# $(1+\varepsilon)$-Approximation for Facility Location in Data Streams[*]

Artur Czumaj[†] Christiane Lammersen[‡] Morteza Monemizadeh[§] Christian Sohler[¶]

## Abstract

We consider the *Euclidean facility location problem* with uniform opening cost. In this problem, we are given a set of $n$ points $P \subseteq \mathbb{R}^2$ and an opening cost $f \in \mathbb{R}^+$, and we want to find a set of facilities $F \subseteq \mathbb{R}^2$ that minimizes

$$f \cdot |F| + \sum_{p \in P} \min_{q \in F} d(p,q) \ ,$$

where $d(p,q)$ is the Euclidean distance between $p$ and $q$.

We obtain two main results:

- A $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}(n \log^2 n \log\log n)$ for constant $\varepsilon$,
- The first $(1+\varepsilon)$-approximation algorithm for the *cost* of the facility location problem for *dynamic geometric data streams*, i.e., when the stream consists of insert and delete operations of points from a discrete space $\{1,\dots,\Delta\}^2$. The streaming algorithm uses $\left(\frac{\log \Delta}{\varepsilon}\right)^{\mathcal{O}(1)}$ space.

Our PTAS is significantly faster than any previously known $(1+\varepsilon)$-approximation algorithm for the problem, and is also relatively simple. Our algorithm for dynamic geometric data streams is the first $(1+\varepsilon)$-approximation algorithm for the cost of the facility location problem with polylogarithmic space, and it resolves an open problem in the streaming area.

Both algorithms are based on a novel and simple decomposition of an input point set $P$ into small subsets $P_i$, such that:

- the cost of solving the facility location problem for each $P_i$ is small (which means that one needs to open only a small, polylogarithmic number of facilities),
- $\sum_i \mathsf{OPT}(P_i) \le (1+\varepsilon) \cdot \mathsf{OPT}(P)$, where for a point set $P$, $\mathsf{OPT}(P)$ denotes the cost of an optimal solution for $P$.

The partitioning can be used directly to obtain the PTAS by splitting the point set in the subsets and efficiently solve the problem for each subset independently. By combining our partitioning with techniques to process dynamic data streams of sampling from the cells of the partition and estimating the cost from the sample, we obtain our data streaming algorithm.

## 1 Introduction.

The *facility location problem* is one of the most extensively studied problems in combinatorial optimization and operations research (see, e.g., [3, 2]). The goal is for a given set of customers to find an optimal placement of facilities that minimizes the cost of customer service. Opening a facility at a certain location incurs a certain cost called *opening cost*, and serving customers at other locations incurs another type of cost called *connection cost*. The goal is to minimize the sum of both costs. Typical examples of applications include placement of servers in a network or location planning for stores, restaurants, etc. The focus of this paper is on the (uniform) facility location problem on the plane. Formally, the problem is defined as follows:

DEFINITION 1.1. *(*EUCLIDEAN UNIFORM FACILITY LOCATION PROBLEM*)*
*Let $P = \{p_1,\dots,p_n\}$ be a set of $n$ points in the Euclidean plane $\mathbb{R}^2$ and let $f \in \mathbb{R}^+$ be some fixed parameter (called* opening cost*). The* (uniform) facility location problem *is to find a set $F \subseteq \mathbb{R}^2$ of points (called* open facilities*) that minimizes*

$$\textit{COST}(P,F) = |F| \cdot f + \sum_{p \in P} d(p,F) \ , \qquad (1.1)$$

*where $d(p,F) = \min_{q \in F} d(p,q)$ is the Euclidean distance of p to the nearest open facility in F. (The first term on the right-hand side of (1.1) is called the* opening cost *and the second one is the* connection cost*.)*

Given a point set $P \subseteq \mathbb{R}^2$, we denote the optimal cost of $P$ by $\mathsf{OPT}_P$, and any set of optimal facilities by $F_{\mathsf{OPT}_P}$ (or $\mathsf{OPT}$ and $F_{\mathsf{OPT}}$, respectively, if $P$ is clear from the context), i.e.,

$$\mathsf{OPT}_P \quad = \quad \min_{F \subseteq \mathbb{R}^2} \mathsf{COST}(P,F) = \mathsf{COST}(P, F_{\mathsf{OPT}_P}) \ .$$

[*]Research partially supported by the EU within the 7th Framework Programme under contract No. 255827, an Ebco Eppich, a PIMS Fellowship, the Centre for Discrete Mathematics and its Applications (DIMAP), and by EPSRC awards EP/D063191/1 and EP/G064679/1. Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A2.

[†]Department of Computer Science and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, A.Czumaj@warwick.ac.uk.

[‡]School of Computing Science, Simon Fraser University, clammers@cs.sfu.ca.

[§]Institute for Computer Science, University of Frankfurt, monemizadeh@em.uni-frankfurt.de. Supported in part by BMBF grant 06FY9097.

[¶]Department of Computer Science, Technische Universität Dortmund, christian.sohler@tu-dortmund.de. Part of this work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A2.

We also consider the facility location problem for individual cells in $\mathbb{R}^2$. For any cell $\mathfrak{c} \in \mathbb{R}^2$, we define the cost of $\mathfrak{c}$ as follows:

$$\mathsf{COST}(\mathfrak{c}) \quad = \quad \min_{F \subseteq \mathbb{R}^2} \left\{ \sum_{p \in P \cap \mathfrak{c}} d(p, F) + |F| \cdot f \right\} \quad ,$$

where $P \cap \mathfrak{c}$ denotes the set of all points in $P$ that are contained in $\mathfrak{c}$.

Furthermore, we will also consider a contribution of any individual cell in $\mathbb{R}^2$ for optimal facilities $F_{\mathsf{OPT}_P}$. Given a point set $P \subseteq \mathbb{R}^2$ and a cell $\mathfrak{c} \subseteq \mathbb{R}^2$, for a fixed set o optimal facilities $F_{\mathsf{OPT}_P}$, let $\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c})$ be the connection cost in $\mathsf{OPT}_P$ of all the points from $P$ contained in $\mathfrak{c}$ plus the opening cost in $\mathsf{OPT}_P$ for the open facilities contained in $\mathfrak{c}$, i.e.,

$$\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}) \quad = \quad \sum_{p \in P \cap \mathfrak{c}} d(p, F_{\mathsf{OPT}_P}) + |F_{\mathsf{OPT}_P} \cap \mathfrak{c}| \cdot f \quad .$$

We will study the facility location problem both in the classical and in the streaming model of computation for dynamic geometric data streams. Since the problem is known to be $\mathbf{NP}$-hard [1], we will focus on *approximation* algorithms for the facility location. Furthermore, for the streaming model, given the structure of the problem (the output may be possibly of $\Omega(|P|)$ size), we will only approximate the *cost* of the facility location problem. Our motivation to consider the cost is

- in a dynamic scenario, for example, when maintaining a wireless network among moving robots, it is helpful to verify whether the current solution has still acceptable cost, and
- the solution computed by the facility location problem can also be viewed as a clustering problem with a regularization term for the number of cluster, i.e., one pays for each additional cluster a cost of $f$ and thus the cost of the facility location problem can be viewed as a structural property of the input data.

**1.1 Our contribution.** The main new technical contribution of this paper is a novel and simple partitioning scheme that allows to compute a good approximate solution to the facility location problem by taking the union of solutions for the partitioned input. That is, we can efficiently find a partition $\mathbf{\Lambda}$ of $[\Delta]^2$ such that $\sum_{\mathfrak{c} \in \mathbf{\Lambda}} \mathsf{COST}(\mathfrak{c}) \le (1 + \varepsilon) \cdot \mathsf{OPT}_P$. Furthermore, the partition is designed to enable to very quickly compute almost optimal solutions for the cells $\mathfrak{c} \in \mathbf{\Lambda}$. For that, we use the observation that if $\mathsf{COST}(\mathfrak{c})$ is small, then an $(1 + \varepsilon)$-approximation to $\mathsf{COST}(\mathfrak{c})$ can be found very quickly via a reduction to the $k$-median problem.

Using our new partitioning scheme, we develop two algorithms:

- an $\mathcal{O}(n \log^2 n \log \log n)$-time algorithm to compute a $(1 + \varepsilon)$-approximation for the planar Euclidean facility location problem with uniform cost for constant $\varepsilon$, and
- a $(1 + \varepsilon)$-approximation algorithm for the *cost* of the facility location problem in *dynamic geometric data streams*, i.e., when the stream consists of insert and delete operations of points from the discrete space $\{1, \dots, \Delta\}^2$, as introduced by Indyk [16]. The streaming algorithm uses $\left(\frac{\log \Delta}{\varepsilon}\right)^{\mathcal{O}(1)}$ space.

While polynomial-time approximation schemes for the planar Euclidean facility location problem with uniform cost have been known before [5, 18], our algorithm is arguably the simplest of those, and it also significantly faster. No $(1+\varepsilon)$-approximation algorithm for the cost of the facility location problem in dynamic geometric data streams with polylogarithmic space has been known before, and the existence of an $(1 + \varepsilon)$-approximation algorithm with polylogarithmic space for this problem has been an open problem in the area of geometric streaming algorithms. The best approximation guarantee obtained in earlier works was constant [21]. Our streaming algorithm samples cells from the obtained partitioning scheme and approximates the cost by taking the average of the costs of the sample. The algorithm itself extends the ideas developed earlier in [24] and [25], with several new observations needed to overcome some technical obstacles caused by using these approaches to obtain a $(1 + \varepsilon)$-approximation algorithm with polylogarithmic space. A byproduct of our work is an $\mathcal{O}(n \log n \log \log n)$-time algorithm that computes a constant-factor approximation for the facility location problem; the running time of this algorithm improves upon earlier constant-time approximation algorithms.

**1.2 Prior work.** The facility location problem has been extensively studied before in the combinatorial optimization and operations research. The problem is known to be $\mathbf{NP}$-hard even in the geometric case. For general metrics, Hochbaum [14] gave the first non-trivial polynomial-time $\mathcal{O}(\log n)$-approximation algorithm and Shmoys, Tardos and Aardal [23] gave the first constant-factor approximation algorithm for arbitrary metric spaces; the approximation factor has since been improved several times and the currently best polynomial-time approximation algorithm due to Li [20] achieves a 1.488-approximation. On the negative side, Guha and Khuller [12] showed that there is no polynomial-time $\alpha$-approximation algorithm for $\alpha < 1.463$, unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\mathcal{O}(\log \log n)})$.

The facility location problem has been also studied in the geometric setting. Arora, Raghavan, and Rao [5] gave the first PTAS for the geometric version of the problem, with a randomized algorithm achieving an $(1+\varepsilon)$-approximation in $\mathcal{O}(n^{1+\mathcal{O}(1/\varepsilon)}\log n)$ time. This has been later improved by Kolliopoulos and Rao [18], who designed an $\mathcal{O}(n\log^8 n\, 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)})$-time algorithm that outputs in expectation a $(1+\varepsilon)$-approximate solution[1].

Facility location has been also studied in the context of data streaming, initiated with work of Indyk [16], who gave a $\mathrm{poly}(\log\Delta)$-space streaming algorithm that approximates the optimal cost of the facility location problem within a factor of $\mathcal{O}(\log^2\Delta)$. The best currently known streaming algorithm using $\mathrm{poly}(\log\Delta)$ space is due to Lammersen and Sohler [21], which gives a $\mathcal{O}(1)$-approximation for this problem.

The facility location problem is closely related to the $k$-median problem. In geometric setting, Arora [4] gave the first PTAS, finding an $(1+\varepsilon)$-approximation in $\mathcal{O}(n^{1+\mathcal{O}(1/\varepsilon)})$ time. The runtime was later improved in a series of papers, culminating with the running-time of $\mathcal{O}(n+2^{\mathcal{O}((1+\log(1/\varepsilon))/\varepsilon)}k^{\mathcal{O}(1)}\log^{\mathcal{O}(1)} n)$ by Har-Peled and Mazumdar [15] and $\mathcal{O}(nk+2^{(k/\varepsilon)^{\mathcal{O}(1)}}\log^{k+2} n)$ by Chen [7].

## 2 Partitioning scheme.

In this section we present a new partitioning scheme for the Euclidean uniform facility location problem. For simplicity of exposition, we assume a real number is always rounded up to the next integer number. Furthermore, we will assume that the input points are drawn from a grid of size $\Delta$, that is, from $\{0,1,\dots,\Delta\}^2$. (While this may seem to be restrictive, this assumption can be made without loss of generality in most of the applications we can think about; see, e.g., Remark 3.1 below.) We assume that $\Delta$ is a power of 2. For a number $x$, let $[x]=\{0,1,\dots,x\}$, and so, any input point is in $[\Delta]^2$.

For any cell $\mathfrak{c}$, we denote the side length of $\mathfrak{c}$ by $\ell_{\mathfrak{c}}$ and the number of points inside $\mathfrak{c}$ by $n_{\mathfrak{c}}$. For any cell $\mathfrak{c}$ and any set $P$ of points on the plane, $P\cap\mathfrak{c}$ denotes the set of all points in $P$ that are contained in $\mathfrak{c}$.

**Randomly shifted nested grid partition.** Our construction will assume that the input points are partitioned with respect to a nested grid dissection. To ensure appropriate properties of the input points, we will assume that the dissection is chosen at *random*. (This is somehow similar to the dissection used by Arora [4] in a PTAS for TSP and in related works.)

To define our dissection, we first place a nested grid set $G_0,\dots,G_i,\dots,G_{\log\Delta+1}$ on the input cell $[2\Delta]^2$ (which is twice the size of the original input cell) so that a cell $\mathfrak{c}$ in a grid $G_i$ has side length $\ell_{\mathfrak{c}}=2^i$. That is, $G_{\log\Delta+1}$ has a single cell $\mathfrak{c}$ with side length $\ell_{\mathfrak{c}}=2\Delta$, and any cell $\mathfrak{c}$ in $G_i$ is split into four sub-cells in $G_{i-1}$.

Next, we randomly shift the dissection. We choose a pair of integers $a,b\in\{0,1,\dots,\Delta\}$ independently and uniformly at random, and then move each point $p\in P$ by the vector $(a,b)$.

We assume we have a unique numbering for cells in $G_i$ so that when we refer to a cell $\mathfrak{c}\in G_i$ we mean that this cell has a unique number $\mathfrak{c}$ for $1\le\mathfrak{c}\le|G_i|$ according to this numbering. We call the four subcells of $\mathfrak{c}\in G_i$, which are in the grid $G_{i-1}$, the *children* of $\mathfrak{c}$, and $\mathfrak{c}$ is the *parent* cell of each one of these four subcells.

**Heavy and light cells.** In our construction, we will need a deterministic constant-factor approximation algorithm for the cost facility location problem $\mathbb{A}$. (In specific applications, as we will use it later in the paper, for our PTAS, $\mathbb{A}$ will be the algorithm developed in this paper in Section 4, and for the streaming algorithm, we will assume that $\mathbb{A}$ is an algorithm that computes optimally the cost facility location problem.)

For any cell $\mathfrak{c}$ in a grid $G_i$, $0\le i\le\log\Delta+1$, let us call $\mathfrak{c}$ *heavy* if $\mathbb{A}$ returns that the cost of $\mathfrak{c}$ is greater than $\aleph f$; otherwise we call $\mathfrak{c}$ *light*. We will require that $\aleph\ge c^*\cdot\left(\frac{\log\Delta+2}{\varepsilon}\right)^2$, for a constant $c^*$ defined later.

**Partition $\mathbf{\Lambda}$ of** $[2\Delta]^2$**.** We define a hierarchical partition $\mathbf{\Lambda}$ of $[2\Delta]^2$ into cells from a grid $G_i$, $0\le i\le\log\Delta+1$, to be the set of all cells $\mathfrak{c}\in\bigcup_{i=0}^{\log\Delta}G_i$ that are light and whose all ancestors are heavy. More formally, we use a recursive procedure *Partition*($\mathfrak{c}$), which we initially call with $\mathfrak{c}$ being the entire square $[2\Delta]^2$:

```
Partition (𝔠) ► Recursively finds a nested partition of cell 𝔠
  • Consider cell $\mathfrak{c}$ with four children $\mathfrak{c}_1$, $\mathfrak{c}_2$, $\mathfrak{c}_3$, $\mathfrak{c}_4$.
  • If cell $\mathfrak{c}$ is heavy, then
      – recursively call Partition($\mathfrak{c}_1$), Partition($\mathfrak{c}_2$), Partition($\mathfrak{c}_3$), and Partition($\mathfrak{c}_4$),
      – return the union of the obtained partitions as the partition of $\mathfrak{c}$.
  • Else, if $\mathfrak{c}$ is light, then
      – return $\{\mathfrak{c}\}$ itself as the partition of $\mathfrak{c}$.
```

The procedure above returns a partition $\mathbf{\Lambda}$ of $[2\Delta]^2$ into a set of cells from grids $G_i$, $0\le i\le\log\Delta+1$.

[1]A preliminary version of that paper [19] claimed a faster running time, but this has been later retracted and corrected in [18].

Now we present our main theorem describing key properties of our partition $\mathbf{\Lambda}$.

THEOREM 2.1. (PARTITION THEOREM) *Let $P$ be any set of points in $[\Delta]^2$ that is randomly shifted in $[2\Delta]^2$. Let $\varrho \in (0,1)$ be any constant. Partition $\mathbf{\Lambda}$ of $[2\Delta]^2$ satisfies the following conditions:*

(1) *there exists a constant $c$ such that for every cell $\mathfrak{c} \in \mathbf{\Lambda}$, $\mathit{COST}(\mathfrak{c}) \le \frac{c\cdot(\log\Delta+2)^2}{\varepsilon^2} \cdot f$,*

(2) *$\sum_{\mathfrak{c}\in\mathbf{\Lambda}} \mathit{COST}(\mathfrak{c}) \le (1+\varepsilon) \cdot \mathit{OPT}_P$ with probability at least $1-\varrho$.*

The rest of this section is devoted to the proof of Theorem 2.1. We first introduce useful notation in Section 2.1, then we prove basic properties of our construction in Sections 2.2 and 2.3, and we finalize the proof in Section 2.4.

In the entire analysis below we assume that $F_{\mathrm{OPT}_P}$ is a *fixed* optimal set of facilities, and with that, $\mathrm{COST}_{\mathrm{OPT}_P}(\mathfrak{c})$ is well defined for any cell $\mathfrak{c}$.

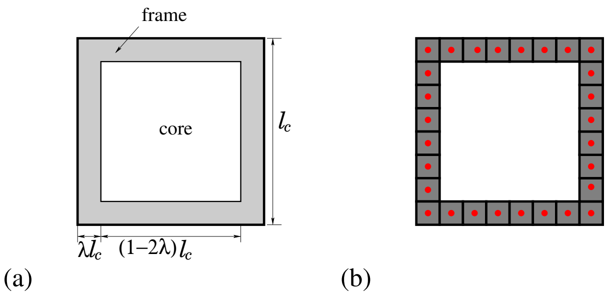

Figure 1: (a) Partition of a cell $\mathfrak{c}$ into a core and a frame, and (b) the buffer-net of $\mathbb{I}_\mathfrak{c}$.

**2.1 Frames, cores, and buffer-nets.** Let us fix $\lambda = \frac{\varepsilon}{12\cdot(1+3/\varrho)\cdot(\log\Delta+2)}$. A *core* of a cell $\mathfrak{c}$ is the $(1-2\lambda)\ell_\mathfrak{c} \times (1-2\lambda)\ell_\mathfrak{c}$ square with the center in the center of $\mathfrak{c}$. A *frame* of a cell $\mathfrak{c}$ is obtained from $\mathfrak{c}$ by removing the core of $\mathfrak{c}$. See Figure 1.a. (In other words, a frame of $\mathfrak{c}$ consists of all points in the plane contained in $\mathfrak{c}$ that are at distance less than $\lambda\cdot\ell_\mathfrak{c}$ from the boundary of $\mathfrak{c}$, and a core of $\mathfrak{c}$ consists of all points in the plane contained in $\mathfrak{c}$ that are at distance greater than or equal to $\lambda\cdot\ell_\mathfrak{c}$ from the boundary of $\mathfrak{c}$.) For a cell $\mathfrak{c}$, let $\mathbb{I}_\mathfrak{c}$ denote its core and $\mathbb{B}_\mathfrak{c}$ denote its frame.

**Buffer-nets.** In our construction, we will frequently allow to open new facilities around either a core or a frame of a cell. Consider any cell $\mathfrak{c}$ and its core $\mathbb{I}_\mathfrak{c}$ and frame $\mathbb{B}_\mathfrak{c}$. $\mathbb{B}_\mathfrak{c}$ may be seen as union of four stripes around $\mathbb{I}_\mathfrak{c}$ of size $(\lambda\cdot\ell_\mathfrak{c})\times\ell_\mathfrak{c}$ each (cf. Figure 1.a). Place $1/\lambda$ points in each stripe, by first partitioning each stripe into $1/\lambda$ disjoint $(\lambda\cdot\ell_\mathfrak{c})\times(\lambda\cdot\ell_\mathfrak{c})$ squares and then choosing the center of each square (cf. Figure 1.b). Call the obtained points the *buffer-net* of $\mathbb{I}_\mathfrak{c}$. Note that each buffer-net of a core has $4(\frac{1}{\lambda}-1)$ points.

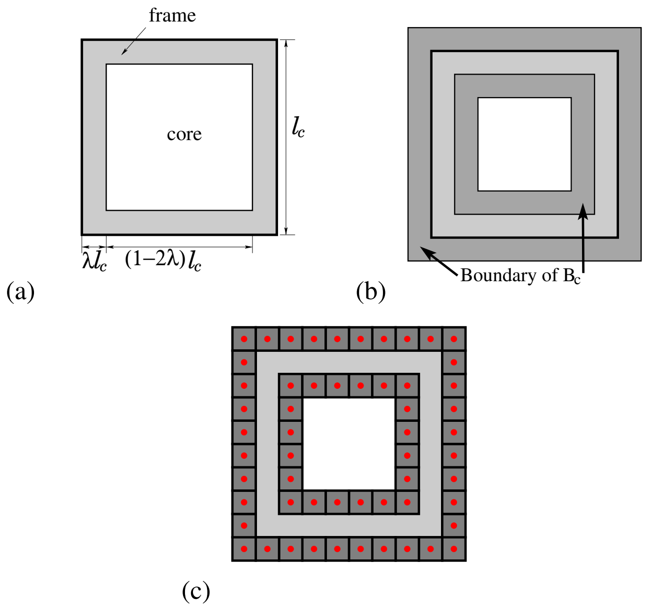

Figure 2: (a) Partition of a cell $\mathfrak{c}$ into a core and a frame, (b) the partition with the boundary, that is, an extension of $\mathbb{B}_\mathfrak{c}$ through the stripes of width $\lambda\cdot\ell_\mathfrak{c}$ on the outside and inside of $\mathbb{B}_\mathfrak{c}$, and (c) the corresponding buffer-net of $\mathbb{B}_\mathfrak{c}$.

A similar construction can be done for $\mathbb{B}_\mathfrak{c}$. Let us draw eight stripes of width $\lambda\cdot\ell_\mathfrak{c}$ around the boundary of $\mathbb{B}_\mathfrak{c}$: four $(\lambda\cdot\ell_\mathfrak{c})\times((1+2\lambda)\cdot\ell_\mathfrak{c})$ stripes outside $\mathbb{B}_\mathfrak{c}$ and four $(\lambda\cdot\ell_\mathfrak{c})\times((1-2\lambda)\cdot\ell_\mathfrak{c})$ stripes inside $\mathbb{B}_\mathfrak{c}$ (see Figure 2.b). (We call the union of the stripes the *boundary* of $\mathbb{B}_\mathfrak{c}$ and denote $\mathbb{Q}_\mathfrak{c}$. [2]) Then, we partition each stripe into disjoint $(\lambda\cdot\ell_\mathfrak{c})\times(\lambda\cdot\ell_\mathfrak{c})$ squares and place points in the center of each square (see Figure 2.c). The obtained points are called the *buffer-net* of $\mathbb{B}_\mathfrak{c}$. Note that each buffer-net of a frame has $8(\frac{1}{\lambda}-1)$ points.

The following key properties of buffer-nets will be used in the paper.

CLAIM 2.1. (BUFFER-NET PROPERTY FOR CORES) *For any cell $\mathfrak{c}$, any point $p\in\mathbb{I}_\mathfrak{c}$, and any point $q\notin\mathfrak{c}$, there is a point $x$ in the buffer-net of $\mathbb{I}_\mathfrak{c}$ such that*

$$d(p,x) \le d(p,q) \ .$$

[2]We will ignore any part of the boundary of $\mathbb{B}_\mathfrak{c}$ that lies outside of the original square $[2\Delta]^2$.

*Proof.* Let $z$ be the minimum distance of $p$ to the frame of $\mathfrak{c}$. Since the frame of $\mathfrak{c}$ has a width of $\lambda\ell_{\mathfrak{c}}$ and due to the construction of the buffer-net of $\mathbb{I}_{\mathfrak{c}}$, the distance of any point from the border of the frame of $\mathfrak{c}$ to the closest point in the buffer-net of $\mathbb{I}_{\mathfrak{c}}$ is at most $\sqrt{2}\lambda\ell_{\mathfrak{c}}/2$. Thus, by triangle inequality, the distance of $p$ to its closest point $x$ in the buffer-net of $\mathbb{I}_{\mathfrak{c}}$ is at most $z + \sqrt{2}\lambda\ell_{\mathfrak{c}}/2 < z + \lambda\ell_{\mathfrak{c}}$. Since the frame of $\mathfrak{c}$ has a width of $\lambda\ell_{\mathfrak{c}}$, the distance of $p$ to the closest point $q \notin \mathfrak{c}$ is at least $z + \lambda\ell_{\mathfrak{c}}$.

CLAIM 2.2. (BUFFER-NET PROPERTY FOR FRAMES) *For any cell $\mathfrak{c}$, any point $p \in \mathbb{B}_{\mathfrak{c}}$, and any point $q$ that is not contained in $\mathbb{B}_{\mathfrak{c}}$ nor in the boundary of $\mathbb{B}_{\mathfrak{c}}$, there is a point $x$ in the buffer-net of $\mathbb{B}_{\mathfrak{c}}$ with $d(p,x) \le d(p,q)$.*

*Proof.* The proof is the same as that of Claim 2.1, with the only difference that $p \in \mathbb{B}_{\mathfrak{c}}$ and $q$ is a point that is not contained in $\mathbb{B}_{\mathfrak{c}}$ nor in the boundary of $\mathbb{B}_{\mathfrak{c}}$ and $z$ is the minimum distance of $p$ to the boundary of $\mathbb{B}_{\mathfrak{c}}$.

## 2.2 Bounding the cost of cells.

We will now prove three auxiliary lemmas bounding the cost of cores and frames.

LEMMA 2.1. *For every cell $\mathfrak{c}$, we have*

$$\textit{COST}(\mathbb{I}_{\mathfrak{c}}) \le \textit{COST}_{\textit{OPT}_P}(\mathfrak{c}) + \frac{4f}{\lambda} \quad .$$

*Proof.* We will construct a feasible solution for $\mathbb{I}_{\mathfrak{c}}$ with the cost upper bounded by $\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}) + 4\left(\frac{1}{\lambda} - 1\right) f$.

For every point $p \in P \cap \mathbb{I}_{\mathfrak{c}}$, if its nearest open facility in $F_{\mathsf{OPT}_P}$ is contained in $\mathfrak{c}$, then we connect $p$ to that facility. Otherwise, we connect $p$ to the nearest point in the buffer-net of $\mathbb{I}_{\mathfrak{c}}$. Claim 2.1 ensures that in the latter case, the chosen point $q$ from the buffer-net of $\mathbb{I}_{\mathfrak{c}}$ satisfies $d(p,q) \le d(p, F_{\mathsf{OPT}_P})$.

We have constructed a feasible solution to the facility location for $\mathbb{I}_{\mathfrak{c}}$ whose cost is upper bounded by $\sum_{p \in P \cap \mathbb{I}_{\mathfrak{c}}} d(p, F_{\mathsf{OPT}_P})$ plus the cost of opening facilities in $F_{\mathsf{OPT}_P} \cap \mathfrak{c}$ plus the cost of opening some facilities in the buffer-net of $\mathbb{I}_{\mathfrak{c}}$. Since the number of points in the buffer-net of $\mathbb{I}_{\mathfrak{c}}$ is at most $4\left(\frac{1}{\lambda} - 1\right)$ and since

$$\begin{aligned} \mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}) &= \sum_{p \in P \cap \mathfrak{c}} d(p, F_{\mathsf{OPT}_P}) + |F_{\mathsf{OPT}_P} \cap \mathfrak{c}| \cdot f \\ &\ge \sum_{p \in P \cap \mathbb{I}_{\mathfrak{c}}} d(p, F_{\mathsf{OPT}_P}) + |F_{\mathsf{OPT}_P} \cap \mathfrak{c}| \cdot f \quad , \end{aligned}$$

we obtain the following:

$$\begin{aligned} &\mathsf{COST}(\mathbb{I}_{\mathfrak{c}}) \\ \le\ & \sum_{p \in P \cap \mathbb{I}_{\mathfrak{c}}} d(p, F_{\mathsf{OPT}_P}) + |F_{\mathsf{OPT}_P} \cap \mathfrak{c}| \cdot f + 4\left(\tfrac{1}{\lambda} - 1\right) f \\ \le\ & \mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}) + 4\left(\tfrac{1}{\lambda} - 1\right) f \quad . \end{aligned}$$

LEMMA 2.2. *For every cell $\mathfrak{c}$, we have*

$$\textit{COST}(\mathbb{B}_{\mathfrak{c}}) \le \textit{COST}_{\textit{OPT}_P}(\mathbb{B}_{\mathfrak{c}} \cup \mathbb{Q}_{\mathfrak{c}}) + \frac{8f}{\lambda} \quad ,$$

*and*

$$\textit{COST}(\mathfrak{c}) \le \textit{COST}_{\textit{OPT}_P}(\mathfrak{c} \cup \mathbb{Q}_{\mathfrak{c}}) + \frac{8f}{\lambda} \quad .$$

*Proof.* The proof is the same as that of Lemma 2.1, with the only difference in a different number of points in the buffer-net of $\mathbb{B}_{\mathfrak{c}}$, which is now $8\left(\frac{1}{\lambda} - 1\right)$ (versus $4\left(\frac{1}{\lambda} - 1\right)$ in the proof of Lemma 2.1), and in the use of Claims 2.1 and 2.2 instead of solely relying on Claim 2.1.

LEMMA 2.3. *For any cell $\mathfrak{c} \in \mathbf{\Lambda}$, the following bound holds:*

$$\mathbf{E}\Big[\sum_{\mathfrak{c} \in \mathbf{\Lambda}} \textit{COST}_{\textit{OPT}_P}(\mathbb{B}_{\mathfrak{c}} \cup \mathbb{Q}_{\mathfrak{c}})\Big] \le 12\lambda(\log\Delta + 2) \cdot \textit{OPT}_P \quad ,$$

*where the expectation is over the choice of the random choice of the shift of the dissection.*

*Proof.* Let us take the nested grids $G_0, G_1, \dots, G_{\log\Delta+1}$, and all cells in each $G_i$. For each cell, let us take its frame and let $\mathfrak{W}_i$ be the union of frames and their boundaries for all cells in $G_i$. It is easy to see that the area of each $\mathfrak{W}_i$ is at most $12\lambda$ fraction of the total area (the area of all frames is at most $4\lambda$ fraction of the total area and the area of all boundaries of the frames is at most $8\lambda$ fraction of the total area). Hence, for any point $p \in P \cup F_{\mathsf{OPT}_P}$, the probability that $p \in \mathfrak{W}_i$ is at most $12\lambda$. Indeed, for any *single* $(1 \times 1)$ cell $\mathfrak{c} \in G_0$ in $[2\Delta]^2$, $\mathbf{Pr}[p \in \mathfrak{c}] \le \frac{1}{(2\Delta)^2}$. Furthermore, the number of single cells $\mathfrak{c}$ that are in $\mathfrak{W}_i$ is at most $12\lambda \cdot (2\Delta)^2$. Hence, we have

$$\mathbf{Pr}[p \in \mathfrak{W}_i] \le 12\lambda.$$

For every $p \in P \cup F_{\mathsf{OPT}_P}$, let us define $val(p)$ as follows:

- if $p \in F_{\mathsf{OPT}_P}$, then $val(p) = f$;
- if $p \notin F_{\mathsf{OPT}_P}$ then $val(p) = d(p, F_{\mathsf{OPT}_P})$.

Observe that $\mathsf{OPT}_P = \sum_{p \in P \cup F_{\mathsf{OPT}_P}} val(p)$. For every $p \in P \cup F_{\mathsf{OPT}_P}$, let $X_p$ be the indicator random variable for the event $p \in \bigcup_{\mathfrak{c} \in \mathbf{\Lambda}} \mathbb{B}_{\mathfrak{c}} \cup \mathbb{Q}_{\mathfrak{c}}$, and let $Y_p$ be the indicator random variable for the event $p \in \bigcup_i \mathfrak{W}_i$. Notice that $\mathbf{E}[X_p] \le \mathbf{E}[Y_p]$.

Let us consider the *contribution* of a single point $p \in P \cup F_{\mathsf{OPT}_P}$ to $\mathbf{E}\big[\sum_{\mathfrak{c} \in \mathbf{\Lambda}} \mathsf{COST}_{\mathsf{OPT}_P}(\mathbb{B}_{\mathfrak{c}} \cup \mathbb{Q}_{\mathfrak{c}})\big]$. If $p \notin \bigcup_{\mathfrak{c} \in \mathbf{\Lambda}} \mathbb{B}_{\mathfrak{c}} \cup \mathbb{Q}_{\mathfrak{c}}$, then the contribution is none; otherwise, the contribution is equal to $val(p)$.

Hence,

$$\begin{aligned}
&\mathbf{E}[\sum_{\mathfrak{c}\in\mathbf{\Lambda}} \mathsf{COST}_{\mathsf{OPT}_P}(\mathbb{B}_\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})] \\
= &\sum_{p\in P\cup F_{\mathsf{OPT}_P}} \mathbf{E}[X_p]\cdot val(p) \\
\le &\sum_{p\in P\cup F_{\mathsf{OPT}_P}} \mathbf{E}[Y_p]\cdot val(p) \\
\le &\sum_{p\in P\cup F_{\mathsf{OPT}_P}} (12\cdot\lambda\cdot(\log\Delta+2))\cdot val(p) \\
= &\ 12\,\lambda\,(\log\Delta+2)\cdot\mathsf{OPT}_P\ .
\end{aligned}$$

**2.3 Bounding the size of $\mathbf{\Lambda}$.** Our next goal will be to estimate the two sums $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\mathsf{COST}(\mathbb{I}_\mathfrak{c})$ and $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\mathsf{COST}(\mathbb{B}_\mathfrak{c})$ in terms of $\mathsf{OPT}_P$. For that we first bound (in Lemma 2.4 below) the number of cells in the partition $\mathbf{\Lambda}$, as a function of $\mathsf{OPT}_P$ and $f$.

We begin with a simple claim that follows directly from the definition of light cells.

CLAIM 2.3. *If the algorithm $\mathbb{A}$ returns an $\alpha$-approximation of the cost of the facility location problem*[3]*, then for every light cell $\mathfrak{c}$ and for every heavy cell $\mathfrak{c}^*$, we have $\mathsf{COST}(\mathfrak{c})\le\alpha\aleph f$ and $\mathsf{COST}(\mathfrak{c}^*)>\aleph f$.*

We call a cell $\mathfrak{c}\in\bigcup_{i=0}^{\log\Delta+1}G_i$ *loaded* if

$$\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})\ge(\aleph-8/\lambda)f.$$

CLAIM 2.4. *If $\mathfrak{c}$ is a heavy cell, then $\mathfrak{c}$ is loaded.*

*Proof.* If $\mathfrak{c}$ is a heavy cell, then by Lemma 2.2,

$$\mathsf{COST}(\mathfrak{c})\le\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})+\frac{8f}{\lambda}\ ,$$

and by Claim 2.3, we obtain $\mathsf{COST}(\mathfrak{c})>\aleph f$. Hence,

$$\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})\ge\mathsf{COST}(\mathfrak{c})-\frac{8\cdot f}{\lambda}>\aleph f-\frac{8\cdot f}{\lambda}.$$

We call a loaded cell $\mathfrak{c}$ *marked* if either (i) $\mathfrak{c}\in\mathbf{\Lambda}$, or (ii) if $\mathfrak{c}$ has no ancestor in $\mathbf{\Lambda}$ and no descendant of $\mathfrak{c}$ is loaded. We will now state some basic properties of marked cells.

CLAIM 2.5. *All marked cells are disjoint.*

*Proof.* Let $\mathfrak{c}$ and $\mathfrak{c}'$ be any two marked cells, $\mathfrak{c}\ne\mathfrak{c}'$. They are not disjoint only if one of them is an ancestor of another. If $\mathfrak{c},\mathfrak{c}'\in\mathbf{\Lambda}$, then since $\mathbf{\Lambda}$ is a partition of $[2\Delta]^2$, we have $\mathfrak{c}$ and $\mathfrak{c}'$ disjoint. If none of $\mathfrak{c},\mathfrak{c}'$ is in $\mathbf{\Lambda}$, then since neither one of them can be an ancestor of another, again we have $\mathfrak{c}$ and $\mathfrak{c}'$ disjoint. Finally, if $\mathfrak{c}\in\mathbf{\Lambda}$ and $\mathfrak{c}'\notin\mathbf{\Lambda}$, then the definition of $\mathfrak{c}'$ being marked implies that $\mathfrak{c}$ cannot be an ancestor of $\mathfrak{c}'$ because $\mathfrak{c}\in\mathbf{\Lambda}$, and $\mathfrak{c}$ cannot be a descendant of $\mathfrak{c}'$ because $\mathfrak{c}$ is loaded.

We have also the following claim.

CLAIM 2.6. *If $\mathfrak{c}$ is a loaded cell and has no proper ancestor in $\mathbf{\Lambda}$, then $\mathfrak{c}$ has a marked descendant (which might be equal to $\mathfrak{c}$).*

In our analysis, we will assign every cell $\mathfrak{c}\in\mathbf{\Lambda}$ to a marked cell $\vartheta(\mathfrak{c})$ as follows. If $\mathfrak{c}\in\mathbf{\Lambda}$ is marked, then $\vartheta(\mathfrak{c})=\mathfrak{c}$. Otherwise, let us consider the parent cell $\mathfrak{c}^*$ of cell $\mathfrak{c}$. Since cell $\mathfrak{c}^*$ is heavy by the definition of $\mathbf{\Lambda}$, Claim 2.4 implies that $\mathfrak{c}^*$ is loaded, and then Claim 2.6 ensures that $\mathfrak{c}^*$ has a marked descendant. Let $\mathfrak{c}'$ be any of the marked descendants of $\mathfrak{c}^*$; we set $\vartheta(\mathfrak{c})=\mathfrak{c}'$.

CLAIM 2.7. *For every cell $\mathfrak{c}\in\mathbf{\Lambda}$,*

$$\mathit{COST}_{\mathit{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})\le\mathit{COST}_{\mathit{OPT}_P}(\vartheta(\mathfrak{c})\cup\mathbb{Q}_{\vartheta(\mathfrak{c})})\ .$$

*Proof.* Observe that if $\mathfrak{c}\in\mathbf{\Lambda}$ is loaded, then our construction ensures that $\vartheta(\mathfrak{c})=\mathfrak{c}$, and hence

$$\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})=\mathsf{COST}_{\mathsf{OPT}_P}(\vartheta(\mathfrak{c})\cup\mathbb{Q}_{\vartheta(\mathfrak{c})})\ .$$

Otherwise, if $\mathfrak{c}\in\mathbf{\Lambda}$ is not loaded, then $\mathsf{COST}_{\mathsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_\mathfrak{c})<(\aleph-8/\lambda)f$, and since $\vartheta(\mathfrak{c})$ is marked,

$$\mathsf{COST}_{\mathsf{OPT}_P}(\vartheta(\mathfrak{c})\cup\mathbb{Q}_{\vartheta(\mathfrak{c})})\ge(\aleph-8/\lambda)f\ ,$$

from which the claim follows.

CLAIM 2.8. *For every marked cell $\mathfrak{c}$, the number of cells $\mathfrak{c}'\in\mathbf{\Lambda}$ for which $\vartheta(\mathfrak{c}')=\mathfrak{c}$, is at most $3(\log\Delta+1)+1$.*

*Proof.* If $\mathfrak{c}=\vartheta(\mathfrak{c}')$, then the parent of cell $\mathfrak{c}'$ must be an ancestor of $\mathfrak{c}$ (not necessarily a proper ancestor, that is, it is possible that $\mathfrak{c}$ is the parent of $\mathfrak{c}'$). Hence, $\mathfrak{c}'$ is a child of an ancestor of $\mathfrak{c}$. Since there are at most $\log\Delta+2$ ancestors of $\mathfrak{c}$, and each of them (except the one at the most bottom level, in $G_0$) has four children, there are at most $4(\log\Delta+1)+1$ cells $\mathfrak{c}'$ for which we may have $\vartheta(\mathfrak{c}')=\mathfrak{c}$. (We have term "1" to count the case when $\mathfrak{c}\in G_0$.) We can reduce this bound further by observing that no ancestor $\mathfrak{c}^*$ of $\mathfrak{c}$ will have $\vartheta(\mathfrak{c}^*)=\mathfrak{c}$, because all ancestors of $\mathfrak{c}$ are loaded. Therefore, for every proper ancestor of $\mathfrak{c}$ there are at most three children $\mathfrak{c}'$ for which we may have $\vartheta(\mathfrak{c}')=\mathfrak{c}$, and hence the number of possible cells $\mathfrak{c}'$ for which we may have $\vartheta(\mathfrak{c}')=\mathfrak{c}$ is upper bounded by $3(\log\Delta+1)+1$.

[3]That is, for any set $X$ of points in the plane, $\mathbb{A}$ computes a value $\mathsf{COST}_{\mathbb{A}_\alpha}(X)$ such that $\mathsf{OPT}_X\le\mathsf{COST}_{\mathbb{A}_\alpha}(X)\le\alpha\cdot\mathsf{OPT}_X$.

CLAIM 2.9. *For any set $\mathcal{X}$ of disjoint cells in $[2\Delta]^2$, $\sum_{\mathfrak{c}\in\mathcal{X}} \textsf{COST}_{\textsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_{\mathfrak{c}}) \le 5\cdot \textsf{OPT}_P$.*

*Proof.* This follows from the fact that every point $p\in P$ appears in at most five terms in

$$\begin{aligned}&\sum_{\mathfrak{c}\in\mathcal{X}} \textsf{COST}_{\textsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_{\mathfrak{c}})\\ =&\sum_{\mathfrak{c}\in\mathcal{X}}\sum_{p\in P\cap(\mathfrak{c}\cup\mathbb{Q}_{\mathfrak{c}})}(d(p,F_{\textsf{OPT}_P})+|F_{\textsf{OPT}_P}\cap(\mathfrak{c}\cup\mathbb{Q}_{\mathfrak{c}})|f).\end{aligned}$$

Indeed, once if $p\in\mathfrak{c}$, and at most four times if $p$ is in the left, the right, the top, and the bottom boundary of some cell $\mathfrak{c}'$.

With this, we are ready to prove the following upper bound on the number of cells in $\mathbf{\Lambda}$.

LEMMA 2.4.

$$|\mathbf{\Lambda}| \le \frac{15\cdot(\log\Delta+2)\cdot \textsf{OPT}_P}{(\aleph-\frac{8}{\lambda})\cdot f}\ .$$

*Proof.*

$$\begin{aligned}&15\cdot(\log\Delta+2)\cdot\textsf{OPT}_P\\ \ge\ &3\cdot(\log\Delta+2)\cdot\sum_{\mathfrak{c}\text{ is marked}}\textsf{COST}_{\textsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_{\mathfrak{c}})\\ \ge\ &\sum_{\mathfrak{c}\text{ is marked}}\textsf{COST}_{\textsf{OPT}_P}(\mathfrak{c}\cup\mathbb{Q}_{\mathfrak{c}})\cdot|\{\mathfrak{c}'\in\mathbf{\Lambda}:\vartheta(\mathfrak{c}')=\mathfrak{c}\}|\\ \ge\ &\sum_{\mathfrak{c}\text{ is marked}}((\aleph-\tfrac{8}{\lambda})\cdot f)\cdot|\{\mathfrak{c}'\in\mathbf{\Lambda}:\vartheta(\mathfrak{c}')=\mathfrak{c}\}|\\ =\ &(\aleph-\tfrac{8}{\lambda})\cdot f\cdot|\mathbf{\Lambda}|\ ,\end{aligned}$$

where the first inequality follows from Claims 2.5 and 2.9, the second one from Claim 2.8, the third one from Claim 2.4 (see the proof of the claim), and the last identity follows from the fact that

$$\sum_{\mathfrak{c}\text{ is marked}}|\{\mathfrak{c}'\in\mathbf{\Lambda}:\vartheta(\mathfrak{c}')=\mathfrak{c}\}|=|\mathbf{\Lambda}|\ .$$

This yields the proof.

### 2.4 Completing the proof of Partition Theorem 2.1.

Now we are ready to complete the proof of the Partition Theorem 2.1. We will first fix the constant $\varrho$ for the probability bound, and then specify the values for $\lambda$ and $\aleph$: we have

$$\lambda=\frac{\varepsilon}{12\cdot(1+3/\varrho)\cdot(\log\Delta+2)}$$

and we will require that

$$\aleph\ge\frac{732\cdot(1+3/\varrho)^2\cdot(\log\Delta+2)^2}{\varepsilon^2}\ .$$

Part (1) of the Partition Theorem 2.1 follows immediately from Claim 2.3 and from the fact that we have assumed that the algorithm $\mathbb{A}$ used to define heavy and light cells is a constant-factor approximation algorithm for the cost facility location problem.

The proof of part (2) of the Partition Theorem 2.1 is split into two parts, one bounding $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{I}_{\mathfrak{c}})$ and another one bounding $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{B}_{\mathfrak{c}})$.

In order to upper bound $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{I}_{\mathfrak{c}})$, we use Lemmas 2.1 and 2.4 to obtain the following:

$$\begin{aligned}\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{I}_{\mathfrak{c}})&\le\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\left(\textsf{COST}_{\textsf{OPT}_P}(\mathfrak{c})+\frac{4f}{\lambda}\right)\\ &=\textsf{OPT}_P+|\mathbf{\Lambda}|\cdot\frac{4f}{\lambda}\\ &\le\textsf{OPT}_P+\frac{60\cdot(\log\Delta+2)}{\lambda\aleph-8}\cdot\textsf{OPT}_P\ .\end{aligned}$$

Next, our bounds for $\lambda$ and $\aleph$ yield the following:

$$\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{I}_{\mathfrak{c}})\le\textsf{OPT}_P+\frac{60(\log\Delta+2)}{\lambda\aleph-8}\textsf{OPT}_P$$

which is

$$\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{I}_{\mathfrak{c}})\le(1+\tfrac{\varepsilon}{1+3/\varrho})\cdot\textsf{OPT}_P\ . \tag{2.2}$$

Next, we upper bound $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{B}_{\mathfrak{c}})$. We have the following:

$$\begin{aligned}\mathbf{E}\Big[\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{B}_{\mathfrak{c}})\Big]&\le\mathbf{E}\Big[\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}_{\textsf{OPT}_P}(\mathbb{B}_{\mathfrak{c}}\cup\mathbb{Q}_{\mathfrak{c}})+\tfrac{8f}{\lambda}\Big]\\ &\le 12\lambda(\log\Delta+2)\cdot\textsf{OPT}_P+\frac{8f|\mathbf{\Lambda}|}{\lambda}\ ,\end{aligned}$$

where the first inequality follows from Lemma 2.2 and the second one follows from Lemma 2.3. Then, we observe that since $\lambda=\frac{\varepsilon}{12(1+3/\varrho)(\log\Delta+2)}$, we obtain $12\,\lambda\,(\log\Delta+2)\cdot\textsf{OPT}_P\le\frac{\varepsilon}{1+3/\varrho}\cdot\textsf{OPT}_P$.

Further, $|\mathbf{\Lambda}|\cdot\frac{4f}{\lambda}\le\frac{\varepsilon}{1+3/\varrho}\cdot\textsf{OPT}_P$ from our analysis above (inequality (2.2)), and hence $\frac{8f|\mathbf{\Lambda}|}{\lambda}\le\frac{2\cdot\varepsilon}{1+3/\varrho}\cdot\textsf{OPT}_P$. This yields

$$\mathbf{E}\Big[\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{B}_{\mathfrak{c}})\Big]\le\frac{3}{1+3/\varrho}\cdot\varepsilon\cdot\textsf{OPT}_P\ .$$

Therefore, Markov inequality implies that with probability at least $1-\varrho$ holds $\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{B}_{\mathfrak{c}})\le\frac{1}{\varrho}\cdot\frac{3}{1+3/\varrho}\cdot\varepsilon\cdot\textsf{OPT}_P$ i.e.,

$$\sum_{\mathfrak{c}\in\mathbf{\Lambda}}\textsf{COST}(\mathbb{B}_{\mathfrak{c}})\le\frac{3}{\varrho+3}\cdot\varepsilon\cdot\textsf{OPT}_P\ . \tag{2.3}$$

Now the proof of part (2) of the Partition Theorem 2.1 follows directly from inequalities (2.2) and (2.3), and from the fact that for every cell $\mathfrak{c}$ we have

$$\mathsf{COST}(\mathfrak{c}) \le \mathsf{COST}(\mathbb{I}_{\mathfrak{c}}) + \mathsf{COST}(\mathbb{B}_{\mathfrak{c}}) \ .$$

If we sum up these bounds, we obtain that with probability at least $1 - \varrho$ the following holds

$$\begin{aligned}\sum_{\mathfrak{c}\in\mathbf{\Lambda}} \mathsf{COST}(\mathfrak{c}) &\le \sum_{\mathfrak{c}\in\mathbf{\Lambda}} \mathsf{COST}(\mathbb{I}_{\mathfrak{c}}) + \sum_{\mathfrak{c}\in\mathbf{\Lambda}} \mathsf{COST}(\mathbb{B}_{\mathfrak{c}}) \\ &\le (1 + \tfrac{\varepsilon}{1+3/\varrho}) \cdot \mathsf{OPT}_P + \frac{3}{\varrho+3}\varepsilon \cdot \mathsf{OPT}_P \\ &= (1+\varepsilon)\cdot \mathsf{OPT}_P \ .\end{aligned}$$

□

## 3 Fast $(1+\varepsilon)$-approximation for facility location.

With Theorem 2.1 at hand, we can design our fast $(1+\varepsilon)$-approximation for the facility location problem and show that it runs in $\mathcal{O}(n \log \Delta \log n \log\log n)$ time (assuming that $\varepsilon$ is an arbitrary constant).

In this section, to define heavy and light cells in the partitioning scheme, we will use as $\mathbb{A}$ a deterministic $\alpha$-factor approximation algorithm for the facility location problem as developed later in Section 4, in Theorem 4.1. This algorithm runs in time $\mathcal{O}(n \log n \log\log n)$ and ensures that $\alpha = \mathcal{O}(1)$.

**PTAS ($P$)**

- Find a partition $\mathbf{\Lambda}$ of $[2\Delta]^2$ as promised in Theorem 2.1, using as $\mathbb{A}$ the algorithm from Section 4.
- For each cell $\mathfrak{c} \in \mathbf{\Lambda}$: Find a $(1+\varepsilon)$-approximation for the facility location problem for $\mathfrak{c}$.
- Return the union of the obtained solutions.

From now on, let $|P| = n$. We have three simple claims that describe properties of the algorithm.

CLAIM 3.1. *Partition $\mathbf{\Lambda}$ of $[2\Delta]^2$ can be found in $\mathcal{O}(n \log \Delta \log n \log\log n)$ time.*

*Proof.* To find the partition $\mathbf{\Lambda}$, we have to go through the nested grid $G_0, G_1, \ldots, G_{\log\Delta+1}$ and check if the appropriated cells are heavy or not. This requires calls to the algorithm $\mathbb{A}$, where in each grid, we have to consider some set of cells with the total number of points being at most $n$. Hence, the total running time is $(\log\Delta + 2) \times \mathcal{O}(n \log n \log\log n)$, by Theorem 4.1.

Our next claim follows directly from an earlier PTAS for facility location due to Kolliopoulos and Rao [18]. (Let us recall that for any cell $\mathfrak{c}$, the number of points from $P$ inside $\mathfrak{c}$ is denoted by $n_{\mathfrak{c}}$.)

CLAIM 3.2. *For any cell $\mathfrak{c} \in \mathbf{\Lambda}$ one can find a $(1+\varepsilon)$-approximation for facility location for $\mathfrak{c}$ in time*

$$\mathcal{O}(n_{\mathfrak{c}} \log^8 n_{\mathfrak{c}} \log n 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)}) \ ;$$

*the algorithm is randomized and it errs with probability at most $1/n^2$.*

*Proof.* The claims follows directly from the result due to Kolliopoulos and Rao [18] for the facility location problem that gives a randomized algorithm that "with high probability" finds a $(1+\varepsilon)$-approximation for the facility location problem for $\mathfrak{c}$ in time $\mathcal{O}(n_{\mathfrak{c}} \cdot \log^8 n_{\mathfrak{c}} \cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)})$. However, since the error probability is proportional to the size of the input $n_{\mathfrak{c}}$, we have to repeat this algorithm $\mathcal{O}(\log n)$ times and then choose the best solution, to ensure that the error probability is at most $1/n^2$.

Our next claim follows from (1) in Theorem 2.1 and from the runtime bounds for the $k$-median problem.

CLAIM 3.3. *Let $\mathfrak{c} \in \mathbf{\Lambda}$ and suppose that $\mathsf{COST}(\mathfrak{c}) \le r \cdot f$ for some $r \ge 2$. Then, one can find a $(1+\varepsilon)$-approximation for the facility location problem for $\mathfrak{c}$ in time*

$$\mathcal{O}(n_{\mathfrak{c}} \log n \log_{1+\varepsilon} r) + 2^{\mathcal{O}((1+\log(1/\varepsilon))/\varepsilon)}(r \log n)^{\mathcal{O}(1)} \ ;$$

*the algorithm is randomized and it errs with probability at most $1/n^2$.*

*Proof.* Let us observe that our assumption about $\mathsf{COST}(\mathfrak{c})$ ensures that there is always an optimal solution to the facility location problem for $\mathfrak{c}$ that uses at most $r$ facilities.

Our algorithm reduces the facility location problem to a sequence of $k$-median problems. Let $\mathfrak{D}$ be the set of all integers of the form $\lfloor(1+\varepsilon)^i\rfloor$ or $\lceil(1+\varepsilon)^i\rceil$, for $i = 0, 1, \ldots, \lceil\log_{1+\varepsilon} r\rceil$. (Notice that $|\mathfrak{D}| = \mathcal{O}(\log_{1+\varepsilon} r)$.)

We first find $(1+\varepsilon)$-approximations for the $k$-median problem for the values $k \in \mathfrak{D}$. For every $k \in \mathfrak{D}$, let $Q_k$ be the set of $k$ medians for the obtained approximation for the $k$-median problem; its cost is equal to $\sum_{p\in P\cap\mathfrak{c}} d(p, Q_k)$. Notice that if $\mathrm{med}_k(\mathfrak{c})$ is the optimal cost of the $k$-median for $\mathfrak{c}$, ($\mathrm{med}_k(\mathfrak{c}) = \min_{X_k\subseteq\mathbb{R}^2:|X_k|=k} \sum_{p\in P\cap\mathfrak{c}} d(p, X_k)$), then

$$\sum_{p\in P\cap\mathfrak{c}} d(p, Q_k) \le (1+\varepsilon)\cdot \mathrm{med}_k(\mathfrak{c}) \ .$$

We find $k^*$ that minimizes $\sum_{p\in P\cap\mathfrak{c}} d(p,Q_k)+k\cdot f$ and return $Q_{k^*}$ as a $(1+\varepsilon)$-approximation for the facility location problem for $\mathfrak{c}$.

Let us first show that if for every $k\in\mathfrak{D}$, $Q_k$ is a $(1+\varepsilon)$-approximation for the $k$-median problem for $\mathfrak{c}$, then the obtained solution $Q_{k^*}$ is a $(1+\varepsilon)$-approximation for the facility location problem for $\mathfrak{c}$. Let $F_{\mathsf{OPT}_P}(\mathfrak{c})$ be an optimal set of facilities for the cell $\mathfrak{c}$, that is,

$$\mathsf{COST}(\mathfrak{c}) = \sum_{p\in P\cap\mathfrak{c}} d(p,F_{\mathsf{OPT}_P}(\mathfrak{c})) + k_\mathfrak{c}\cdot f \ ,$$

with $k_\mathfrak{c}=|F_{\mathsf{OPT}_P}(\mathfrak{c})|$.

Observe that $\mathrm{med}_{k_\mathfrak{c}}(\mathfrak{c}) = \sum_{p\in P\cap\mathfrak{c}} d(p,F_{\mathsf{OPT}_P}(\mathfrak{c}))$. Let $k^+$ be the smallest integer in $\mathfrak{D}$ that is greater than or equal to $k_\mathfrak{c}$. We claim that

$$\sum_{p\in P\cap\mathfrak{c}} d(p,Q_{k^+}) + k^+\cdot f \le (1+\varepsilon)\cdot\mathsf{COST}(\mathfrak{c}),$$

what would yield the claim.

Let us first observe that since the solutions to the $k$-median problem are non-increasing with $k$, $\mathrm{med}_{k_\mathfrak{c}}(\mathfrak{c})\ge \mathrm{med}_{k^+}(\mathfrak{c})$, and hence

$$\sum_{p\in P\cap\mathfrak{c}} d(p,Q_{k^+}) \le (1+\varepsilon)\sum_{p\in P\cap\mathfrak{c}} d(p,F_{\mathsf{OPT}_P}(\mathfrak{c})) \ .$$

Next, we note that our construction of $\mathfrak{D}$ implies that $k^+\le(1+\varepsilon)k_\mathfrak{c}$. Therefore, if we combine these two claims, then we obtain:

$$\begin{aligned}
&\sum_{p\in P\cap\mathfrak{c}} d(p,Q_{k^+}) + k^+\cdot f\\
\le\ &(1+\varepsilon)\sum_{p\in P\cap\mathfrak{c}} d(p,F_{\mathsf{OPT}_P}(\mathfrak{c})) + (1+\varepsilon)\cdot k_\mathfrak{c}\cdot f\\
=\ &(1+\varepsilon)\cdot\mathsf{COST}(\mathfrak{c}) \ .
\end{aligned}$$

Since

$$\sum_{p\in P\cap\mathfrak{c}} d(p,Q_{k^*}) + k^*\cdot f \le \sum_{p\in P\cap\mathfrak{c}} d(p,Q_{k^+}) + k^+\cdot f \ ,$$

this implies that $Q_{k^*}$ is a $(1+\varepsilon)$-approximation for the facility location problem for $\mathfrak{c}$.

Now, we analyze the running time of our algorithm. We run $|\mathfrak{D}|$ times a $(1+\varepsilon)$-approximation algorithm for the $k$-median problem for an instance with $n_\mathfrak{c}$ points, $n_\mathfrak{c}=|P\cap\mathfrak{c}|$. Har-Peled and Mazumdar [15] show how this problem can be solved in time $\mathcal{O}(n_\mathfrak{c}+2^{\mathcal{O}((1+\log(1/\varepsilon))/\varepsilon)}k^{\mathcal{O}(1)}\log^{\mathcal{O}(1)} n_\mathfrak{c})$. However, to achieve the required error probability at most $1/n^2$, we will repeat the algorithm by Har-Peled and Mazumdar [15] $\mathcal{O}(\log n)$ number of times, and we will choose the best solution. Since in our case $k\le r$, the total running time of the algorithm is bounded by

$$\mathcal{O}((n_\mathfrak{c}+2^{\mathcal{O}((1+\log(1/\varepsilon))/\varepsilon)}(r\log n)^{\mathcal{O}(1)})\cdot\log n)\times|\mathfrak{D}| \ ,$$

as required.

With Claims 3.1–3.3 at hand, we area ready to prove our main theorem.

THEOREM 3.1. *Algorithm PTAS($P$) with probability at least $\frac{2}{3}$ finds a $(1+\mathcal{O}(\varepsilon))$-approximation for the facility location of $P$ in time $\mathcal{O}(n\cdot\log n\cdot\log\Delta\cdot\log\log n + n\cdot\log n\cdot\log\log(n\Delta)\cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)})$.*

*Proof.* Theorem 2.1 and Claim 3.1 ensure that in time $\mathcal{O}(n\log\Delta\log n\log\log n)$ one can find a partition $\mathbf{\Lambda}$ such that with probability at least $\frac{3}{4}$ the three properties listed in Theorem 2.1 are satisfied.

Next, we consider the time needed to find feasible solutions for the facility location problem for all cells $\mathfrak{c}\in\mathbf{\Lambda}$, whose cost is (with probability at least $1-\frac{1}{n}$) upper bounded by $(1+\varepsilon)\cdot\mathsf{COST}(\mathfrak{c})$ for all $\mathfrak{c}\in\mathbf{\Lambda}$. We will use two alternative ways of doing this, using either Claim 3.2 or 3.3, depending on the size of $P\cap\mathfrak{c}$.

Let $c$ be a positive constant such that the running time of the algorithm in Claim 3.3 is

$$\mathcal{O}(n_\mathfrak{c}\cdot\log n\cdot\log_{1+\varepsilon} r) + 2^{\mathcal{O}((1+\log(1/\varepsilon))/\varepsilon)}(r\log n)^c \ .$$

Let

$$\zeta = 2^{\mathcal{O}((1+\log(1/\varepsilon))/\varepsilon)}\cdot(r\log n)^c \ .$$

We define $\mathbf{\Lambda}_S$ to be the subset of cells $\mathfrak{c}\in\mathbf{\Lambda}$ with $n_\mathfrak{c}\le\zeta$ and $\mathbf{\Lambda}_L$ as the subset of cells $\mathfrak{c}\in\mathbf{\Lambda}$ with $n_\mathfrak{c}>\zeta$ (notice that $\mathbf{\Lambda}_L=\mathbf{\Lambda}\setminus\mathbf{\Lambda}_S$).

For each cell $\mathfrak{c}\in\mathbf{\Lambda}_S$, we will find $(1+\varepsilon)$-approximate solutions for the the facility location problem using Claim 3.2, and for the cells $\mathfrak{c}\in\mathbf{\Lambda}_L$, we will find $(1+\varepsilon)$-approximate solutions for the the facility location problem using Claim 3.3

Since we perform at most $n$ calls to the randomized algorithms in Claims 3.2 and 3.3, the probability that we will find $(1+\varepsilon)$-approximate solutions for the the facility location problem for *all* cells in $\mathbf{\Lambda}$ is at least $1-1/n$.

Therefore, by Theorem 2.1, the total cost of the union of the solutions to these facility location problems is (with probability at least $\frac{3}{4}-\frac{1}{n}$) bounded by:

$$\begin{aligned}
\sum_{\mathfrak{c}\in\mathbf{\Lambda}}(1+\varepsilon)\cdot\mathsf{COST}(\mathfrak{c}) &\le (1+\varepsilon)^2\cdot\mathsf{OPT}_P\\
&\le (1+\mathcal{O}(\varepsilon))\cdot\mathsf{OPT}_P \ .
\end{aligned}$$

Now, we must analyze the running time needed to approximate the cost of facility location for all cells in $\mathbf{\Lambda}$.

The running time of approximating the cost of facility location for the cells in $\mathbf{\Lambda}_S$ is by Claim 3.2 upper bounded by the following:

$$\begin{aligned}
&\sum_{\mathfrak{c}\in\mathbf{\Lambda}_S} \mathcal{O}(n_\mathfrak{c} \cdot \log^8 n_\mathfrak{c} \cdot \log n) \cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)} \\
&\leq \sum_{\mathfrak{c}\in\mathbf{\Lambda}_S} \mathcal{O}(n_\mathfrak{c} \cdot \log^8 \zeta \cdot \log n) \cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)} \\
&\leq \mathcal{O}(n \cdot \log^8 \zeta \cdot \log n) \cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)} \ . \qquad (3.4)
\end{aligned}$$

The running time of approximating the cost of facility location for the cells in $\mathbf{\Lambda}_L$ is by Claim 3.3 upper bounded by

$$\sum_{\mathfrak{c}\in\mathbf{\Lambda}_L} \mathcal{O}(n_\mathfrak{c} \cdot \log n \cdot \log_{1+\varepsilon} r + \zeta) \ ,$$

with $r = \mathcal{O}((\log \Delta)/\varepsilon)^2)$.

Then, the definition of $\mathbf{\Lambda}_L$ implies that $|\mathbf{\Lambda}_L| \leq n/\zeta$ and hence, the running time is upper bounded by the following:

$$\begin{aligned}
&\sum_{\mathfrak{c}\in\mathbf{\Lambda}_L} \mathcal{O}\left(n_\mathfrak{c} \cdot \log n \cdot \log_{1+\varepsilon} r + \zeta\right) \\
&= \mathcal{O}(n \cdot \log n \cdot \log_{1+\varepsilon}(\log \Delta/\varepsilon)) + \mathcal{O}(|\mathbf{\Lambda}_L| \cdot \zeta) \\
&= \mathcal{O}(n \cdot \log n \cdot \log_{1+\varepsilon}(\log \Delta/\varepsilon)) \ . \qquad (3.5)
\end{aligned}$$

Therefore, inequalities (3.4) and (3.5) imply that the running time of approximating the cost of facility location for all cells in $\mathbf{\Lambda}$ using Claims 3.2 and 3.3 is upper bounded $\mathcal{O}(n \cdot \log^8 \zeta \cdot \log n) \cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)} + \mathcal{O}(n \cdot \log n \cdot \log_{1+\varepsilon}(\log \Delta/\varepsilon))$ which is asymptotically equal to

$$\mathcal{O}(n \cdot \log n \cdot \log\log(n\Delta)) \cdot 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)} \ ,$$

what completes the proof.

REMARK 3.1. *Theorem 3.1 has been presented in the framework convenient for our streaming algorithms and for some natural applications. However, one can extend Theorem 3.1 to the problem of facility location on the plane, where the input point set $P$ and the set of facilities are allowed to be at any location in $\mathbb{R}^2$ (not just in the grid $[\Delta]^2$); see, e.g., [18, Section 2], for more details of the reduction. The running time of the obtained $(1 + \mathcal{O}(\varepsilon))$-approximation algorithm is*

$$\mathcal{O}(n \log n \log\log n(\log n + 2^{\mathcal{O}(\log(1/\varepsilon)/\varepsilon)})) \ .$$

THEOREM 3.2. *For any constant $\varepsilon > 0$, there is an algorithm that for any set of $n$ points $P$ in $\mathbb{R}^2$ with probability at least $\frac{2}{3}$ finds a $(1 + \varepsilon)$-approximation for the facility location of $P$ in time $\mathcal{O}(n \log^2 n \log\log n)$.*

## 4 Fast constant-factor approximation algorithm for facility location.

Our PTAS in Section 3 requires a fast constant-factor approximation algorithm for the uniform facility location problem. In this section we present such a fast ($\mathcal{O}(n \log n \log\log n)$-time) constant-factor approximation algorithm. The algorithm is a simple modification of an algorithm by Mettu and Plaxton [22] and its extensions in [17, 9].

Let $P \subseteq \mathbb{R}^2$ be a set of $n$ points in the plane. In this section, we will assume, without loss of generality, that the input is scaled in such a way that $f = 1$. Hence, we have

$$\textsf{COST}(P, F) = |F| + \sum_{p\in P} \min_{q\in F} d(p, q) \ .$$

The algorithm requires the following definitions of the ($\ell_\infty$-)*radius* of a point. Let $Q(q, m)$ be the square centered at $q$ with side length $2m$ (i.e., $Q(q, m)$ is the $\ell_\infty$-ball with radius $m$). Given $q \in P$, the *radius* of $q$, denoted by $R(q)$, is the smallest value $m$ such that $\sum_{p\in P\cap Q(q,m)} \min \bigcup_{i=1,2}\{|p^{(i)} - q^{(i)} + m|, |q^{(i)} - p^{(i)} + m|\} = f$, where $p^{(i)}$ and $q^{(i)}$ refer to the $i$-th coordinate of $p$ and $q$, respectively. In other words, the radius of $q$ is equal to the sum of distances of the points inside the square $Q(q, m)$ to the boundary of $Q(q, m)$.

The *discrete radius* of $q \in P$, denoted by $r(q)$, is the smallest value $m = 2^i$, $i \in \mathbb{Z}$, such that there are at least $1/m$ points inside $Q(q, m)$. The definition of radius is an adaptation of a similar definition in [22]. It had been extended to $\ell_\infty$-balls in [9]. The definition of discrete radius is similar to a definition in [17].

We first prove that the discrete radius approximates the radius of a point up to a constant factor. (A similar lemma for balls in metric spaces is in [17]. Our lemma is a simple adaption to our setting.)

LEMMA 4.1. *Let $q \in P$ be an arbitrary point. Then, we have*

$$\frac{1}{2} \cdot R(q) \quad \leq r(q) < \quad 2 \cdot R(q) \ .$$

*Proof.* The first inequality follows from the fact that, inside a square of radius $2r(q)$, we have at least $1/r(q)$ points whose distance to the border is at least $r(q)$. Hence, $R(q) \leq 2r(q)$.

By the definition of the discrete radius, we know that there are less than $2/r(q)$ points inside the square with center $q$ and radius $r(q)/2$. Each of the points has a distance of at most $r(q)/2$ to the boundary of the square. Hence, $R(q) > r(q)/2$, which proves the second inequality.

Thus, the discrete radius is a 4-approximation of the radius of a point.

Next, we can state the algorithm.

```
FACILITYLOCATION($P$)
  • For each point $q \in P$ do
      – Compute the discrete radius $r(q)$ of $q$
  • Sort the discrete radii in non-decreasing order
  • For each point $q$ in this order do
      – If there is no open facility within distance
        at most $8r(q)$ then $F = F \cup q$
  • Return $F$
```

In the rest of this section, we first show that Algorithm FACILITYLOCATION($P$) computes a $\mathcal{O}(1)$-approximation and then we address how it can be implemented in $\mathcal{O}(n \log n \log\log n)$ time.

We observe that, for any $q_1, q_2 \in F$, we have that the squares $Q(q_1, 4r(q_1))$ and $Q(q_2, 4r(q_2))$ are disjoint. This together with Lemma 4.1 implies that the squares $Q(q_1, R(q_1))$ and $Q(q_2, R(q_2))$ are disjoint for any $q_1, q_2 \in F$.

We will show that, for any solution $F$ computed by our algorithm,

$$\text{COST}(P,F) = |F| + \sum_{p\in P} \min_{q\in F} d(p,q) \le 17 \sum_{p\in P} R(p) \ .$$

Then, we will prove that the cost of an optimal solution is more than $\sum_{p\in P} R(p)/16$. This proves that the algorithm is a constant-approximation algorithm. Both proofs are similar to earlier proofs in [17] and based on the work in [22].

LEMMA 4.2. *Let $F$ be a set of facilities computed by Algorithm* FACILITYLOCATION *on input $P$. Then, we have*

$$\textit{COST}(P,F) \quad \le \quad 17 \sum_{p\in P} R(p) \ .$$

*Proof.* A simple consequence of our algorithm is that any point $p \in P$ has a distance of at most $8r(p)$ to the nearest open facility. Furthermore, by Lemma 4.1, we know that

$$r(P) \le 2\, R(p) \ .$$

Hence, any point $p \in P$ has connection cost at most $16\, R(p)$. To proceed, we need the following claim.

CLAIM 4.1. *Let $F$ be a set of facilities computed by algorithm* FACILITYLOCATION. *Then, for any $q \in F$, we have*

$$\sum_{p \in P \cap Q(q,R(q))} R(p) \quad \ge \quad 1 \ .$$

*Proof.* Let $p \in P$ be an arbitrary point inside $Q(q, R(q))$, and let $\ell$ be its distance to the boundary of $Q(q, R(q))$. Then, the square $Q(p, \ell)$ is completely contained in $Q(q, R(q))$. This implies that, for any point $r \in P \cap Q(p, \ell)$, the distance to the boundary of $Q(p, \ell)$ is at most as large as the distance to the boundary of $Q(q, R(q))$. This implies that $R(p) \ge \ell$. This together with the definition of $R(q)$ implies the claim.

We have already observed that, for any $q_1, q_2 \in F$, the squares $Q(q_1, R(q_1))$ and $Q(q_2, R(q_2))$ are disjoint. Now, the above claim implies that $|F| \le \sum_{p\in P} R(p)$. Hence, the lemma follows.

Our next step is to prove that the cost of an optimal solution $F_{\text{OPT}_P}$ is $\Omega(\sum_{p\in P} R(p))$.

LEMMA 4.3. *Let $F_{\text{OPT}_P}$ be an optimal solution. Then, we have*

$$\textit{COST}(P, F_{\text{OPT}_P}) > \sum_{p\in P} R(p)/16 \ .$$

*Proof.* We divide $P$ into two subsets $P_A$ and $P_B$. The subset $P_A$ contains all points $p$ whose distance to $F_{\text{OPT}_P}$ is at least $r(p)/8$. The subset $P_B$ contains the remaining points. For a facility $q \in F_{\text{OPT}_P}$, let $P_B(q)$ be the subset of points in $P_B$ for which $q$ is the closest facility in $F_{\text{OPT}_P}$, breaking ties arbitrarily. We need the following claim.

CLAIM 4.2. *Let $q \in F_{\text{OPT}_P}$ be an arbitrary facility, and let $p \in P_B(q)$. Then, for all $t \in P_B(q)$, we have*

$$d(p,t) < r(p)/2 \ .$$

*Proof.* By the triangle inequality, we have $d(p,t) \le d(p,q) + d(q,t)$. From the definition of $P_B(q)$ and $p \in P_B(q)$, we know that $d(q,p) < r(p)/8$. Thus, the claim follows if we can prove that $d(q,t) \le 3r(p)/8$.

Assume, for the purpose of contradiction, that $d(q,t) > 3r(p)/8$. This implies $r(t) > 3r(p)$. Since $r(t)$ is a power of 2, we know that $r(t) \ge 4r(p)$ and so $r(t)/4 \ge r(p)$. By the definition of $P_B(q)$ and $t \in P_B(q)$, we know that $d(t,q) < r(t)/8$. Due to triangle inequality and the above arguments, we obtain

$$d(t,p) + r(p) \le d(t,q) + d(q,p) + r(p) < r(t)/2 \ .$$

Hence, the square $Q(t, r(t)/2)$ contains the square $Q(p, r(p))$. This is a contradiction, because then the radius of $t$ would be at most $r(t)/2$.

Let $t$ be the point in $P_B(q)$ with biggest discrete radius, i.e., $r(p) \le r(t)$ for any $p \in P_B(q)$. By the definition of the discrete radius, we know that

$$|Q(t, r(t)/2)| < 2/r(t) \ .$$

Furthermore, due to Claim 4.2, all the points in $P_B(q)$ are contained in $Q(t, r(t)/2)$. This implies that

$$\begin{aligned} 2 &> |Q(t,r(t)/2)| \cdot r(t) \\ &\ge \sum_{p \in P_B(q)} r(t) \\ &\ge \sum_{p \in P_B(q)} r(p) \\ &\ge \sum_{p \in P_B(q)} \frac{R(p)}{2} \ , \end{aligned}$$

where the last inequality follows from Lemma 4.1. Hence, we get,

$$|F_{\mathsf{OPT}_P}| > \sum_{q \in F_{\mathsf{OPT}_P}} \sum_{p \in P_B(q)} \frac{R(p)}{4} \ .$$

Since

$$\sum_{p \in P_A} \min_{q \in F_{\mathsf{OPT}_P}} d(p,q) \ \ge \ \sum_{p \in P_A} \frac{R(p)}{16} \ ,$$

we obtain the following inequality,

$$\begin{aligned} \mathsf{COST}(P, F_{\mathsf{OPT}_P}) &= |F_{\mathsf{OPT}_P}| + \sum_{p \in P} \min_{q \in F_{\mathsf{OPT}_P}} d(p,q) \\ &> \sum_{p \in P_B} \frac{R(p)}{4} + \sum_{p \in P_A} \frac{R(p)}{16} \\ &\ge \sum_{p \in P} \frac{R(p)}{16} \ , \end{aligned}$$

which completes the proof of Lemma 4.3.

We conclude by Lemmas 4.1, 4.2, and 4.3 that Algorithm FACILITYLOCATION yields a constant-approximation.

It remains to discuss how to implement the algorithm. For this purpose, we build a $2D$ layered range tree for the input point set $P$. Such a range tree can be used to answer orthogonal range counting queries in $\mathcal{O}(\log n)$ time, and it has a construction time of $\mathcal{O}(n \log n)$.

Now, observe that, for any $p \in P$, we have

$$1/2^{\lfloor \log n \rfloor} \le r(p) \le 1 \ .$$

Thus, we use binary search to find the right value of $r(p)$ in $\mathcal{O}(\log \log n)$ queries to the range tree. Hence, computing the discrete radii of all the points in $P$ requires $\mathcal{O}(n \log n \log \log n)$ time. We can sort the radii in non-decreasing order in $\mathcal{O}(n \log n)$ time.

Next, we partition $P$ into subsets $P_0, \ldots, P_{\lfloor \log n \rfloor}$, where $P_i$ is the set of points whose radius is $1/2^i$. For each $P_i$, we construct a layered range tree. Then, we start with a point $q \in P$ with smallest radius. We insert $q$ into our set of facilities $F$ and use the range trees to identify all points $p$ with $r(p) \ge r(q)$ that contain $q$ inside $Q(p, 8r(p))$. We delete these points from their corresponding range tree.

Note that, for each facility $q$ in $P_i$ that we put into $F$, we know that at least all the points in $Q(q, r(q))$ will be deleted. Since, for any $q' \in F \backslash \{q\}$ the squares $Q(q, r(q))$ and $Q(q', r(q'))$ do not overlap and the number of points in $Q(q, r(q))$ is at least $2^i$, there are at most $n/2^i$ facilities in $F \cap P_i$.

Furthermore, inserting $q \in P_i$ in $F$ leads to delete operations in $i+1$ range trees. Since deleting in a layered range tree can be done in $\mathcal{O}(k \log n)$ time by deleting the point from the second level data structures (we may keep the point in the first level tree as we just need $O((k+1)\log n)$ running time, where $n$ is the number of input points rather than the number of points in the tree), where $k$ is the number of deleted points, and there are at most $n$ points to delete in total, the running time to compute $F$ is upper bounded by

$$\mathcal{O}(n \log n) + \sum_{i=0}^{\lfloor \log n \rfloor} \frac{n}{2^i}(i+1) \cdot \mathcal{O}(\log n) = \mathcal{O}(n \log n) \ .$$

Thus, the total running time of the algorithm is $\mathcal{O}(n \log n \log \log n)$.

Therefore, we can conclude the discussion in this section with the following theorem.

THEOREM 4.1. *Given a set $P \subseteq \mathbb{R}^2$ of $n$ points in plane, Algorithm* FACILITYLOCATION *computes a constant-factor approximation of the uniform facility location problem for $P$ in $\mathcal{O}(n \log n \log \log n)$ time.*

## 5 A streaming algorithm.

In this section, we derive a $(1+\varepsilon)$-approximation algorithm in the dynamic geometric streaming setting, where the items of the stream are insertions and deletions of points from $[\Delta]^2 = \{1, \ldots, \Delta\}^2$, and where $\varepsilon > 0$ is an arbitrary fixed real.

Let us fix a failure probability bound $\varrho$, $0 < \varrho < 1$. Let $\aleph^*_{\varrho/3}$ be the minimal value for $\aleph$ for which Partition Theorem 2.1 holds with the failure probability $\varrho/3$; note that

$$\aleph^* = \Theta\left(\left(\frac{\log \Delta + 2}{\varepsilon}\right)^2\right) \ .$$

Let $\boldsymbol{\kappa} = 2 \cdot \aleph^* \cdot f$ (and thus, Partition Theorem 2.1 will hold with the probability at least $1 - \varrho/3$ for any choice of the threshold defining heavy and light cells that is greater than or equal to $\boldsymbol{\kappa}/2$).

Our streaming algorithm will use the partitioning scheme as presented in Section 2, and in particular, Partition Theorem 2.1. For reasons that will become clear later, to distinguish between light and heavy cells we define a *random threshold* $T$ as $T = \boldsymbol{\kappa} \cdot \mathfrak{rand}$, where $\mathfrak{rand}$ is chosen uniformly at random from the interval $[1, 2]$.

We use the threshold $T$ to distinguish between *heavy* and *light* cells in the randomly shifted nested grid partition $G_0, G_1, \dots, G_{\log\Delta+1}$ (cf. Section 3). Any cell $\mathfrak{c}$ with $\mathsf{COST}(\mathfrak{c}) \geq T$ will be called *heavy* and any cell with $\mathsf{COST}(\mathfrak{c}) < T$ will be called *light*.[4] A light cell whose parent cell is heavy will be called a *maximal light* cell.

**5.1 Focusing on $\delta$-detectable cells.** For a parameter $\delta$, a maximal light cell $\mathfrak{c}$ with parent cell $\mathfrak{c}^*$ is called *$\delta$-detectable* if

$$\mathsf{COST}(\mathfrak{c}) \leq (1-\delta)T \text{ and } (1+\delta)T \leq \mathsf{COST}(\mathfrak{c}^*) \ .$$

Our first lemma proves that for sufficiently small $\delta$, with constant probability the vast majority of the maximal light cells are $\delta$-detectable and as the result, we can "ignore" the cost of maximal light cells that are not $\delta$-detectable.

LEMMA 5.1. *Let $T$ be chosen as stated above and let $\varrho$ be an arbitrary positive constant. If $\delta = c\varepsilon/\log\Delta$, for an appropriate constant $c$, then with probability at least $1-\varrho$ the following inequalities hold:*

$$(1-\varepsilon)\cdot \mathit{OPT} \leq \sum_{i=0}^{\log\Delta+1} \sum_{\delta\text{-detectable cell } \mathfrak{c}\in G_i} \mathit{COST}(\mathfrak{c})$$

*and*

$$\sum_{i=0}^{\log\Delta+1} \sum_{\delta\text{-detectable cell } \mathfrak{c}\in G_i} \mathit{COST}(\mathfrak{c}) \ \leq (1+\varepsilon)\cdot \mathit{OPT} \ .$$

*Proof.* The upper bound follows from the Partition Theorem 2.1 (where will set the failure probability to be $\varrho/3$) that ensures with probability at least $1-\varrho/3$ we have

$$\sum_{\text{maximal light cells } \mathfrak{c}\in\bigcup_i G_i} \mathsf{COST}(\mathfrak{c}) \leq (1+\varepsilon)\cdot \mathsf{OPT} \ .$$

Hence, with probability at least $1-\varrho/3$ the following holds:

$$\begin{aligned} &\sum_{i=0}^{\log\Delta+1} \sum_{\delta\text{-detectable cell } \mathfrak{c}\in G_i} \mathsf{COST}(\mathfrak{c}) \\ \leq\ & \sum_{\text{maximal light cells } \mathfrak{c}\in\bigcup_{i=0}^{\log\Delta+1} G_i} \mathsf{COST}(\mathfrak{c}) \\ \leq\ & (1+\varepsilon)\cdot \mathsf{OPT} \ . \end{aligned}$$

Now we proceed to the lower bound. We begin with basic definitions.

- We define $Q$ to be the set of cells $\mathfrak{c}$ in $G_0, \dots, G_{\log\Delta+1}$ that would be maximal light if the threshold was chosen as $\boldsymbol{\kappa}/2$. (That is,

  $$\mathsf{COST}(\mathfrak{c}) \leq \boldsymbol{\kappa}/2 \text{ and } \mathsf{COST}(\mathfrak{c}^*) > \boldsymbol{\kappa}/2 \ ,$$

  where $\mathfrak{c}^*$ is the parent cell of $\mathfrak{c}$.)
- Let $MQ$ be the set of all cells set of cells $\mathfrak{c}$ in $G_0, \dots, G_{\log\Delta+1}$ such that $\mathsf{COST}(\mathfrak{c}) \leq 2\boldsymbol{\kappa}$ and that either $\mathfrak{c}\in Q$ or $\mathfrak{c}$ is an ancestor of a cell in $Q$.

We observe that only the cells in $MQ$ can be maximal light (when the threshold $T = \boldsymbol{\kappa}\cdot\mathfrak{rand}$ is chosen, as above). Indeed, firstly, any cell $\mathfrak{c}$ that is maximal light with the threshold $T$ must be light, and hence satisfy $\mathsf{COST}(\mathfrak{c}) \leq T \leq 2\boldsymbol{\kappa}$, and secondly, any such cell $\mathfrak{c}$ is either in $Q$, or is an ancestor of a cell in $Q$.

Next, we have the following claim.

CLAIM 5.1. *With probability at least $1-\varrho/3$ we have that*

$$\sum_{\mathfrak{c}\in MQ} \mathit{COST}(\mathfrak{c}) = \mathcal{O}(\log\Delta)\cdot \mathit{OPT} \ .$$

*Proof.* Let $BQ$ be the set of cells in $MQ$ that are not in $Q$ and that have no descendant in $BQ$. Observe that every cell $\mathfrak{c}$ in $BQ$ has $\boldsymbol{\kappa}/2 < \mathsf{COST}(\mathfrak{c}) \leq 2\boldsymbol{\kappa}$, and since no descendant of $\mathfrak{c}$ has cost greater than $\boldsymbol{\kappa}/2$, we conclude that all four children of $\mathfrak{c}$ are in $Q$.

Therefore, since the cost of any cell is upper bounded by the sum of the costs of of its children, we have

$$\sum_{\mathfrak{c}\in BQ} \mathsf{COST}(cc) \leq 4\cdot \sum_{\mathfrak{c}\in Q} \mathsf{COST}(\mathfrak{c}) \ .$$

Next, we observe that every cell $\mathfrak{c}$ in $MQ\setminus Q$ is either in $BQ$ or has a descendant in $BQ$. Also, every cell has at most $\log\Delta+1$ ancestors and every cell $\mathfrak{c}\in MQ\setminus Q$ has $\boldsymbol{\kappa}/2 < \mathsf{COST}(\mathfrak{c}) \leq 2\boldsymbol{\kappa}$.

[4]Let us notice that the definitions of heavy and light cells in this section are a small modification of the definitions from Section 3. The difference is twofold: firstly, the definition in Section 3 explicitly uses an approximation algorithm $\mathbb{A}$, whereas here we have an exact bound for the cost of a cell; secondly, we use a slightly different threshold value. We notice however that our choice of $\boldsymbol{\kappa}$ ensures that $\boldsymbol{\kappa} = \Theta(\aleph\cdot f)$, and hence, it allow us to warrant that the Partition Theorem 2.1 is satisfied.

Thus, we can conclude that

$$\sum_{\mathfrak{c}\in MQ\setminus Q} \mathsf{COST}(cc) \leq \sum_{\mathfrak{c}\in BQ} 4\cdot(\log\Delta+1)\cdot\mathsf{COST}(cc) \leq 16\cdot(\log\Delta+1)\cdot\sum_{\mathfrak{c}\in Q}\mathsf{COST}(\mathfrak{c}) \ .$$

Hence,

$$\sum_{\mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) = \sum_{\mathfrak{c}\in MQ\setminus Q}\mathsf{COST}(cc)+\sum_{\mathfrak{c}\in Q}\mathsf{COST}(cc) \leq (17+16\cdot\log\Delta)\cdot\sum_{\mathfrak{c}\in Q}\mathsf{COST}(\mathfrak{c}) \ .$$

Now, to conclude the proof of Claim 5.1, we observe that by taking maximal light cells for the threshold $T=\boldsymbol{\kappa}/2$, Theorem 2.1 ensures that with probability at least $1-\varrho/3$ we get

$$\sum_{\mathfrak{c}\in Q}\mathsf{COST}(\mathfrak{c}) \leq (1+\varepsilon)\cdot\mathsf{OPT} \ .$$

We will use Claim 5.1 to conclude the proof of the lower bound in Lemma 5.1. Let us first assume that $\sum_{\mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) = \mathcal{O}(\log\Delta\cdot\mathsf{OPT})$. As observed above, every cell $\mathfrak{c}$ that is maximal light and that is not $\delta$-detectable must be in $MQ$. Notice that the definition of the set $MQ$ has been done independently of the choice of $\mathfrak{rand}$ in the definition of the threshold $T$, but the choice of $\mathfrak{rand}$ is essential for being $\delta$-detectable.

We observe then if a cell $\mathfrak{c}$ with parent cell $\mathfrak{c}^*$ is not $\delta$-detectable, then we must have $(1-\delta)T < \mathsf{COST}(\mathfrak{c})$ or $\mathsf{COST}(\mathfrak{c}^*) < (1+\delta)T$, and hence

$$\min\{|T-\mathsf{COST}(\mathfrak{c})|, |T-\mathsf{COST}(\mathfrak{c}^*)|\} \leq \delta T \ .$$

Next, we characterize the values $|T-\mathsf{COST}(\mathfrak{c})|$ and $|T-\mathsf{COST}(\mathfrak{c}^*)|$ with respect to the random choice of $\mathfrak{rand}$ that defines $T=\boldsymbol{\kappa}\cdot\mathfrak{rand}$. Since we choose $\mathfrak{rand}$ uniformly at random from $[1,2]$, the probability that $|T-\mathsf{COST}(\mathfrak{c})|\leq\delta T$ is at most $2\delta$, and so is the probability that $|T-\mathsf{COST}(\mathfrak{c}^*)|\leq\delta T$.

Thus, at most an $(4\delta)$-fraction of cells in $MQ$ are not $\delta$-detectable. Hence, by Markov inequality and by our assumption that

$$\sum_{\mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) = \mathcal{O}(\log\Delta\cdot\mathsf{OPT}) \ ,$$

there is a constant $c>0$ such that for every $t>0$,

$$\mathbf{Pr}\Big[\sum_{\text{not } \delta\text{-detectable cell } \mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) \geq t\Big] \leq \frac{4\delta\sum_{\mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c})}{t} \leq \frac{c\delta\cdot\log\Delta\cdot\mathsf{OPT}}{t} \ .$$

Hence, for a given $\varrho>0$, if we set $\delta=\frac{\varepsilon\varrho}{3c\log\Delta}$, then (by taking $t=\varepsilon\cdot\mathsf{OPT}$ in the bound above) we obtain that with probability at least $1-\varrho/3$ it holds:

$$\sum_{\text{not } \delta\text{-detectable cell } \mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) \leq \varepsilon\cdot\mathsf{OPT} \ .$$

This implies that condition on $\sum_{\mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) = \mathcal{O}(\log\Delta\cdot\mathsf{OPT})$, we have with probability at least $1-\varrho/3$ the following bound:

$$\sum_{\delta\text{-detectable cell } \mathfrak{c}\in\bigcup_{i=0}^{\log\Delta+1}G_i}\mathsf{COST}(\mathfrak{c}) \geq \mathsf{OPT} - \sum_{\text{not } \delta\text{-detectable cell } \mathfrak{c}\in MQ}\mathsf{COST}(\mathfrak{c}) \geq (1-\varepsilon)\cdot\mathsf{OPT} \ ,$$

where the first inequality follows from the fact that

$$\sum_{\mathfrak{c} \text{ is maximal light}}\mathsf{COST}(\mathfrak{c}) \geq \mathsf{OPT}$$

for every choice of $T$.

Now, we can conclude the analysis of the lower bound by combing this bound with Claim 5.1, to ensure that

$$\sum_{\mathfrak{c} \text{ is } \delta\text{-detectable cell}}\mathsf{COST}(\mathfrak{c}) \geq (1-\varepsilon)\cdot\mathsf{OPT}$$

with probability at least $1-\frac{2}{3}\varrho$.

In summary, we have shown that

$$\sum_{\mathfrak{c} \text{ is } \delta\text{-detectable cell}}\mathsf{COST}(\mathfrak{c}) \leq (1+\varepsilon)\cdot\mathsf{OPT}$$

with probability at least $1-\varrho/3$, and that

$$\sum_{\mathfrak{c} \text{ is } \delta\text{-detectable cell}}\mathsf{COST}(\mathfrak{c}) \geq (1-\varepsilon)\cdot\mathsf{OPT}$$

with probability at least $1-\frac{2}{3}\varrho$. Thus, with probability at least $1-\varrho$, it holds that

$$(1-\varepsilon)\cdot\mathsf{OPT} \leq \sum_{i=0}^{\log\Delta+1}\ \sum_{\delta\text{-detectable cell } \mathfrak{c}\in G_i}\mathsf{COST}(\mathfrak{c})$$

and

$$\sum_{i=0}^{\log\Delta+1}\ \sum_{\delta\text{-detectable cell } \mathfrak{c}\in G_i}\mathsf{COST}(\mathfrak{c}) \leq (1+\varepsilon)\cdot\mathsf{OPT} \ ,$$

what completes the proof.

**5.2 Sampling algorithm in streaming setting.** Now, we will implement a variant of the following sampling algorithm in the streaming setting. The algorithm receives a point set $P$ and a sample size $s$ as input.

```
FL-SAMPLING $(P, s)$
  • Compute a partition of the input space into light cells as in the previous section
  • Let Apx be a $\mathcal{O}(1)$-approximation to the facility location cost of $P$
  • For $i = 0$ **to** $\log\Delta + 1$ do
      – Sample each cell from grid $G_i$ with probability $\frac{s}{\mathrm{Apx}}$ and let $\mathfrak{S}_\mathfrak{i}$ be the resulting set
      – Let $F_i = 0$
      – For each $\delta$-detectable cell $\mathfrak{c} \in \mathfrak{S}_\mathfrak{i}$ do
          ∗ Determine a $(1+\varepsilon)$-approximation $C$ of the facility location cost of $P \cap \mathfrak{c}$
          ∗ Let $F_i = F_i + C$
  • Return $\frac{\mathrm{Apx}}{s} \cdot \sum_{i=0}^{\log\Delta+1} F_i$
```

We now discuss the necessary modifications that are required to implement the above algorithm in the streaming setting. Since we do not know Apx in advance we will use one instance for any guess of cost $2^i \cdot f$, $0 \le i \le 2\log\Delta$. This requires $\mathcal{O}(\log\Delta)$ such guesses.

We will also maintain an instance of the $\mathcal{O}(1)$-approximation algorithm for the cost of the facility location problem by Lammersen and Sohler [21]. Using the result of this algorithm, we can decide afterwards, which instance of our main algorithm we will use.

Our next change is that we determine $\mathfrak{S}_\mathfrak{i}$ to be the set of cells that are mapped to 1 by a pairwise independent hash function $h_i$ with range $1, \dots, r = \lceil \mathrm{Apx}/s \rceil$. Now that we have defined the set $\mathfrak{S}_\mathfrak{i}$ implicitly, a critical point is that $\mathfrak{S}_\mathfrak{i}$ can be very large and so we cannot store all points that are mapped to a cell in $\mathfrak{S}_\mathfrak{i}$.

What rescues us here is the fact that $\mathfrak{S}_\mathfrak{i}$ with high probability contains only few maximal light cells. In fact, the number of maximal light cells in grid $G_i$ is bounded by the number of heavy cells in grid $G_{i+1}$. Recall that the parent of a maximal light cell in grid $G_i$ is a heavy cell in grid $G_{i+1}$. And so, the number of heavy cells in grid $G_{i+1}$ is

$$\mathcal{O}(\mathsf{OPT}/T) = \mathcal{O}(\mathrm{Apx}/T) \ ,$$

where $\mathsf{OPT}_P$ is the optimal cost of $P$ and $T$ is the random threshold in above. Hence, the expected number of maximal light cells in $\mathfrak{S}_\mathfrak{i}$ is $\mathcal{O}(s/T)$.

This can be exploited by using a second stage pairwise independent hash function $h'_i$ that maps each cell from $\mathfrak{S}_\mathfrak{i}$ to a random value between 1 and $t = cs^2/(25\varepsilon^2 T^2)$ for a large enough constant $c$.

Whenever a point inside a cell $\mathfrak{c}$ in $\mathfrak{S}_\mathfrak{i}$ is encountered in the stream, we subtract the coordinates of the lower left corner of its cell to obtain a new point (vector) $p'$ and then we pass this point to a substream $h'_i(\mathfrak{c})$. For $t = cs^2/(25\varepsilon^2 T^2)$ we see that each substream will contain points from at most one maximal light cell and some "noise points" mapped from other cells to it, which will be sufficiently small to approximate the cost of the maximal light cell with factor $(1+\varepsilon)$.

The scheme just described is a small modification of an algorithm introduced by [24]. The idea to use random shifts in this context is from [25]. See also [26, 27].

It remains to explain how to detect whether a substream contains a $\delta$-detectable cell. To do that, we also determine the points inside the parent cell of each sample cell in $\mathfrak{S}_\mathfrak{i}$ and we also hash them to different substreams to ensure that with high probability no stream contains more than one heavy cell. We summarize the above discussion in algorithm STREAMINGSAMPLECELL.

```
STREAMINGSAMPLECELL $(i)$
  • Let $r = \lceil \mathrm{Apx}/s \rceil$ and $t = cs^2/(25\varepsilon^2 T^2)$
  • Choose pairwise independent hash functions
    $h_i : \{1, \dots, |G_i|\} \to \{1, \dots, r\}$ and
    $h'_i : \{1, \dots, |G_i|\} \to \{1, \dots, t\}$
  • For each point $p$ in the stream do
      – Determine cells $\mathfrak{c}_1 \in G_i$ and $\mathfrak{c}_2 \in G_{i+1}$ that contain $p$
      – If $h_i(\mathfrak{c}_1) = 1$ then [a]
          ∗ Let $p = p - c_1$, where $c_1$ is the lower left corner of $\mathfrak{c}_1$
          ∗ Let $p' = p - c_2$, where $c_2$ is the lower left corner of $\mathfrak{c}_2$
      – Insert $p$ into $(k,\varepsilon)$-$Coreset$ $\mathrm{MED}^{\text{light}}_{h'_i(\mathfrak{c}_1)}$
      – Insert $p'$ into $(k,\varepsilon)$-$Coreset$ $\mathrm{MED}^{\text{heavy}}_{h'_i(\mathfrak{c}_1)}$
```

[a]**CHRISTIANE:** Am I right that we also have to insert $p$ into the coreset if $\mathfrak{c}_2$ has any child $\mathfrak{c}_3 \neq \mathfrak{c}_1$ with $h_i(\mathfrak{c}_3) = 1$?

In algorithm STREAMINGSAMPLECELL we use $(k,\varepsilon)$-$Coreset$ data structure of [11] for the $k$-median problem maintained in the streaming fashion. The formulation of the $k$-median problem is close to the facility location problem.

Let $P$ be a (possibly weighted) point set in $[\Delta]^2$ in which $w_p$ is the weight of a point $p \in P$. In the $k$-median problem we would like to find a set $C \subseteq [\Delta]^2$ of $k$ centers such that

$$\mathsf{COST}(P, C) = \sum_{p \in P} \min_{c \in C} w_p \cdot d(p, c)$$

is minimized.

A weighted set $S \subseteq [\Delta]^2$ is called a $(k,\varepsilon)$-*Coreset* for $P$, if for any set $C \subseteq [\Delta]^2$ with $|C| \le k$ we have

$$|\mathsf{COST}(S, C) - \mathsf{COST}(P, C)| \le \varepsilon \cdot \mathsf{COST}(P, C) \ .$$

The $(k,\varepsilon)$-*Coreset* data structure of [11] is as follows.

THEOREM 5.1. *[11] Let $P \subseteq [\Delta]^2$ be a point set given in the dynamic geometric data stream model. Let $k \ge 1$, $\varepsilon > 0$, and $0 < \delta < 1$. There exists a randomized streaming algorithm that with probability $1-\delta$ maintains a $(k,\varepsilon)$-Coreset of $P$ using a space of $\mathcal{O}(\log^7 \Delta \cdot k^2 \cdot \varepsilon^{-8})$. This streaming algorithm processes an operation* INSERT$(p)$ *or* DELETE$(p)$ *in $\mathcal{O}(\log^4 \Delta \cdot \varepsilon^{-4})$ time. We can compute an $(1\pm\varepsilon)$-approximation from the coreset in time $\mathcal{O}(k^5 \log^9 \Delta + k^2 \log^5 \Delta \cdot 2^{\log(1/\varepsilon)/\varepsilon})$.*

We use the coreset data structure of Theorem 5.1 to compute a $(1+\varepsilon)$-approximation for the facility location problem for the points in each substream. In particular, for each choice of $k \in \{1, \ldots, \lceil T/f \rceil\}$ we maintain $(k,\varepsilon)$-*Coreset*s $\textsc{Med}^{\text{light}}_{h'_i(\mathfrak{c}_1)}$ and $\textsc{Med}^{\text{heavy}}_{h'_i(\mathfrak{c}_1)}$ of $\mathfrak{c}_1$ and $\mathfrak{c}_2$ respectively.

Next, we compute $(1\pm\varepsilon)$-approximation of the cost of the $k$-median problem from the coresets and add the term $kf$ to obtain the corresponding facility location cost for each choice of $k \in \{1, \ldots, \lceil T/f \rceil\}$. To obtain the cost of the facility location problem we finally choose the cost which is minimum of all facility location costs for all values of $k \in \{1, \ldots, \lceil T/f \rceil\}$ .

Our next step will be to define an algorithm to extract the approximate cost of the $\delta$-detectable cells in grid $G_i$. The final output value of our algorithm will be the sum of these values. The algorithm simply tests for each cell, whether it is $\delta$-detectable and sums up the cost of the cells for which the test outputs true. In the end, the value is scaled by $r$ as each cell is taken into $h_i^{-1}(1)$ with probability $1/r$.

Since the $k$-median data structures and also the hashing scheme introduces some error, the algorithm may also count some cells, which are not $\delta$-detectable. However, as we will prove below, it will with high probability only count cells that are maximal light and so it will count every $\delta$-detectable cell in $h_i^{-1}(1)$.

```
STREAMINGREPORT $(i)$
• Let $F = 0$
• For $j = 1$ to $t$ do
    – Let $F_1$ be the value reported by $\textsc{Med}^{\text{light}}_j$
    – Let $F_2$ be the value reported by $\textsc{Med}^{\text{heavy}}_j$
    – If $F_1 \le (1-\varepsilon) \cdot T$ and $F_2 \ge (1+\varepsilon) \cdot T$
      then $F = F + F_1$
• Return $r \cdot F$
```

Next, we will analyze how the substreams of $\mathfrak{S}_i$ behave. We will prove that each substream has at most one 'big' (costly) cell and that the remaining cells add up to at most $\varepsilon$ times the cost of the big cell. Formally, we will say that a cell $\mathfrak{c}$ is *big*, if $\mathsf{COST}(\mathfrak{c}) \ge \varepsilon T$. A cell that is not big will be called *small*.

CLAIM 5.2.

$$\sum_{\mathfrak{c} \in G_i;\, \mathfrak{c} \text{ is big}} \mathit{COST}(\mathfrak{c}) \quad = \quad \mathcal{O}(\mathit{OPT}) \ .$$

*Proof.* For each big cell $\mathfrak{c}$, we open all facilities that are opened in an optimal solution in $\mathfrak{c}$ and all of its neighboring cells. Clearly, the sum of the connection costs of all big cells will be at most that of an optimal solution and each facility is used 9 times. Thus, the claim follows.

We will now use $\mathfrak{B}_i \subseteq \mathfrak{S}_i = h_i^{-1}(1)$ to be the set of big cells in $\mathfrak{S}_i$.

CLAIM 5.3. *There is a constant $c > 1$ such that*

$$\mathbf{Pr}[|\mathfrak{B}_i| > c \cdot s/\varepsilon] \le 1/200 \ .$$

*Proof.* By our choice of $r$ and the previous claim we know that the expected sum of costs of big cells mapped to 1 is $\mathcal{O}(s)$. Hence, in expectation there can be at most $\mathcal{O}(s/\varepsilon)$ such cells in $\mathfrak{B}_i$. Now, the claim follows by Markov's inequality.

CLAIM 5.4. *There is a constant $c$ such that, for $t \ge c \cdot (s/\varepsilon)^2$, the probablity that two cells in $\mathfrak{B}_i$ are mapped to the same value by $h'_i$ is less than $1/100$.*

*Proof.* For the moment, let us assume that $|\mathfrak{B}_i| \le c' \cdot s/\varepsilon$, for some constant $c' > 1$. Then, by pairwise independence of $h_i$ the probability that two fixed cells collide is $1/t$. Taking the union bound over all pairs gives that with probability at most $c' \cdot s/(\varepsilon \cdot t)$ there is a collision of big cells.

For $c \leq 200c'^2$, this probability is at most $1/200$. Due to Claim 5.3, with probability at least $1 - 1/200$, we have $|\mathfrak{B}_\mathfrak{i}| \leq c' \cdot s/\varepsilon$, which completes the proof.

It remains to analyze how the 'noise' introduced by small cells influences our result. Since there may be many cells that contain input points but no open facility of an optimal solution, we define $\mathsf{COST}^*(\mathfrak{c})$ of a cell $\mathfrak{c}$ to be the cost of an optimal solution, where a single facility is opened in the middle of the cell for free (and where additional facilities may be opened at a cost of $f$).

CLAIM 5.5. *There exists a constant $c > 1$ such that*

$$\mathbf{Pr}[\sum_{\mathfrak{c}\in\mathfrak{S}_\mathfrak{i}\,;\,\mathfrak{c} \text{ is small}} \mathsf{COST}^*(\mathfrak{c}) \geq cs] \quad \leq \quad 1/100 \ .$$

*Proof.* Obviously, the sum of the costs of all small cells in any level $i$ is $\mathsf{OPT}$. Since each small cell is in $\mathfrak{S}_\mathfrak{i}$ with probability $1/r$ and $T \geq 1$, the expected sum of the costs of all small cells in $\mathfrak{S}_\mathfrak{i}$ is $\mathsf{OPT} \cdot s/\mathrm{Apx} = \mathcal{O}(s)$. Now, the claim follows by Markov's inequality.

CLAIM 5.6. *There exists a constant $c > 1$ such that for $t \geq c^2s^2/(25\varepsilon^2T^2)$ with probability at least $98/100$ for every $j \in \{1, \ldots, t\}$*

$$\sum_{\mathfrak{c}\in {h'_i}^{-1}(j);\mathfrak{c} \text{ is small}} \mathsf{COST}^*(\mathfrak{c}) \quad \leq \quad \varepsilon T/2 \ .$$

*Proof.* We condition on the fact that Claim 5.5 is satisfied. We will use Chebyshev's inequality for fixed $j$ and then use the union bound. For a cell $\mathfrak{c}$ let

- $X_\mathfrak{c} = \mathsf{COST}^*(\mathfrak{c})$ if $h'_i(\mathfrak{c}) = j$
- $X_\mathfrak{c} = 0$ otherwise.

We have $\mathbf{Var}[X_\mathfrak{c}] \leq \frac{1}{t}(\mathsf{COST}^*(\mathfrak{c}))^2$. Let

$$X = \sum_{\mathfrak{c}\in {h'_i}^{-1}(j);\mathfrak{c} \text{ is small}} X_\mathfrak{c} \ .$$

By pairwise independence,

$$\begin{aligned}\mathbf{Var}[X] &= \sum_{\mathfrak{c}\in {h'_i}^{-1}(j);\mathfrak{c} \text{ is small}} \mathbf{Var}[X_\mathfrak{c}] \\ &\leq (cs)^2/t \ ,\end{aligned}$$

where $c$ is the constant from Claim 5.5.

By Chebyshev's inequality we have

$$\begin{aligned}\mathbf{Pr}[X \geq \varepsilon T] &\leq \frac{\mathbf{Var}[X]}{\varepsilon^2T^2} \\ &\leq \frac{c^2s^2}{4\varepsilon^2T^2t} \ .\end{aligned}$$

Thus, for $t \geq c^2s^2/(25\varepsilon^2T^2)$ the claim follows by taking the union bound over this event and the event from Claim 5.5.

Thus, with probability at least $9/10$ all above claims are true. In this case, the big cells will be hashed to different buckets and the noise by light cells with be at most $\varepsilon T/2 + f < \frac{2}{3}\varepsilon T$, where we need to add $f$ to place the facility in the middle of the cell.

Thus, if the extraction algorithm computes a $(1 + \varepsilon/3)$-approximation of the cost of every cell, it will report any $\delta$-detectable cell in $\mathfrak{G}_\mathfrak{i}$ and it will not report any cell, which is not maximal light. Furthermore, the cost of every such cell will be approximated within a factor of $(1 + \varepsilon)$.

It remains to use Chebyshev's inequality to prove that for some $s$, $s = (\log \Delta/\varepsilon)^{\mathcal{O}(1)}$, $rF$ is an additive approximation with error at most $\pm\frac{\varepsilon}{\lceil\log\Delta\rceil+1} \cdot \mathsf{OPT}$ of the contribution of maximal light cells in level $i$.

Finally, we need to use standard amplification techniques (run parallel instances and use the median) to ensure that for each level, we can approximate the above cost with probability at least $1 - \frac{1}{3(\lceil\log\Delta\rceil+1)}$ to conclude that the algorithm returns a correct approximation with probability at least $2/3$.

THEOREM 5.2. *Given access to a dynamic geometric data stream of insert and delete operations of points from $\{1, \ldots, \Delta\}^2$, there is a streaming algorithm that uses $(\log \Delta/\varepsilon)^{\mathcal{O}(1)}$ space and with probability at least $\frac{9}{10}$ maintains a value $\tilde{F}$ that is a $(1+\varepsilon)$-approximation to the cost of the facility location problem.*

## References

[1] N. Megiddo, K. J. Supowit, *On the complexity of some common geometric location problems* SIAM Journal on Computing, Vol. 13, No. 1. (1984), pp. 182-196.

[2] R. L. Francis and P. B. Mirchandani, *Discrete Location Theory* John Wiley and Sons, Inc., New York , 1990.

[3] Z. Drezner and H. W. Hamacher, *Facility Location: Applications and Theory* Springer Verlag , 2004.

[4] S. Arora, *Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems* Journal of the ACM, Vol. 45, No. 5. (1998), pp. 753-782.

[5] S. Arora and P. Raghavan and S. Rao, *Approximation Schemes for Euclidean k-Medians and Related Problems* Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC), (1998), pp. 106-113.

[6] M. Bădoiu and A. Czumaj and P. Indyk and C. Sohler, *Facility Location in Sublinear Time* Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP), (2005), pp. 866-877.

[7] K. Chan, *On Coresets for $k$-Median and $k$-Means Clustering in Metric and Euclidean Spaces and Their Applications* SIAM Journal on Computing, Vol. 39, No. 3. (2009), pp. 923-947.

[8] G. Cornuéjols and G. L. Nemhauser, and L. A. Wolsey, *The Uncapacitated Facility Location Problem* Discrete Location Theory, (1990), pp. 119-171.

[9] B. Degener and J. Gehweiler and C. Lammersen, *Kinetic Facility Location* Algorithmica, Vol. 57, No. 3. (2010), pp. 562-584.

[10] G. Frahling and P. Indyk and C. Sohler, *Sampling in Dynamic Data Streams and Applications* International Journal of Computational Geometry and Applications, Vol. 18, No. 1. (2008), pp. 3-28.

[11] G. Frahling and C. Sohler, *Coresets in Dynamic Geometric Data Streams* Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), (2005), pp. 209-217.

[12] S. Guha and S. Khuller, *Approximation Algorithms for Connected Dominating Sets* Algorithmica, Vol. 20, No. 4. (1998), pp. 374-387.

[13] J. Gehweiler and C. Lammersen and C. Sohler, *A Distributed $O(1)$-Approximation Algorithm for the Uniform Facility Location Problem* Proceedings of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), (2006), pp. 237-243.

[14] D. S. Hochbaum, *Heuristics for the Fixed Cost Median Problem* Mathematical Programming, Vol. 22, No. 1. (1982), pp. 148-162.

[15] S. Har-Peled and S. Mazumdar, *On Coresets for $k$-Means and $k$-Median Clustering* Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), (2004), pp. 291-300.

[16] P. Indyk, *Algorithms for Dynamic Geometric Problems over Data Streams* Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), (2005), pp. 373-380.

[17] M. Bădoiu and A. Czumaj and P. Indyk and C. Sohler, *Facility Location in Sublinear Time* Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP), (2005), pp. 866-877.

[18] S. G. Kolliopoulos and S. Rao, *A Nearly Linear-Time Approximation Scheme for the Euclidean $k$-Median Problem* SIAM Journal on Computing, Vol. 37, No. 3. (2007), pp. 757-782.

[19] S. G. Kolliopoulos and S. Rao, *A Nearly Linear-Time Approximation Scheme for the Euclidean $k$-Median Problem* Proceedings of the 7th Annual European Symposium on Algorithms (ESA), (1999), pp. 378-389.

[20] S. Li, *A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem* Proceedings of the 38th Annual International Colloquium on Automata, Languages and Programming (ICALP), (2011), pp. 77-88.

[21] C. Lammersen and C. Sohler, *Facility Location in Dynamic Geometric Data Streams* Proceedings of the 16th Annual European Symposium on Algorithms (ESA), (2008), pp. 660-671.

[22] R. R. Mettu and C. G. Plaxton, *The Online Median Problem* SIAM Journal on Computing, Vol. 32, No. 3. (2003), pp. 816-832.

[23] D. B. Shmoys and É. Tardos and K. Aardal, *Approximation Algorithms for Facility Location Problems* Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC), (1997), pp. 265-274.

[24] A. Andoni and K. DoBa and P. Indyk and D. Woodruff, *Efficient Sketches for Earth-Mover Distances with Applications* Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS), (2009), pp. 324-330.

[25] P. Indyk and D. P. Woodruff, *Optimal Approximations of the Frequency Moments of Data Streams* Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), (2005), pp. 202-208.

[26] T.S. Jayram and D.P. Woodruff, *The Data Stream Space Complexity of Cascaded Norms* Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS), (2009), pp. 765-774.

[27] M. Monemizadeh and D. Woodruff, $1$-*Pass Relative-Error $L_p$-Sampling with Applications* Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), (2010), pp. 1143-1160.