell $c \in \Gamma(\Psi_i^j)$.

Recall that the likelihood class Ψ_i^j contains those cells $c \in \Psi_i$ for which $(1+\epsilon)^j \leq n_{parent(c)} < (1+\epsilon)^{j+1}$. Then, $\frac{(1+\epsilon)^j}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} |C_c| \leq \mathbf{E}[Y_\ell] \leq \frac{(1+\epsilon)^{j+1}}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} |C_c|$.

Next, we define the random variable $Y = \frac{n}{z(1+\epsilon)^j} \cdot \sum_{\ell=1}^{z} Y_\ell$ for which we have $|\Psi_i^j| = \sum_{c \in \Gamma(\Psi_i^j)} |C_c| \leq \mathbf{E}[Y] \leq (1+\epsilon) \cdot \sum_{c \in \Gamma(\Psi_i^j)} |C_c| = (1+\epsilon) \cdot |\Psi_i^j|$.

Since the likelihood class Ψ_i^j is contributing, Lemma 28 tells us that for $\alpha = \frac{\epsilon^3}{18 \delta_{FL} \log^3(n)}$ we have $\text{cost}_{FL}^{\epsilon}(\Psi_i^j, f) = \sum_{c \in \Psi_i^j} \text{cost}_{FL}^{\epsilon}(c, F_c, f) \geq \alpha\sqrt{n} \cdot f$. All cells in the likelihood class Ψ_i^j are light which means that $\text{cost}_{FL}^{\epsilon}(c, F_c, f) \leq \delta_{FL} \cdot f$. This essentially means that $|\Psi_i^j| \geq \frac{\alpha\sqrt{n} \cdot f}{\delta_{FL} \cdot f} = \frac{\alpha}{\delta_{FL}} \cdot \sqrt{n}$. We can assume that $n \geq 2^{12} \epsilon^{-4} \log^2(n) (\frac{\delta_{FL}}{\alpha})^2$, otherwise the facility location instance of the point set P has at most $\text{poly}(\epsilon^{-1} \log(n))$ points, that can be $(1+\epsilon)$-approximately solved using $\text{poly}(\epsilon^{-1} \log(n))$ RANGECOUNT queries. Therefore, $\mathbf{E}[Y] \geq \frac{\alpha}{\delta_{FL}} \cdot \sqrt{n} \geq 60\epsilon^{-2} \log(n)$. Now, we use the Hoeffding bound [22] where we set $M = 4$ to obtain we have $\mathbf{Pr}\left[|w_i^j - |\Psi_i^j|| \geq \epsilon \cdot |\Psi_i^j|\right] = \mathbf{Pr}\left[\left|\mathbf{E}[Y] - Y\right| \geq \epsilon \cdot \mathbf{E}[Y]\right] \leq 2\exp(-(\frac{\epsilon^2 \cdot \mathbf{E}[Y]}{12})) \leq 2/n^5$.

Thus, with probability at least $1 - 2/n^5$, the estimator w_i^j is within $(1+\epsilon)$-approximation of $|\Psi_i^j|$. This proves the lemma. ◀

Almost uniformly sampling from a contributing class. Let Ψ_i^j be a contributing likelihood class. Here we show that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, the cell c is sampled almost uniformly at random. We later show that with high probability, Algorithm 4 $(1+\epsilon)$-ESTIMATOR samples significant light cells from each contributing likelihood class Ψ_i^j.

▶ **Lemma 16.** *Suppose we sample a cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$. Then, the cells in Ψ_i^j are sampled almost uniformly at random. That is, the probability that we sample a cell $c \in \Psi_i^j$ is $(1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \leq \mathbf{Pr}[c] \leq (1+\epsilon) \cdot \frac{1}{|\Psi_i^j|}$.*

Proof. Recall that for every light cell $c \in \Psi_i^j$, we have $(1+\epsilon)^j \leq n_{parent(c)} < (1+\epsilon)^{j+1}$. Thus $\frac{(1+\epsilon)^j}{n} \leq \mathbf{Pr}[c] = \frac{n_{parent(c)}}{n} \leq \frac{(1+\epsilon)^{j+1}}{n}$. We conclude that the probability of sampling light cells Ψ_i^j is within $(1+\epsilon)$ of $\frac{(1+\epsilon)^j}{n}$. ◀

▶ **Lemma 17.** *Let Ψ_i^j be a contributing class. Then, with probability at least $1 - 1/n^{20}$, a set of at least $x = 2^9 \delta_{FL} \epsilon^{-3} \log^2(n)$ heavy cells are sampled from the set $\Gamma(\Psi_i^j)$.*

Proof. Recall that $z = 2^{14} \delta_{FL}^2 \log^5(n) \epsilon^{-6} \cdot \sqrt{n}$. We define z runs $\mathcal{R}_1, \cdots, \mathcal{R}_z$ where during a run $\mathcal{R}_\ell$ we sample a cell $c \in \mathcal{G}_{i+1}$. Next we study the probability of sampling a heavy cell $c \in \Gamma(\Psi_i^j)$. We define an indicator random variable Y_ℓ corresponding to the run $\mathcal{R}_\ell$ which is one if a heavy cell $c \in \Gamma(\Psi_i^j)$ is sampled using Subroutine TelescopingSampling($\mathcal{G}_{i+1}$) in Line 14 (of Algorithm 4 $(1+\epsilon)$-Estimator) and zero otherwise. Then, we have $\mathbf{E}[Y_\ell] = \mathbf{Pr}[Y_\ell = 1] = \frac{1}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} n_c$.

Based on Lemma 29, the number of points in the contributing likelihood class Ψ_i^j must be at least $n(\Psi_i^j) = \sum_{c \in \Psi_i^j} n_c \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$, for $\alpha = \frac{\epsilon^3}{18 \delta_{FL} \log^3(n)}$. Therefore, $\mathbf{E}[Y_\ell] = \frac{1}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} n_c \geq \frac{\frac{\alpha\sqrt{n}}{(1+\epsilon)}}{n} \geq \frac{\alpha}{(1+\epsilon)} \cdot \frac{1}{\sqrt{n}}$. Next, we define the random variable $Y = \sum_{\ell=1}^{z} Y_\ell$ whose expectation is $\mathbf{E}[Y] = \sum_{\ell=1}^{z} \frac{1}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} n_c \geq 2^{10} \delta_{FL} \epsilon^{-3} \log^2(n)$.

Using the Chernoff bound [13] and since $\delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2$, we obtain

$$\mathbf{Pr}\left[\left|\mathbf{E}[Y] - Y\right| \geq \epsilon \cdot \mathbf{E}[Y]\right] \leq 2\exp(-(\frac{\epsilon^2 \cdot 2^{10} \delta_{FL} \epsilon^{-3} \log^2(n)}{3})) \leq 1/n^{20} \ .$$

Thus, with probability at least $1 - 1/n^{20}$, we sample at least $(1-\epsilon) \cdot 2^{10} \delta_{FL} \epsilon^{-3} \log^2(n) \geq 2^9 \delta_{FL} \epsilon^{-3} \log^2(n)$ heavy cells from the set $\Gamma(\Psi_i^j)$. Note that up to 4 cells in Λ_i^j can have the same heavy parent and they are sampled together. This proves this lemma. ◀

An unbiased estimator for contributing classes. Next, we prove that having an estimation of the size of a contributing likelihood class Ψ_i^j and assuming that we can sample cells in Ψ_i^j almost uniformly at random, we can then $(1+\epsilon)$-approximate the facility location cost of the contributing likelihood class Ψ_i^j.

▶ **Lemma 18** (Estimator for Contributing Classes)**.** *Let Ψ_i^j be a contributing class. Suppose we can sample a cell from Ψ_i^j almost uniformly at random. That is, the probability that we sample a cell $c \in \Psi_i^j$ is $(1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \leq \mathbf{Pr}[c] \leq (1+\epsilon) \cdot \frac{1}{|\Psi_i^j|}$. Let S be a sampled set of $x \geq 2^9 \delta_{FL} \epsilon^{-3} \log^2(n)$ light cells $S = \{c_1, \cdots, c_x\} \subseteq \Psi_i^j$ that are sampled almost uniformly at random. Let y be an arbitrary number in the range $[\frac{|\Psi_i^j|}{(1-\epsilon)}, \frac{|\Psi_i^j|}{(1+\epsilon)}]$. Then, with probability at least $1 - 1/n^5$, we have $(1-\epsilon) \cdot \sum_{c \in \Psi_i^j} cost_{FL}^{\epsilon}(c, F_c, f) \leq \frac{y}{x} \cdot \sum_{\ell=1}^{x} cost_{FL}^{\epsilon}(c_\ell, F_{c_\ell}, f) \leq (1+\epsilon) \cdot \sum_{c' \in \Psi_i^j} cost_{FL}^{\epsilon}(c, F_c, f)$.*

Proof. We define x random variables $X_1, \cdots, X_x$ where X_ℓ corresponds to the cost of the light cell c_ℓ sampled from Ψ_i^j. Then, since $\mathbf{E}[X_\ell] = \sum_{c \in \Psi_i^j} \mathbf{Pr}[c] \cdot \mathrm{cost}_{FL}^{\epsilon}(c, F_c, f)$, we have $(1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \leq \mathbf{E}[X_\ell] \leq (1+\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$, where $\mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) = \sum_{c \in \Psi_i^j} \mathrm{cost}_{FL}^{\epsilon}(c, F_c, f)$. By the linearity of expectation for the random variable $X = \sum_{\ell=1}^{x} X_\ell$, we obtain $(1-\epsilon)^2 \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \leq \frac{y}{x} \cdot \mathbf{E}[X] \leq (1+\epsilon)^2 \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$.

Recall that the minimum and the maximum facility location cost of a significant light cell $c \in \Psi_i^j$ are $\tau \cdot f$ and $\delta_{FL} \cdot f$, respectively, where $\tau = \frac{\epsilon}{9\log(n)}$ and $\delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2$. Thus, $X_\ell \leq \delta_{FL} \cdot f$. Also, $\mathbf{E}[X_\ell] \geq (1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \text{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \geq (1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \tau \cdot f |\Psi_i^j| = (1-\epsilon)\tau f$.

Therefore, $\mathbf{E}[X] \geq x(1-\epsilon)\tau f \geq 2^5 \delta_{FL} \epsilon^{-2} \log(n) \cdot f$. We use the Hoeffding bound [22] where we set $M = \delta_{FL} \cdot f$ and since $f > 0$ to obtain

$$\mathbf{Pr}\left[(1-\epsilon)^3 \cdot \text{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \leq \frac{y}{x} \cdot \mathbf{E}[X] \leq (1+\epsilon)^3 \cdot \text{cost}_{FL}^{\epsilon}(\Psi_i^j, f)\right]$$
$$= \mathbf{Pr}\left[|X - \mathbf{E}[X]| \geq \epsilon \cdot \mathbf{E}[X]\right] \leq 2 \cdot \exp(-\frac{\mathbf{E}[X] \cdot \epsilon^2}{3 \cdot M}) \leq 2 \cdot \exp(-\frac{2^5 \delta_{FL} \epsilon^{-2} \log(n) \cdot f \epsilon^2}{3\delta_{FL} \cdot f}) \leq \frac{2}{n^5} \ .$$

We replace ϵ by $\epsilon/3$ to finish the proof of this lemma. ◀

Finishing Proof of Theorem 2. Now we are ready to finish the proof of Theorem 2. Lemma 18 shows that if a likelihood class Ψ_i^j is contributing, we can approximate the facility location cost $\text{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$ to within $(1+\epsilon)$-factor if (1) we can sample x significant light cells of Ψ_i^j almost uniformly at random, and (2) we can $(1+\epsilon)$-approximate the size $|\Psi_i^j|$.

Lemma 16 shows that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, the cell c is sampled almost uniformly at random. Lemma 17 proves that with probability at least $1 - 1/n^{20}$, Algorithm 4 $(1+\epsilon)$-ESTIMATOR samples at least x significant light cells from each contributing likelihood class Ψ_i^j. Finally, Lemma 15, with probability at least $1 - 2/n^5$, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is a $(1+\epsilon)$-approximation of the size of the contributing likelihood class Ψ_i^j.

Recall that in Lemma 22 we proved that the cost of significant light cells of $\Psi = \{c \in \Lambda : \text{cost}_{FL}^{\epsilon}(c, F_c, f) \geq \tau f\}$ is an $(1+\epsilon)$-approximation of $OPT_{FL}(P, f)$. Therefore, we can ignore insignificant light cells of Λ. All significant light cells of Ψ are partitioned into likelihood classes Ψ_i^j. Using Lemma 27, we can safely ignore the contribution of the non-contributing likelihood classes Ψ_i^j. Putting everything together, the estimator $\mathcal{Z}$ that is computed in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is an $(1+\epsilon)$-approximation of $OPT_{FL}(P, f)$.

As for the query complexity of Algorithm 4 $(1+\epsilon)$-ESTIMATOR. In Lemma 9, to compute the estimator $\mathcal{A}$ we use $O(\epsilon^{-8} \cdot \log^6(n) \cdot \sqrt{n})$ RANGECOUNT queries. For the estimator $\mathcal{B}$, we sample $z = 2^{14} \delta_{FL}^2 \log^5(n) \epsilon^{-6} \cdot \sqrt{n}$ cells using Subroutine TELESCOPINGSAMPLING($\mathcal{G}_i$) for each grid $\mathcal{G}_{i \leq \frac{3}{4}\log(n)}$. Therefore, we invoke the tester HEAVYTESTER for less than $5z\log(n)$ cells (z times for the sampled cells and $4z$ for their children) where each such a call makes $O(\epsilon^{-8} \cdot \log^6(n))$ RANGECOUNT queries as is shown in Lemma 5. Therefore, in total, the query complexity of the $(1+\epsilon)$-estimator $\mathcal{Z}$ is $\tilde{O}(\sqrt{n})$. This finishes the proof of Theorem 2.

4 Hard instance for the sublinear geometric model

In this section, we prove Lemma 3. See Figure 2 for the illustration of the hard instance that we explain next.

Suppose the opening cost is $f = n^{3/4}$. In the grid $\mathcal{G}_{\frac{3}{4}\cdot\log(n)}$, we have $n^{1/2}$ cells each one having side length $n^{3/4}$. We choose a set $A \subset \mathcal{G}_{\frac{3}{4}\cdot\log(n)}$ of $n^{1/4}$ cells arbitrarily and assign $n^{3/4}$ points to each such a cell. Observe that the sparsity of the grid $\mathcal{G}_{\frac{3}{4}\cdot\log(n)}$ is $n^{1/4}$. That is in average from every $n^{1/4}$ cells in this grid, only one of them has $n^{3/4}$ points and the rest are empty. Note that every cell in A is heavy since it has at least $\frac{f}{n^{3/4}} = 1$ points.

At grids $\mathcal{G}_i$ and $\mathcal{G}_j$ for $i = \frac{\log(n)}{2}$ and $j = \frac{\log(n)}{4}$, a cell has side length $n^{1/2}$ and $n^{1/4}$. We sample a subset $B \subset A$ of $\log(n)$ cells uniformly at random. A cell $c \in B$ has a set $\mathcal{D}_c$ of $\sqrt{n}$ descendants in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$, each one having $n^{\frac{1}{4}}$ points equally distant (at distance

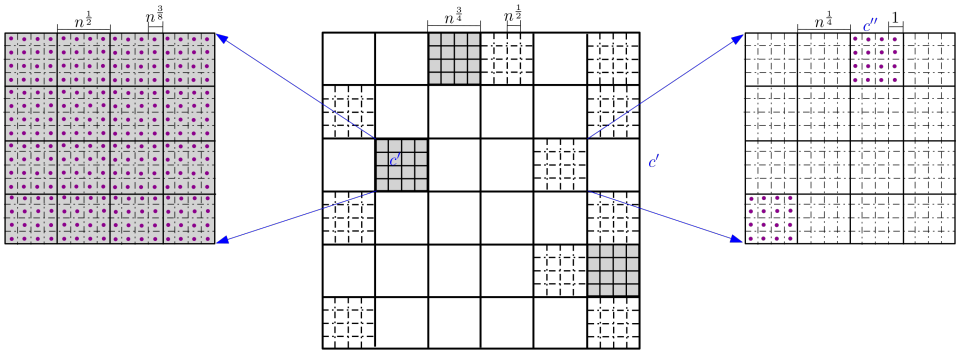

Figure 2 In this figure, cells of B are colored gray and cells in the set $A\backslash B$ are shown with a net of dash-dotted lines. The rest of cells that are shown as white cells are empty cells. The cell $c \in B$ has $\sqrt{n}$ descendants in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$ that are all heavy cells. The cell $c' \in A\backslash B$ has n descendants in the grid $\mathcal{G}_{\frac{1}{4}\cdot\log(n)}$ out of which $n^{1/4}$ of them are heavy and the rest are empty cells.

$n^{\frac{3}{8}}$) from its top, bottom, left and right points. Every cell in $\mathcal{D}_c$ has $n^{\frac{1}{4}} = \frac{f}{n^{1/2}}$ points, so it is a heavy cell in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$. In the optimal solution for the facility location problem, we must open at least one facility within distance $n^{1/2}$ of each heavy cell in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$. Thus, the optimal cost of the facility location problem for the cell set B is at least $\mathcal{L} = \log(n) \cdot \frac{n^{1/2}}{9} \cdot f$ where in the denominator we have 9 since the points inside at most 9 cells (i.e., a grid of 3×3) in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$ can be assigned to one open facility that we open in one of them, say the center one.

Each cell $c' \in A\backslash B$ has n descendants in the grid $\mathcal{G}_j$, but only $n^{1/4}$ of them that are chosen uniformly at random are heavy, i.e., has $\frac{f}{n^{1/4}} = n^{1/2}$ points and the rest are empty cells. Suppose that for each heavy descendant (in the grid $\mathcal{G}_j$) of each cell $c' \in A\backslash B$, we open at most one facility. The total cost of cells of the set $A\backslash B$ would be at most $\mathcal{U} \leq |A\backslash B| \cdot n^{1/4} \cdot \frac{f}{n^{1/4}} \cdot n^{1/4}\sqrt{2} \leq \sqrt{2}n^{1/2} \cdot f$

Now, observe that with respect to the grids $\mathcal{G}_{\geq\frac{3}{4}\cdot\log(n)}$, all the cells in the set A look exactly the same and they all have $n^{3/4}$ points each. Since $|B| = \log(n)$ and $|A| = n^{1/4}$, in expectation we need to sample $\frac{|A|}{|B|} = \frac{n^{1/4}}{\log(n)}$ cells so that we can have at least one cell from B in the sampled set; otherwise, we cannot estimate the number of cells and the cost of each cell in the set B and thus, we cannot approximate the cost of the facility location problem within a factor better than $\Omega(\frac{\mathcal{L}}{\mathcal{U}}) = \Omega(\log(n))$.

References

1 Pankaj K. Agarwal. Range searching. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 809–837. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch36.

2 Pankaj K. Agarwal, Edward F. Grove, T. M. Murali, and Jeffrey Scott Vitter. Binary search partitions for fat rectangles. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 482–491. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548507.

3 N. Alon and J. Spencer. *The probabilistic method*. J. Wiley & Sons, New York, 2nd edition, 2000.

4 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583. ACM, 2014. doi:10.1145/2591796.2591805.

5 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 639–648. ACM, 2009. doi:10.1145/1536414.1536501.

6 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010. doi:10.1137/090762968.

7 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 106–113. ACM, 1998. doi:10.1145/276698.276718.

8 Mihai Badoiu, Artur Czumaj, Piotr Indyk, and Christian Sohler. Facility location in sublinear time. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 866–877. Springer, 2005. doi:10.1007/11523468_70.

9 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017. doi:10.1145/3125644.

10 Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021. doi:10.1109/FOCS52979.2021.00089.

11 Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. doi:10.1145/361002.361007.

12 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. doi:10.1137/S0097539702403244.

13 Herman Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952. doi:10.1214/aoms/1177729330.

14 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM J. Comput.*, 35(1):91–109, 2005. doi:10.1137/S0097539703435297.

15 Artur Czumaj, Christiane Lammersen, Morteza Monemizadeh, and Christian Sohler. $(1+\epsilon)$-approximation for facility location in data streams. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1710–1728. SIAM, 2013. doi:10.1137/1.9781611973105.123.

16 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183. ACM, 2004. doi:10.1145/1007352.1007386.

17 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

18 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In Joseph S. B. Mitchell and Günter Rote, editors, *Proceedings of the 21st ACM Symposium on Computational Geometry, Pisa, Italy, June 6-8, 2005*, pages 142–149. ACM, 2005. doi:10.1145/1064092.1064116.

19 Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 209–217. ACM, 2005. doi:10.1145/1060590.1060622.

20 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In Takao Asano, Shin-Ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, volume 7074 of *Lecture Notes in Computer Science*, pages 374–383. Springer, 2011. doi:10.1007/978-3-642-25591-5_39.

21 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300. ACM, 2004. doi:10.1145/1007352.1007400.

22 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. URL: http://www.jstor.org/stable/2282952.

23 Piotr Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 154–159. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814587.

24 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 373–380. ACM, 2004. doi:10.1145/1007352.1007413.

25 Piotr Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 39–42. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283388.

26 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.

27 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948. SIAM, 2010. doi:10.1137/1.9781611973075.76.

28 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. URL: https://mitpress.mit.edu/books/introduction-computational-learning-theory.

29 Michael J. Kearns and Umesh V. Vazirani. Computational learning theory. *SIGACT News*, 26(1):43–45, 1995. doi:10.1145/203610.606411.

30 Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Comput.*, 37(3):757–782, 2007. doi:10.1137/S0097539702404055.

31 Philip M. Long and Lei Tan. Pac learning axis-aligned rectangles with respect toproduct distributions from multiple-instance examples. *Mach. Learn.*, 30(1):7–21, January 1998. doi:10.1023/A:1007450326753.

32 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. doi:10.1017/cbo9780511814075.

33 Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.81.

34 Sharath Raghvendra and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. *J. ACM*, 67(3):18:1–18:19, 2020. doi:10.1145/3393694.

35 Menachem Sadigurschi and Uri Stemmer. On the sample complexity of privately learning axis-aligned rectangles. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28286–28297, 2021. URL: `https://proceedings.neurips.cc/paper/2021/hash/ee0e95249268b86ff2053bef214bfeda-Abstract.html`.

36 Srikanta Tirthapura and David P. Woodruff. Rectangle-efficient aggregation in spatial data streams. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 283–294. ACM, 2012. `doi:10.1145/2213556.2213595`.

37 Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000.

38 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 225–234. ACM, 2009. `doi:10.1145/1536414.1536447`.

A Missing proof of Subsection 3.1

Proof of Lemma 5. Testing whether a cell c is a heavy or a light cell needs to find out if c has a facility location cost of at least $\delta_{FL}f$ or lower than this quantity. To this end, we first develop a deterministic sublinear algorithm for the k-median problem that using $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries returns a $(1+\epsilon)$-approximate solution. We then show how we can use this sublinear algorithm to develop the tester algorithm HeavyTester.

What is interesting is the difference between the query complexity of the k-median and the facility location problems in the sublinear geometric model. For the k-median problem, we can report a $(1+\epsilon)$-approximate solution (not the cost) in the sublinear model using $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries. However, for the facility location, Lemma 3 shows that we do not hope to $o(\log n)$-approximate the cost of the facility location problem in the sublinear geometric model using $\Omega(\frac{\sqrt[4]{n}}{\log n})$ RangeCount queries.

▶ **Definition 19** (k-median problem). *In the* k-median problem*, we are given a point set* $P \subset [2n]^2$ *of size* n *in a discrete space and a number* $k \in \mathbb{N}$ *of centers. The goal is to return a set* $C^* \subset [2n]^2$ *of* k *centers that minimizes the cost function* $cost_{k\text{-}med}(P, C^*) = \sum_{p\in P} dist(p, C^*)$*, where* $dist(p, C^*) = \min_{c\in C^*} dist(p, c)$ *is the Euclidean distance of* p *to its nearest center in* C^**. We denote the optimal* k*-median cost of the point set* P *by* $OPT_{k\text{-}med}(P, k)$*.*

▶ **Lemma 20** (Sublinear algorithm for k-median). *Let* P *be a point set of size* n *in* $[2n]^2$*. Let* $k > 0$ *be the number of centers and* $0 < \epsilon \leq 1$ *be the error parameter. Then, there exists a deterministic sublinear algorithm that returns a* $(1+\epsilon)$*-approximate solution of the* k*-median problem for* P *in the sublinear geometric model using* $O(k\epsilon^{-4}\log^2(n))$ RangeCount *queries.*

Assume for a moment that this lemma is correct. We next explain the tester algorithm.

Algorithm HeavyTester(c). For every choice $k \in \{1, 2, \cdots, \delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2\}$, we run the sublinear algorithm of Lemma 20 on the input set $P \cap c$ that reports a set C_k of k centers. We then compute the k-median cost of $P \cap c$ using the center set C_k and add the opening cost kf to the k-median cost to compute the facility location cost. Among all δ_{FL} runs, we find the one that has the lowest facility location cost. If the lowest facility location cost is less than δ_{FL}, we report that the cell c is a light cell, otherwise we report it as a heavy cell. This finishes the description of the algorithm HeavyTester(c).

Query complexity of HeavyTester(c). We invoke δ_{FL} parallel runs each choice $k \in [\delta_{FL}]$. Each such a run needs $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries using Lemma 20. Thus, using $O(\delta_{FL}^2 \cdot \epsilon^{-4}\log^2(n)) = O(\epsilon^{-8} \cdot \log^6(n))$ RangeCount queries, we can detect if the cell c is a heavy cell or a light one. This finishes the proof of the first part of Lemma 5.

Now we prove the second part of this lemma. Imagine the case when c is a cell in the set Λ. Recall that $c \in \Lambda$ if c is a light cell and its *parent*(c) is a heavy cell. We run the tester HeavyTester for c and *parent*(c). This tester is a deterministic algorithm and so, it correctly reports c is a light cell and *parent*(c) is a heavy cell. Lemma 4 ensures that for the correct guess of the number of facilities in c, the set F_c of facilities that the tester HeavyTester(c) returns will satisfy the following what finishes the proof of Lemma 5. $\text{cost}^{\epsilon}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \text{dist}(p, F_c) + |F_c| \cdot f \leq (1+\epsilon) \cdot OPT_{FL}(P \cap c, f)$.

Proof of Lemma 20. We prove this lemma using a modification that we do to the quadtree construction that Frahling and Sohler [19] developed for the k-median problem. The quadtree construction in [19] is based on the notions of dense and sparse cells.

Dense and sparse cells: Let P be a point set of size n in $[2n]^2$. Let $0 < \epsilon \leq 1$ be an error parameter. Let $\delta_{k\text{-}med} = 2^{20} \cdot \frac{k\log(n)}{\epsilon^3}$. We say a cell $c \in \mathcal{G}_i$ of side length 2^i is *dense* if it contains $n_c \geq \delta_{k\text{-}med} \cdot \frac{OPT_{k\text{-}med}(P,k)}{2^i}$ points; otherwise it is a *sparse* cell.

Observe that for the definition of dense and sparse cells, we assume we know the optimal k-median cost $OPT_{k\text{-}med}(P,k)$. In general, we can $(1+\epsilon)$-approximate $OPT_{k\text{-}med}(P,k)$ using $\log_{(1+\epsilon)}(4n^2) = O(\epsilon^{-1}\log(n))$ guesses. To see this, observe that in the discrete space $[2n]^2$, the maximum pairwise distance between two points is at most $4n$ and the minimum distance is 1. The set P has n points, so the ratio between the maximum and the minimum k-median costs is at most $n \times 4n = 4n^2$. We then consider $t = \log_{(1+\epsilon)}(4n^2) = O(\epsilon^{-1}\log(n))$ parallel guesses $r_0, \cdots, r_t$ for the optimal k-median cost $OPT_{k\text{-}med}(P,k)$ where $r_j = (1+\epsilon)^j$. We must have an index $j \in [t]$ for which $(1+\epsilon)^j \leq OPT_{k\text{-}med}(P,k) < (1+\epsilon)^{j+1}$.

Sublinear algorithm for k-median based on quadtree construction of [19]. We run the following algorithm for all $O(\epsilon^{-1}\log(n))$ guesses of $OPT_{k\text{-}med}(P,k)$ in parallel. Let us consider the j^{th} guess $r_j = (1+\epsilon)^j$ for $OPT_{k\text{-}med}(P,k)$. We create a run j for the j^{th} guess for which we do the following.

Let $\mathcal{K}_j$ and $\mathcal{R}_j$ be two empty sets. Given the single square $c = [2n]^2$ in the grid $\mathcal{G}_{\log(n)+1}$, we build a tree similar to the quadtree as follows. In particular, suppose that we have a cell $c \in \mathcal{G}_i$. If c is sparse, we add c to $\mathcal{K}_j$ and stop; otherwise, (i.e., if c is dense), we split it into 4 equal sub-cells c_1, c_2, c_3, c_4 (they are in the grid G_{i-1}) of the same side length and recurse for those sub-cells that are dense. We add all **non-empty sparse cells** that are constructed in this way to $\mathcal{K}_j$. If during the run j, the size of $\mathcal{K}_j$ is more than $O(k\epsilon^{-3}\log(n))$, we immediately stop that run because that run corresponds to the guess $(1+\epsilon)^j$ that is much smaller than the optimal k-median cost $OPT_{k\text{-}med}(P,k)$. At the end of this recursive procedure, for each cell $c \in \mathcal{K}_j$, we consider the center of c as the representative point r_c of c and assign the weight $w(r_c) = n_c = |P \cap c|$ to r_c. Observe that for every cell $c \in \mathcal{K}_j$, we need one RangeCount query to find out $w(r_c) = n_c = |P \cap c|$. Let $\mathcal{R}_j$ be the set of weighted representative points that we compute in this way for the run j.

Next, we need to compute the k-median cost of the representative set $\mathcal{R}_j$. To this end, we use the deterministic $(1+\epsilon)$-approximation algorithm that Har-Peled and Mazumdar [21] develop for the k-median problem. The running time of this algorithm is

$O(n + 2^{O((1+1/\epsilon)}k^{O(1)}\log^{O(1)}(n))$. We denote this algorithm by $\text{ALG}^{\epsilon}_{k\text{-}med}$. For the weighted representative set $\mathcal{R}_j$, we then invoke the $\text{ALG}^{\epsilon}_{k\text{-}med}(\mathcal{R}_j)$ that returns a set C_j of k centers and its k-median cost $\Gamma_j = \text{cost}_{k\text{-}med}(\mathcal{R}_j, C_j) = \sum_{r_c \in \mathcal{R}_j} w(r_c) \cdot \text{dist}(p, C_j)$, where $\text{dist}(p, C_j) = \min_{c \in C_j} \text{dist}(p, c)$ is the distance of p to its nearest center in C_j.

Among all parallel guesses $r_0, \cdots, r_t$ where $[t = O(\epsilon^{-1}\log(n))]$, we choose the smallest guess $r_{j^*} = (1+\epsilon)^{j^*}$ for which we have $(1+\epsilon)^{j^*} \leq \Gamma_{j^*} < (1+\epsilon)^{j^*+1}$. Let $\mathcal{K}_{j^*}$ and $\mathcal{R}_{j^*}$ be the set of non-empty sparse cells and the set of representative points that correspond to the guess $(1+\epsilon)^{j^*}$, respectively. We then have the following guarantee.

▶ **Lemma 21** ([19]). *Let $P \subseteq [2n]^2$ be a point set of size n and $k \in \mathbb{N}$ be a natural number. Let $0 < \epsilon \leq 1$ be the error parameter. Then:*

- *The number of non-empty sparse cells in $\mathcal{K}_{j^*}$ (or, the number of representative points in $\mathcal{R}_{j^*}$) that are found using the above construction is $|\mathcal{K}_{j^*}| = |\mathcal{R}_{j^*}| = O(k\epsilon^{-3}\log(n))$.*
- *The output of Algorithm [21] $\text{ALG}^{\epsilon}_{k\text{-}med}$ on input set $\mathcal{R}_{j^*}$ is a set C_{j^*} of k centers such that $(1-\epsilon)OPT_{k\text{-}med}(P,k) \leq \Gamma_{j^*} = cost_{k\text{-}med}(\mathcal{R}_{j^*}, C_{j^*}) \leq (1+\epsilon)OPT_{k\text{-}med}(P,k)$.*

Query Complexity of sublinear algorithm for k-median. Let us consider a run j. Recall that if during the run j, the size of $\mathcal{K}_j$ is more than $O(k\epsilon^{-3}\log(n))$, we immediately stop this run. Observe that for every cell $c \in \mathcal{K}_j$, we need one RangeCount query to find out its weight $w(r_c) = n_c = |P \cap c|$. Thus, the number of RangeCount queries that we make to build the set $\mathcal{K}_j$ is $O(k\epsilon^{-3}\log(n))$. In addition, observe that the algorithm $\text{ALG}^{\epsilon}_{k\text{-}med}(\mathcal{R}_j)$ does not make any RangeCount query. We have $t = O(\epsilon^{-1}\log(n))$ guesses for the optimal k-median cost $OPT_{k\text{-}med}(P,k)$. Thus, the number of RangeCount queries that we make to develop our sublinear algorithm for the k-median problem is $O(k\epsilon^{-4}\log^2(n))$. This finishes the proof of Lemma 20. ◀

◀

B Missing proofs of Subsection 3.3

Proof of Lemma 7. Let us consider the ancestors $c_{i+1}, \cdots, c_j, \cdots, c_{\log(n)}$ where cell c_j for $i+1 \leq j \leq \log(n)$ is the ancestor of cell c in grid $\mathcal{G}_j$. Observe that the probability that we sample the cell c is $\mathbf{Pr}[c] = \mathbf{Pr}\left[c_{\log(n)-1}|c_{\log(n)}\right] \cdot \mathbf{Pr}\left[c_{\log(n)-2}|c_{\log(n)-1}\right] \cdots \mathbf{Pr}\left[c_{i+1}|c_{i+2}\right] \cdot \mathbf{Pr}\left[c|c_{i+1}\right] = \frac{n_{c_{\log(n)-1}}}{n} \cdot \frac{n_{c_{\log(n)-2}}}{n_{c_{\log(n)-1}}} \cdots \frac{n_{c_{i+1}}}{n_{c_{i+2}}} \cdot \frac{n_c}{n_{c_{i+1}}} = \frac{n_c}{n}$. ◀

C Missing proofs of Subsection 3.5.1

Proof of Lemma 9. Recall that the estimator $\mathcal{A}$ is for the contribution of those light cells in Λ that are in the grids $\mathcal{G}_{\frac{3}{4}\cdot\log(n) \leq i \leq \log(n)+1}$. Let S be the set of all cells in $\mathcal{G}_{\frac{3}{4}\cdot\log(n) \leq i \leq \log(n)+1}$. In total, all these grids have at most $|S| \leq \frac{2n^2}{n^{6/4}} + \frac{1}{4} \cdot \frac{2n^2}{n^{6/4}} + \cdots + \frac{1}{4^i} \cdot \frac{2n^2}{n^{6/4}} + \cdots + 1 \leq 4\sqrt{n}$ cells. For every heavy cell $c \in S$, its sub-cells c_1, c_2, c_3, c_4 are created and tested if they are heavy or light cell. Recall that we run the HeavyTester algorithm for those sub-cells to determine which ones are heavy or light. Using Lemma 5, to detect if a cell c is heavy or light, the tester algorithm HeavyTester(c) uses $O(\epsilon^{-8} \cdot \log^6(n))$ RangeCount queries.

Let us consider an empty set T in the beginning. For each heavy cell $c \in S$, we add its light sub-cells to the set T. The first claim of this lemma is proven by observing that $|T| \leq 4|S| = 16\sqrt{n}$. As for the second claim, we invoke the HeavyTester algorithm for $|S| + |T| \leq 20\sqrt{n}$ cells each one uses $O(\epsilon^{-8} \cdot \log^6(n))$ RangeCount queries. ◀

Proof of Lemma 10. Based on Lemma 5, the tester algorithm HeavyTester is a deterministic algorithm that correctly detects if an arbitrary cell c is heavy or light. Thus, Algorithm 4 $(1+\epsilon)$-Estimator detects all light cells $\Lambda_{\mathcal{A}}$. For every light cell $c \in \Lambda_{\mathcal{A}}$, Lemma 5 shows that the Tester algorithm $(1+\epsilon)$-approximates the optimal cost $OPT^{\epsilon}_{FL}(P \cap c, f)$. Thus, the term $\mathcal{A}$ is a $(1+\epsilon)$-estimator for the cost of light cells of the set $\Lambda_{\mathcal{A}}$. ◀

D Missing proofs of Subsection 3.5.2

The following lemma shows that we can safely remove insignificant light cells from the set Λ and only consider significant light cells Ψ.

▶ **Lemma 22.** *Let $\tau = \frac{\epsilon}{9\log(n)}$. Let $\Psi = \{c \in \Lambda : cost^{\epsilon}_{FL}(c, F_c, f) \geq \tau f\}$ be the set of light cells in Λ that are significant. Then,*

$$OPT_{FL}(P, f) \leq cost^{\epsilon}_{FL}(\Psi, f) = \sum_{c \in \Psi} cost^{\epsilon}_{FL}(c, F_c, f) \leq (1+\epsilon) \cdot OPT_{FL}(P, f) \ .$$

Proof. In order to prove Lemma 22, we first define charging heavy cells. We prove that at most $3\log(n)$ insignificant light cells are created during the construction of any charging heavy cells. We can assign all these insignificant light cells to the charging heavy cell whose construction creates these cells. We show that the cost of a charging heavy c is at least $\Omega(\delta_{FL} \cdot f)$. On the other hand, overall the total facility location cost of the insignificant light cells that are assigned to c is at most $\epsilon \cdot f$ that can be charged to $\text{cost}^{\epsilon}_{FL}(c, F_c, f)$.

▶ **Definition 23** (Charging heavy cell)**.** *Let c be a heavy cell having children c_1, c_2, c_3, c_4. We say c is a charging heavy cell if all children c_1, c_2, c_3, c_4 are light.*

▶ **Lemma 24.** *Let c be a heavy cell. Then, either c is a charging heavy cell or at least one of its descendants is a charging heavy cell.*

Proof. For the sake of contradiction assume that c is not a charging heavy cell. As otherwise, we have nothing to prove. Let c' be a copy of c. Let c_1, c_2, c_3, c_4 be the four children of c'. Let $C^H_{c'}$ be the set of heavy children of c'. By our assumption, $|C^H_c| > 0$. Let us pick a heavy child of c' from C^H_c, say c_1 is heavy. We let $c' = c_1$. Recursively, either all children of c' are light or at least one of the children of c' is heavy for which we recurse. This recursion repeats for $O(\log n)$ times. At the end we either find a charging heavy cell or arrive at a cell $c \in \mathcal{G}_0$ that has side length one and can store only one point p_c. If that cell c is still heavy, we can open a facility at p_c of cost f which contradicts with the fact that the cost of c must be at least $\delta_{FL} \cdot f$. Thus, the recursion will end by outputting a charging heavy cell. ◀

Let us fix a charging heavy cell c at a grid $\mathcal{G}_i$. Let us consider the quadtree construction that we explained for the facility location problem in Section 2.1. Starting from the root of the quadtree that corresponds to the square $[2n]^2$ going down to the cell c, we see the ancestors of c. At any grid $\mathcal{G}_{j>i}$, the ancestor(c, j) of c creates 4 children. At most 3 of them are insignificant light cells and one is the ancestor$(c, j-1)$ of c at grid $\mathcal{G}_{j-1}$ which is a heavy cell. Suppose we assign all the insignificant light cells that ancestor(c, j) creates to c. If there are more than one heavy cell among the children of ancestor(c, j), we can arbitrarily assign the insignificant light children of ancestor(c, j) to either of them. Let $\mathcal{O}_c$ be the set of all insignificant light cells that are assigned to c. We have the following bound for $|\mathcal{O}_c|$.

▶ **Corollary 25.** *Let c be a charging heavy cell. Let $\mathcal{O}_c$ be the set of all insignificant light cells that are assigned to c. Then, $|\mathcal{O}_c| \leq 3\log(n)$.*

▶ **Corollary 26.** *Let c be a charging heavy cell. Let $\mathcal{O}_c$ be the set of all insignificant light cells that are assigned to c. Then, the total facility location cost of insignificant light cells $\mathcal{O}_c$ is $cost^{\epsilon}_{FL}(\mathcal{O}_c, f) = \sum_{c' \in \mathcal{O}_c} cost^{\epsilon}_{FL}(c', F_{c'}, f) \leq \frac{\epsilon}{3} \cdot f$.*

Now we finish the proof of Lemma 22. Recall that the charging heavy cell c has cost $\text{cost}^{\epsilon}_{FL}(c, F_c, f) \geq \delta_{FL} \cdot f$. On the other hand, the four children $C_c = \{c_1, c_2, c_3, c_4\}$ of c are light that are in the set Λ as we have seen in Lemma 4. This lemma shows that for each cell $c' \in C_c$ we must have $\text{cost}^{\epsilon}_{FL}(c', F_c, f) \leq (1+\epsilon) \cdot OPT_{FL}(P \cap c', f)$. However, $\sum_{c' \in C_c} \text{cost}^{\epsilon}_{FL}(c', F_c, f) \geq \frac{\delta_{FL} \cdot f}{4}$. Otherwise, we can upper bound $\text{cost}^{\epsilon}_{FL}(c, F_c, f)$ by $\sum_{c' \in C_c} \text{cost}^{\epsilon}_{FL}(c', F_c, f) \leq \frac{\delta_{FL} \cdot f}{4}$ which essentially means that c is light cell contradicting our assumption that c is a charging heavy cell. Thus, $\frac{\delta_{FL} \cdot f}{4(1+\epsilon)} \leq \frac{\sum_{c' \in C_c} \text{cost}^{\epsilon}_{FL}(c', F_c, f)}{(1+\epsilon)} \leq \min(\sum_{c' \in C_c} OPT_{FL}(P \cap c', f), OPT_{FL}(P \cap c', f))$. Using Corollary 26, for the set $\mathcal{O}_c$ of insignificant light cells that are assigned to c we have $\text{cost}^{\epsilon}_{FL}(\mathcal{O}_c, f) \leq \frac{\epsilon}{3} \cdot f$ which is less than ϵ-fraction of the optimal facility location cost $OPT_{FL}(c, f)$ of c. Therefore, the overall facility location cost of insignificant light cells in Λ is at most $\epsilon \cdot OPT_{FL}(P, f)$. Thus, for set $\Psi = \{c \in \Lambda : \text{cost}^{\epsilon}_{FL}(c, F_c, f) \geq \tau f\}$ of significant light cells we have $OPT_{FL}(P, f) \leq \text{cost}^{\epsilon}_{FL}(\Psi, f) = \sum_{c \in \Psi} \text{cost}^{\epsilon}_{FL}(c, F_c, f) \leq (1+\epsilon) \cdot OPT_{FL}(P, f)$. ◀

Next lemma shows that the cost of non-contributing classes Ψ^j_i is only $\frac{\epsilon}{2}$-fraction of the optimal facility location cost of point set P. Thus, when we develop a sublinear algorithm that $(1+\epsilon)$-approximates $OPT_{FL}(P, f)$, we can ignore the cost of non-contributing classes.

▶ **Lemma 27.** *Suppose we know that $OPT_{FL}(P, f) \geq \sqrt{n} \cdot f$. Let Ψ^j_i be a likelihood class that is non-contributing, i.e., $|\Psi^j_i| < \beta\sqrt{n}$, where $\beta = \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)}$. We can safely ignore the contribution of the class Ψ^j_i toward the optimal facility location cost of the point set P.*

Proof. Recall that the class Ψ^j_i consists of only light cells and that a cell c is light if $\text{cost}^{\epsilon}_{FL}(c, F_c, f) < \delta_{FL} \cdot f$. Recall that we test whether a cell is light or heavy using the tester algorithm HeavyTester of Lemma 5. Since $|\Psi^j_i| < \beta\sqrt{n}$, for $\beta = \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)}$, the overall cost of the class Ψ^j_i is upper-bounded by $\text{cost}^{\epsilon}_{FL}(\Psi^j_i, f) = \sum_{c \in \Psi^j_i} \text{cost}^{\epsilon}_{FL}(c, F_c, f) \leq \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)} \cdot \sqrt{n} \cdot \delta_{FL} \cdot f \leq \frac{\epsilon^2}{2\log^2 n} \cdot OPT_{FL}(P, f)$.

Note that based on Definition 13, we can have at most $\epsilon^{-1} \cdot \log(n)$ classes for every grid $\mathcal{G}_i$ and we have at $\log(n) + 1$ grids. Thus, the total cost of non-contributing classes is at most $\sum_{i=1}^{\log(n)+1} \sum_{j=1}^{\epsilon^{-1} \cdot \log(n)} \text{cost}^{\epsilon}_{FL}(\Psi^j_i, f) \leq \frac{\epsilon}{2} \cdot OPT_{FL}(P, f)$. Thus we can safely ignore the contribution of these grids toward the optimal facility location cost of the point set P. ◀

We first find a lower bound for the cost of contributing classes. Next, we obtain a lower bound for the minimum number of points in a contributing likelihood class.

▶ **Lemma 28.** *Let $\alpha = \frac{\epsilon^3}{18\delta_{FL} \log^3(n)}$ and $\beta = \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)}$. Let Ψ^j_i be a likelihood class that is contributing, i.e., $|\Psi^j_i| \geq \beta\sqrt{n}$. Then, $cost^{\epsilon}_{FL}(\Psi^j_i, f) = \sum_{c \in \Psi^j_i} cost^{\epsilon}_{FL}(c, F_c, f) \geq \alpha\sqrt{n} \cdot f$.*

Proof. A class Ψ^j_i is contributing if $|\Psi^j_i| \geq \beta\sqrt{n}$. Moreover, every class Ψ^j_i consists of light cells that are significant. A light cell $c \in \Lambda$ is significant if $\text{cost}^{\epsilon}_{FL}(c, F_c, f) \geq \tau f$, where $\tau = \frac{\epsilon}{9\log(n)}$. Thus, we have the following lower bound for the facility location cost of the class Ψ^j_i: $\text{cost}^{\epsilon}_{FL}(\Psi^j_i, f) = \sum_{c' \in \Psi^j_i} \text{cost}^{\epsilon}_{FL}(c', f) \geq |\Psi^j_i| \cdot \tau \cdot f \geq \frac{\epsilon^3}{18\delta_{FL} \log^3(n)} \cdot \sqrt{n} \cdot f$. ◀

Next, using the lower bound that we obtained for the cost of contributing classes, we find a lower-bound on the number of points in every contributing class. This will help us to prove that can sample cells of contributing classes almost uniformly at random.

▶ **Proposition 29.** *For a contributing likelihood class Ψ_i^j we have $n(\Psi_i^j) = \sum_{c\in\Psi_i^j} n_c \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$.*

Proof. Let us consider a significant light cell $c \in \Psi_i^j$. Since Algorithm [15] $\text{ALG}_{FL}^{\epsilon}$ is a $(1+\epsilon)$-approximation algorithm for the facility location problem, we have $\text{cost}_{FL}^{\epsilon}(c', f) \leq (1+\epsilon)\cdot OPT_{FL}(P\cap c, f)$. This essentially means that $OPT_{FL}(P\cap c, f) \geq \frac{\text{cost}_{FL}^{\epsilon}(c',f)}{(1+\epsilon)} \geq \frac{\tau}{(1+\epsilon)}\cdot f$.

Now we take the sum over the cells of Ψ_i^j, apply Lemma 28, and let $\alpha = \frac{\epsilon^3}{18\delta_{FL}\log^3(n)}$ to obtain $\sum_{c\in\Psi_i^j} OPT_{FL}(P\cap c, f) \geq \sum_{c\in\Psi_i^j} \frac{\text{cost}_{FL}^{\epsilon}(c',f)}{(1+\epsilon)} \geq |\Psi_i^j|\cdot\frac{\tau}{(1+\epsilon)}\cdot f \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)}\cdot f$.

We claim that $n(\Psi_i^j) = \sum_{c\in\Psi_i^j} n_c \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$. As for the contradiction, suppose this claim is not correct. That is assume that $n(\Psi_i^j) < \frac{\alpha\sqrt{n}}{(1+\epsilon)}$. Then, imagine that we open one facility for each point in Ψ_i^j. Since we assume that $n(\Psi_i^j) \leq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$, the overall opening cost (in fact, the facility location cost since the connection cost is zero) of Ψ_i^j is less than $\frac{\alpha\sqrt{n}}{(1+\epsilon)}\cdot f$ which can not be the case. Thus, overall the cells in Ψ_i^j must have at least $\frac{\alpha\sqrt{n}}{(1+\epsilon)}$ points. ◀