# Clustering High Dimensional Dynamic Data Streams

Vladimir Braverman [1] Gereon Frahling [2] Harry Lang [1] Christian Sohler [3] Lin F. Yang [1]

## Abstract

We present data streaming algorithms for the $k$-median problem in high-dimensional dynamic geometric data streams, i.e. streams allowing both insertions and deletions of points from a discrete Euclidean space $\{1, 2, \ldots \Delta\}^d$. Our algorithms use $k\epsilon^{-2}\mathrm{poly}(d \log \Delta)$ space/time and maintain with high probability a small weighted set of points (a coreset) such that for every set of $k$ centers the cost of the coreset $(1+\epsilon)$-approximates the cost of the streamed point set. We also provide algorithms that guarantee only positive weights in the coreset with additional logarithmic factors in the space and time complexities. We can use this positively-weighted coreset to compute a $(1+\epsilon)$-approximation for the $k$-median problem by any efficient offline $k$-median algorithm. All previous algorithms for computing a $(1+\epsilon)$-approximation for the $k$-median problem over dynamic data streams required space and time exponential in $d$. Our algorithms can be generalized to metric spaces of bounded doubling dimension.

## 1 Introduction

The analysis of very large data sets is still a big challenge. Particularly, when we would like to obtain information from data sets that occur in the form of a data stream like, for example, streams of updates to a data base system, internet traffic and measurements of scientific experiments in astro- or particle physics (e.g. (Liu et al., 2015)). In such scenarios it is difficult and sometimes even impossible to store the data. Therefore, we need algorithms that process the data sequentially and maintain a summary of the data using space much smaller than the size of the stream. Such algorithms are often called streaming algorithms (for more introduction on streaming algorithms, please refer to (Muthukrishnan, 2005)).

One fundamental technique in data analysis is clustering. The idea is to group data into clusters such that data inside the same cluster is similar and data in different clusters is different. Center based clustering algorithms also provide for each cluster a cluster center, which may act as a representative of the cluster. Often data is represented as vectors in $\mathbb{R}^d$ and similarity between data points is often measured by the Euclidean distance. Clustering has many applications ranging from data compression to unsupervised learning.

In this paper we are interested in clustering problems over dynamic data streams, i.e. data streams that consist of updates, for example, to a database. Our stream consists of insert and delete operations of points from $\{1, \ldots, \Delta\}^d$. We assume that the stream is consistent, i.e. there are no deletions of points that are not in the point set and no insertions of points that are already in the point set. We consider the $k$-median clustering problem, which for a given a set of points $P \subseteq \mathbb{R}^d$ asks to compute a set $C$ of $k$ points that minimizes the sum of distances of the input points to their nearest points in $C$.

### 1.1 Our Results

We develop the first $(1+\epsilon)$-approximation algorithm for the $k$-median clustering problem in dynamic data streams that uses space polynomial in the dimension of the data. To our best knowledge, all previous algorithms required space exponentially in the dimension. Formally, our main theorem states,

**Theorem 1.1** (Main Theorem)**.** *Fix $\epsilon \in (0, 1/2)$, positive integers $k$ and $\Delta$, Algorithm 1 makes a single pass over the dynamic streaming point set $P \subset [\Delta]^d$, outputs a weighted set $S$, such that with probability at least 0.99, $S$ is an $\epsilon$-coreset for $k$-median of size $O\left(kd^4L^4/\epsilon^2\right)$, where $L = \log \Delta$. The algorithm uses $\tilde{O}\left(kd^7L^7/\epsilon^2\right)$ bits in the worst case, processes each update in time $\tilde{O}(dL^2)$ and outputs the coreset in time $\mathrm{poly}(d, k, L, 1/\epsilon)$ after one pass of the stream.*

The theorem is restated in Theorem 3.6 and the proof is presented in Section 3.3. The coreset we constructed may contain negatively weighted points. Thus naïve offline algorithms do not apply directly to finding $k$-clustering solutions on the coreset. We also provide an alternative approach that output only non-negatively weighted coreset. The new algorithm is slightly more complicated. The space complexity and coreset size is slightly worse than the one with negative weights but still polynomial in $d$, $1/\epsilon$ and $\log \Delta$ and optimal in $k$ up to polylog$k$ factor.

**Theorem 1.2** (Alternative Results)**.** *Fix $\epsilon \in (0, 1/2)$, posi-*

[1]Johns Hopkins University, USA [2]Linguee GmbH [3]TU Dortmund. Correspondence to: Lin F. Yang <lyang@jhu.edu>, Christian Sohler <christian.sohler@tu-dortmund.de>.

*Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

*tive integers $k$ and $\Delta$, Algorithm 6 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set $S$ with* non-negative weights *for each point, such that with probability at least* $0.99$*, $S$ is an $\epsilon$-coreset for $k$-median of size $\tilde{O}\left(kd^4L^4/\epsilon^2\right)$. The algorithm uses $\tilde{O}\left(kd^8L^8/\epsilon^2\right)$bits in the worst case. For each update of the input, the algorithm needs* $\text{poly}(d, 1/\epsilon, L, \log k)$ *time to process and outputs the coreset in time* $\text{poly}(d, k, L, 1/\epsilon)$ *after one pass of the stream.*

The theorem is restated in Theorem 4.3 in Section 4 and the proof is presented therein. Both approaches can be easily extended to maintain a coreset for a general metric space.

### 1.2 Our Techniques

From a high level, both algorithms can be viewed as a combination of the ideas introduced by Frahling and Sohler (Frahling & Sohler, 2005) with the coreset construction by Chen (Chen, 2009).

To explain our high-level idea, we first summarize the idea of Chen (Chen, 2009). In their construction, they first obtain a $(\alpha, \beta)$-bi-criterion solution. Namely find a set of at most $\alpha k$ centers such that the $k$-median cost to these $\alpha k$ centers is at most $\beta\mathsf{OPT}$, where $\mathsf{OPT}$ is the optimal cost for a $k$-median solution. Around each of the $\alpha k$ points, they build logarithmically many concentric ring regions and sample points from these rings. Inside each ring, the distance from a point to its center is upper and lower bounded. Thus the contribution to the optimal cost from the points of this ring is lower bounded by the number of points times the inner diameter of the ring. To be more precise, their construction requires a partition of $\tilde{O}(\alpha k)$ sets of the original data points satisfying the following property: for the partition $P_1, P_2, \ldots, P_{k'}$, $\sum_i |P_i|\text{diam}(P_i) \lesssim \beta\mathsf{OPT}$. They then sample a set of points from each part to estimate the cost of an arbitrary $k$-set from $[\Delta]^d$ up to $O(\epsilon|P_i|\text{diam}(P_i)/\beta)$ additive error. Combining the samples of the $k'$ parts, this gives an additive error of at most $\epsilon\mathsf{OPT}$ and therefore an $\epsilon$-coreset.

The first difficulty in generalizing the construction of Chen to dynamic streams is that it depends on first computing approximate centers, which seems at first glance to require two passes. Surprisingly (since we would like to be polynomial in $d$), we can resolve this difficulty using a grid-based construction. The grid structure can be viewed as a $(2^d)$-ary tree of cells. The root level of the tree is a single cell containing the entire set of points. Going down a level through the tree, each parent cell is split evenly into $2^d$ subcells. Thus in total there are $\log_2 \Delta$ grid levels. Each cell of the finest level contains at most a single point.

Without using any information of a pre-computed $(\alpha, \beta)$-bi-criterion solution to the $k$-median problem, as it does in (Chen, 2009), our first idea (similar to the idea used in (Indyk, 2004)) is to use a randomly shifted grid (i.e. shift each coordinate of the cell of the root level by a random value $r$, where $r$ is uniformly chosen from $\{1, 2, \ldots \Delta\}$, and redefine the tree by splitting cells into $2^d$ subcells recursively). We show that with high probability, in each level, at most $\tilde{O}(k)$ cells are close to (or containing) a center of an optimal solution to the $k$-median. For the remaining cells, we show that each of them cannot contain too many points, since otherwise they would contribute too much to the cost of the optimal solution (since each point in these cells is far away from each of the optimal centers). We call the cells containing too many points in a level *heavy* cells. The immediate non-heavy children of the heavy cells form a partition of the entire point sets (i.e. the cells that are not heavy, but have heavy parents). Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{k'}$ be these cells, and we can immediately show that $\sum_i |\mathcal{C}_i|\text{diam}(\mathcal{C}_i) \leq \beta\mathsf{OPT}$ for some $\beta = O(d^{3/2})$. If we can identify the heavy cells (e.g. use heavy hitter algorithms), and sample points from their immediate non-heavy children in a dynamic stream, we will obtain a construction similar to Chen (Chen, 2009).

Our second idea allows us to significantly reduce space requirements and also allows us to do the sampling more easily. For each point $p$, the cells containing it form a path on the grid tree. We write each point as a telescope sum as the cell centers on the path of the point ( recall that the grids of each level are nested and the $c^0$ is the root of the tree). For example, let $c^0, c^1, \ldots, c^L$ be the cell centers of the path, where $c_L = p$, and define $\mathbf{0}$ to be the zero vector. Then $p = c^L - c^{L-1} + c^{L-1} - c^{L-2} \ldots + c^0 - \mathbf{0} + \mathbf{0}$. In this way, we can represent the distance from a point to a set of points as the distance of cell centers to that set of points. For example, let $Z \subset [\Delta]^d$ be a set of points, and $d(p, Z)$ be the distance from $p$ to the closest point in $Z$. Then $d(p, Z) = d(c^L, Z) - d(c^{L-1}, Z) + d(c^{L-1}, Z) - d(c^{L-2}, Z) + \ldots + d(\mathbf{0}, Z)$. Thus we can decompose the cost a set $Z$ into $L + 2$ levels: the cost in level $l \in [0, L]$ is $\sum_p d(c^l_p, Z) - d(c^{l-1}_p, Z)$, where $c^p_l$ is the center of the cell containing $p$ in level $l$ and the cost in the $(-1)$-st level is $|P|d(\mathbf{0}, Z)$, where $P$ is the entire points set. Since $|d(c^l_p, Z) - d(c^{l-1}_p, Z)|$ is bounded by the cell diameter of the level, we can sample points from the non-heavy cells of the entire level, and guarantee that the cost of that level is well-approximated. Notice that (a) we do not need to sample $\tilde{O}(k)$ points from every part of the partition, thus we save a $k$ factor on the space and (b) we do not need to sample the actual points, but only an estimation of the number of points in each cell, thus the sampling becomes much easier (there is no need to store the sampled points).

In the above construction, we are able to obtain a coreset, but the weights can be negative due the the telescope sum. It is not easy find an offline $k$-median algorithm to output the solutions from a negatively-weighted coreset. To remove the negative weights, we need to adjust the weights of cells. But the cells with a small number of points (compared to the heavy cells) are problematic – the sampling-based empirical estimations of the number of points in them has too much error to be adjusted.

In our second construction, we are able to remove all the negative weights. The major difference is that we introduce a cut-off on the telescope sum. For example, $d(p, Z) =$

$d(c_p^L, Z) - d(c_p^{l(p)}, Z) + d(c_p^{l(p)}) - d(c_p^{l(p)-1}) + \ldots + d(\mathbf{0}, Z)$ where $l(p)$ is a cutoff level of point $p$ such that the cell containing $p$ in level $l(p)$ is heavy but no longer heavy in level $l(p)+1$. We then sample point $p$ with some probability defined according to $l(p)$. In other words, we only sample points from heavy cells and not from non-heavy ones. Since a heavy cell contains enough points, the sampling-based estimation of the number of points is accurate enough and thus allows us to adjust them to be all positive.

Finally, to handle the insertion and deletions, we use a $F_2$-heavy hitter algorithm to identify the heavy cells. We use pseudo-random hash functions (e.g. Nisan's construction (Nisan, 1992; Indyk, 2000b) or $k$-wise independent hash functions) to do the sampling and use a `K-Set` data structure (Ganguly, 2005) to store the sampled points in the dynamic stream.

### 1.3 Related Work

There is a rich history in studies of geometric problems in streaming model. Among these problems some excellent examples are: approximating the diameter of a point set (Feigenbaum et al., 2005; Indyk, 2003), approximately maitain the convex hull (Cormode & Muthukrishnan, 2003; Hershberger & Suri, 2004), the min-volume bounding box (Chan, 2004; Chan & Sadjad, 2006), maintain $\epsilon$-nets and $\epsilon$-approximations of a data stream (Bagchi et al., 2007). Clustering problem is another interesting and popular geometric problem studied in streaming model. There has been a lot of works on clustering data streams for the $k$-median and $k$-means problem based on coresets (Har-Peled & Mazumdar, 2004; Har-Peled & Kushal, 2005; Chen, 2009; Feldman et al., 2007; 2013; Feldman & Langberg, 2011). Additionally (Charikar et al., 1997; Guha et al., 2000; Meyerson, 2001) studied the problem in the more general metric space. The currently best known algorithm for $k$-median problem in this setting is an $O(1)$-approximation using $O(k\text{polylog}n)$ space (Charikar et al., 2003). However, all of the above methods do not work for dynamic streams.

The most relevant works to ours are those by Indyk (Indyk, 2004), Indyk & Price (Indyk & Price, 2011) and Frahling & Sohler (Frahling & Sohler, 2005). Indyk (Indyk, 2004) introduced the model for dynamic geometric data streamings. He studied algorithms for (the weight of) minimum weighted matching, minimum bichromatic matching and minimum spanning tree and $k$-median clustering. He gave a exhaustive search $(1+\epsilon)$ approximation algorithm for $k$-median and a $(\alpha, \beta)$-bi-criterion approximation algorithm. Indyk & Price (Indyk & Price, 2011) studied the problem of sparse recovery under Earth Mover Distance. They show a novel connection between EMD/EMD sparse recovery problem to $k$-median clustering problem on a two dimensional grid. The most related work to current one is Frahling & Sohler (Frahling & Sohler, 2005), who develop a streaming $(1+\epsilon)$-approximation algorithms for $k$-median as well as other problems over dynamic geometric data streams. All previous constructions for higher dimensional grid require space exponential in the dimension $d$.

## 2 Preliminaries

For integer $a \le b$, we denote $[a] := \{1, 2, \ldots, a\}$ and $[a, b] := \{a, a+1, \ldots, b\}$ for integer intervals. We will consider a point set $P$ from the Euclidean space $\{1, \ldots, \Delta\}^d$. Without loss of generality, we always assume $\Delta$ is of the form $2^L$ for some integer $L$, since otherwise we can always pad $\Delta$ without loss of a factor more than 2. Our streaming algorithm will process insertions and deletions of points from this space. We study the $k$-median problem, which is to minimize $\text{cost}(P, Z) = \sum_{p \in P} d(p, Z)$ among all sets $Z$ of $k$ centers from $\mathbb{R}^d$ and where $d(p, q)$ denotes the Euclidean distance between $p$ and $q$ and $d(p, Z)$ for a set of points $Z$ denotes the distance of $p$ to the closest point in $Z$. The following definition is from (Har-Peled & Mazumdar, 2004).

**Definition 2.1.** *Let $P \subseteq [\Delta]^d$ be a point set. A small weighted set $S$ is called an $\epsilon$-coreset for the $k$-median problem, if for every set of $k$ centers $Z \subset [\Delta]^d$ we have* [1]

$$(1-\epsilon) \cdot \text{cost}(P, Z) \le \text{cost}(S, Z) \le (1+\epsilon) \cdot \text{cost}(P, Z),$$

*where $\text{cost}(S, Z) := \sum_{s \in S} \text{wt}(s) d(s, Z)$ and $\text{wt}(s)$ is the weight of point $s \in S$.*

Through out the paper, we assume parameters $\epsilon, \rho, \delta, \in (0, \frac{1}{2})$ unless otherwise specified. For our algorithms and constructions we define a nested grid with $L$ levels, in the following manner.

**Definition of grids** Let $v = (v_1, \ldots, v_d)$ be a vector chosen uniformly at random from $[0, \Delta - 1]^d$. Partition the space $\{1, \ldots, \Delta\}^d$ into a regular Cartesian grid $\mathcal{G}_0$ with side-length $\Delta$ and translated so that a vertex of this grid falls on $v$. Each cell of this grid can be expressed as $[v_1 + n_1\Delta, v_1 + (n_1+1)\Delta) \times \ldots \times [v_d + n_d\Delta, v_d + (n_d+1)\Delta)$ for some $(n_1, \ldots, n_d) \in \mathbb{Z}^d$. For $i \ge 1$, define the regular grid $\mathcal{G}_i$ as the grid with side-length $\Delta/2^i$ aligned such that each cell of $\mathcal{G}_{i-1}$ contains $2^d$ cells of $\mathcal{G}_i$. The finest grid is $\mathcal{G}_L$ where $L = \lceil \log_2 \Delta \rceil$; the cells of this grid therefore have side-length at most 1 and thus contain at most a single input point. Each grid forms a partition of the point-set $\mathcal{S}$. There is a $d$-ary tree such that each vertex at depth $i$ corresponds to a cell in $\mathcal{G}_i$, and this vertex has $2^d$ children which are the cells of $\mathcal{G}_{i+1}$ that it contains. For convenience, we define $\mathcal{G}_{-1}$ as the entire dataset and it contains a single cell $\mathcal{C}_{-1}$. For each cell $\mathcal{C}$, we also treat it as a subset of the input points (i.e. $\mathcal{C} \cap P$) if there is no confusion.

We denote $Z^* \subset [\Delta]^d$ as the optimal solution for $k$-median and $\mathsf{OPT}$ as the optimal cost for $Z^*$. The proof of the following lemma is delayed to Section A.

**Lemma 2.2.** *Fix a set $Z \subset [\Delta]^d$, then with probability at least $1-\rho$, for every level $i \in [0, L]$, the number of cells that satisfy $d(\mathcal{C}, Z) \le \Delta/(2^{i+1}d)$ is at most $e|Z|(L+1)/\rho$.*

[1]For simplicity of the presentation, we define the coreset for all sets of $k$ centers $Z \subset [\Delta]^d$, but it can be generalized to all sets of k centers $Z \subset \mathbb{R}^d$ with an additional $\text{polylog}(1/\epsilon)$ factor in the space. We discuss this point further in Section 6.

### 2.1 Outline

In Section 3, we introduce the coreset with negative weights. In Section 4, we introduce a modified construction with all positive weights. Section 6 comes with the final remarks.

## 3 Generally Weighted Coreset

In this section, we present our generally weighted coreset construction. In Section 3.1, we introduce the telescope sum representation of a point $p$ and the coreset framework. In Section 3.2, we illustrate our coreset framework with an offline construction. In Section 3.3 we present an one pass streaming algorithm that implements our coreset framework.

### 3.1 The Telescope Sum and Coreset Framework

Our first technical idea is to write each point as a telescope sum. We may interpret this sum as replacing a single point by a set of points in the following way. Each term $(p - q)$ of the sum can be viewed as a pair of points $p$ and $q$, where $p$ has weight 1 and $q$ has weight $-1$. The purpose of this construction is that the contribution of each term $(p - q)$ (or the corresponding two points) is bounded. This can be later exploited when we introduce and analyze our sampling procedure.

We now start to define the telescope sum, which will relate to our nested grids. For each $\mathcal{C} \in G_i$, denote $c(\mathcal{C})$ (or simply $c$) as its center. For each point $p \in P$, define $\mathcal{C}(p, i)$ as the cell that contains $p$ in $\mathcal{G}_i$, and $c_p^i$ is the center of $\mathcal{C}(p, i)$. Then we can write

$$p = c_p^{-1} + \sum_{i=0}^{L} c_p^i - c_p^{i-1}.$$

where we set $c_p^{-1} = \mathbf{0}$ (we also call this the cell center of the $(-1)$-st level for convenience). The purpose of this can be seen when we consider the distance of $p$ to an arbitrary $k$-centers $Z \subset [\Delta]^d$, we can write the cost of a single point $p$, as

$$d(p, Z) = d(c_p^{-1}, Z) + \sum_{i=0}^{L} d(c_p^i, Z) - d(c_p^{i-1}, Z).$$

Note that $c_p^L = p$ since the cells of $\mathcal{G}_L$ contain a single point. Thus the cost of the entire set $\mathrm{cost}(P, Z)$ can be written as,

$$\sum_{i=0}^{L} \sum_{p \in P} d(c_p^i, Z) - d(c_p^{i-1}, Z) + \sum_{p \in P} d(c_p^{-1}, Z). \tag{1}$$

As one can see, we transform the cost defined using the original set of points to the "cost" defined using cell centers. To estimate the cost, it remains to estimate each of the terms, $\sum_{p \in P} d(c_p^i, Z) - d(c_p^{i-1}, Z)$ for $i \in [0, L]$ and $\sum_{p \in P} d(c_p^{-1}, Z)$. In other words, assign weights to each of the centers of the grid cells. For $i \in [0, L]$, and a cell $\mathcal{C} \in \mathcal{G}_i$, denote $\mathcal{C}^P$ as the parent cell of $\mathcal{C}$ in grid $\mathcal{G}_{i-1}$. Thus we can rewrite the cost term as follows,

$$\begin{aligned}
\mathrm{cost}(\mathcal{G}_i, Z) &:= \sum_{p \in P} d(c_p^i, Z) - d(c_p^{i-1}, Z) \\
&= \sum_{\mathcal{C} \in \mathcal{G}_i} \sum_{p \in \mathcal{C}} d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \\
&= \sum_{\mathcal{C} \in \mathcal{G}_i} |\mathcal{C}| \left[ d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right] \\
&= \sum_{\mathcal{C} \in \mathcal{G}_i} |\mathcal{C}| d(c(\mathcal{C}), Z) \\
&\quad - \sum_{\mathcal{C}' \in \mathcal{G}_{i-1}} \sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \subset \mathcal{C}'} |\mathcal{C}| d(c(\mathcal{C}'), Z).
\end{aligned} \tag{2}$$

For $i = -1$, we denote $\mathrm{cost}(\mathcal{G}_{-1}, Z) = |P| d(c_p^{-1}, Z)$. Then this leads to our following coreset construction framework.

**Generally Weighted Construction** The coreset $S$ in the construction is composed by a weighted subset of centers of grid cells. The procedure of the construction is to assign some (integer) value to each cell center. For instance, maintain a integer valued function $\widehat{|\cdot|}$ on cells (using small amount of space). $\widehat{|\mathcal{C}|}$ is called the *value* of the cell $\mathcal{C}$. Let $c$ be the center of $\mathcal{C}$, then the weight for $c$ is

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\mathcal{C}' : \mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}} \widehat{|\mathcal{C}'|}. \tag{3}$$

And for the $L$-th grid $\mathcal{G}_L$, the weight for each cell $\mathcal{C}$ is just $\widehat{|\mathcal{C}|}$. Note that there might be negative weights for some cells.

As a naïve example, we set $\widehat{|\mathcal{C}|} := |\mathcal{C}|$ as the exact number of points of a cell $\mathcal{C}$. Then we would expect the cells in every level except those in $\mathcal{G}_L$ have weight 0. In other words, we stored the entire point set as the coreset. As we will show, if we allow $\widehat{|C|}$ as an approximation of $|C|$ up to additive error, we can compress the number of non-zero weighted centers to be a smaller number.

**Definition 3.1.** *Given a grid structure, and a real valued function $\widehat{|\cdot|}$ on the set of cells. We define a function $\widehat{\mathrm{cost}} : [\Delta]^d \times \mathcal{G} \to \mathbb{R}$ as follows, for $i \in [0, L]$ and $Z \subset [\Delta]^d$,*

$$\begin{aligned}
\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) &:= \sum_{\mathcal{C} \in \mathcal{G}_i} \widehat{|\mathcal{C}|} d(c(\mathcal{C}), Z) \\
&\quad - \sum_{\mathcal{C}' \in \mathcal{G}_{i-1}} \sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \subset \mathcal{C}'} \widehat{|\mathcal{C}|} \cdot d(c(\mathcal{C}'), Z),
\end{aligned} \tag{4}$$

*and $\widehat{\mathrm{cost}}(\mathcal{G}_{-1}, Z) = \widehat{|\mathcal{C}_{-1}|} d(\mathbf{0}, Z)$, where $\mathcal{C}_{-1}$ is the cell in $\mathcal{G}_{-1}$ containing the entire set of points.*

**Lemma 3.2.** *Fix an integer valued function $\widehat{|\cdot|}$ on the set of cells and parameter $0 < \epsilon < \frac{1}{2}$. Let $S$ be the set of all cell centers with weights assigned by Equation (3). If $\widehat{|\mathcal{C}_{-1}|} = |P|$ (recall that $\mathcal{C}_{-1}$ is the first cell containing the entire dataset) and for any $Z \subset [\Delta]^d$ with $|Z| \le k$ and*

$i \in [0, L]$

$$\left|\mathrm{cost}(\mathcal{G}_i, Z) - \widehat{\mathrm{cost}}(\mathcal{G}_i, Z)\right| \le \frac{\epsilon \mathit{OPT}}{L+1},$$

*then $S$ is an $\epsilon$-coreset for $k$-median.*

*Proof.* Given an arbitrary set of centers $Z \subset [\Delta]^d$,

$$\begin{aligned}
\mathrm{cost}(S, Z) &= \sum_{s\in S} \mathrm{wt}(s) d(s, Z) \\
&= \sum_{i\in[0,L]} \sum_{\mathcal{C}\in\mathcal{G}_i} d(c(\mathcal{C}), Z)\big(|\widehat{\mathcal{C}}| - \sum_{\substack{\mathcal{C}':\mathcal{C}'\in\mathcal{G}_{i+1}\\ \mathcal{C}'\subset\mathcal{C}}} |\widehat{\mathcal{C}'}|\big) \\
&\quad + |P| d(\mathbf{0}, Z) \\
&= \sum_{i\in[0,L]} \big[ \sum_{\mathcal{C}\in\mathcal{G}_i} |\widehat{\mathcal{C}}| d(c(\mathcal{C}), Z) - \sum_{\substack{\mathcal{C}'\in\mathcal{G}_{i-1}\\ \mathcal{C}\in\mathcal{G}_i:\mathcal{C}\subset\mathcal{C}'}} |\widehat{\mathcal{C}}| d(c(\mathcal{C}'), Z)\big] \\
&\quad + |P| d(\mathbf{0}, Z) = \sum_{i\in[0,L]} \widehat{\mathrm{cost}}(\mathcal{G}_i, Z).
\end{aligned}$$

It follows that $|\mathrm{cost}(S, Z) - \mathrm{cost}(P, Z)| \le \epsilon\mathsf{OPT}$. ☐

## 3.2 An Offline Construction

In this section, we assume we have $(10, 10)$-bi-criterion approximation to $k$-median. Let $Z' = \{z'_1, z'_2, \ldots, z'_{10k}\}$ be the centers and $o$ is the cost satisfying $\mathsf{OPT} \le o \le 10\mathsf{OPT}$. This can be done using (Indyk, 2000a). We will show how we construct the coreset base on the framework described in the last section.

**An Offline Construction** For each point in level $\mathcal{G}_{-1}$, we sample it with probability $\pi_{-1} = 1$ (i.e. count the number of points exactly) and set $|\widehat{\mathcal{C}_{-1}}| := |P|$. For each level $i \in [0, L]$, we pick the set of all cells $\mathcal{C}$ satisfying $d(\mathcal{C}, Z') \le W/(2d)$, where $W$ is the side length of $\mathcal{C}$. Denote the set of these cells as $C_{Z'}$. We count the number of points in each of these cells exactly, and set $|\widehat{\mathcal{C}}| := |\mathcal{C}|$. For the points in the rest of cells, for each $i \in [0, L]$, we sample the points with probability

$$\pi_i = \min\left(\frac{200(L+1)^2\Delta d^2}{2^i\epsilon^2 o} \ln\frac{2(L+1)\Delta^{kd}}{\rho}, 1\right) \quad (5)$$

uniformly and independently. Denote $S_i$ as the set of sampled points at level $i$. For each $\mathcal{C} \notin C_{Z'}$, set

$$|\widehat{\mathcal{C}}| := |S_i \cap \mathcal{C}|/\pi_i.$$

Then, from the bottom level to the top level, we assign the weight to the cell centers of each of the cells and their parent cells with non-zero $|\widehat{\mathcal{C}}|$ using (3). Denote $\mathcal{S}$ as the coreset, which contains the set of cell centers of non-zero weight.

**Theorem 3.3.** *Fix $\epsilon, \rho \in (0, 1/2)$, then with probability at least $1 - 8\rho$, the offline construction $\mathcal{S}$ is an $\epsilon$-coreset for k-median and that*

$$|\mathcal{S}| = O\left(\frac{d^4kL^4}{\epsilon^2}\log\frac{1}{\rho} + \frac{kL^2}{\rho}\right).$$

*Proof of Theorem 3.3.* By definition $|\widehat{\mathcal{C}_{-1}}| = |P|$, it is suffice to show that with probability at least $1 - 4\rho$, for every $i \in [0, L]$ and every $k$-set $Z \subset [\Delta]^d$,

$$\left|\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) - \mathrm{cost}(\mathcal{G}_i, Z)\right| \le \epsilon\mathsf{OPT}/(L+1).$$

It follows from Lemma 3.2 that, $\mathcal{S}$ is an $\epsilon$-coreset.

Let $S_i$ be the sampled points of level $i$. Fix a $k$-set $Z \subset [\Delta]^d$, for each $i \in [0, L]$, by equation (2), we have that, $\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) = \sum_{\mathcal{C}\in C_{Z'}} |\widehat{\mathcal{C}}| \big(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\big) + \sum_{p\in\mathcal{S}_i}(d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i$. Note that $E(\widehat{\mathrm{cost}}(\mathcal{G}_i, Z)) = \mathrm{cost}(\mathcal{G}_i, Z)$. The first term contributes 0 to the difference $\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) - \mathrm{cost}(\mathcal{G}_i, Z)$ since each $|\widehat{\mathcal{C}}|$ is exact. It remains to bound the error contribution from the second part. Denote $A_2 = \sum_{p\in\mathcal{S}_i}(d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i$. Recall that $Z'$ is the centers of the bi-criterion solution and $C_{Z'}$ is the set of cells with distance less than $W/(2d)$ to $Z'$, where $W$ is the side-length of a cell. Let $\mathcal{A}$ be event that $|C_{Z'}| \le e|Z'|(L+1)^2/\rho = O(kL^2/\rho)$. By Lemma 2.2, $\mathcal{A}$ happens with probability at least $1-\rho$. Conditioning on $\mathcal{A}$ happening, for each point $p \in \mathcal{C} \notin C_{Z'}$, we have that $d(p, Z') \ge \mathrm{diam}(\mathcal{C})/(2d^{3/2})$. Therefore, $\sum_{p\in\mathcal{C}\notin C_Z} \mathrm{diam}(\mathcal{C}) \le (2d^{3/2})\sum_{p\in\mathcal{C}\notin C_Z} d(p, Z') \le 20d^{3/2}\mathsf{OPT}$. By Lemma 3.4, with probability at least $1 - \frac{\rho}{(L+1)\Delta^{kd}}$, $|A_2 - E(A_2)| \le \frac{\epsilon\mathsf{OPT}}{L+1}$. Since there are at most $\Delta^{kd}$ many different $k$-sets from $[\Delta]^d$, thus, for a fixed $i \in [0, L]$ with probability at least $1 - \frac{\rho}{L+1}$, for all $k$-sets $Z \subset [\Delta]^d$, $\left|\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) - \mathrm{cost}(\mathcal{G}_i, Z)\right| \le \epsilon\mathsf{OPT}/(L+1)$. By the union bound, with probability at least $1 - 4\rho$, $\mathcal{S}$ is the desired coreset.

It remains to bound the size of $\mathcal{S}$. Conditioning on $\mathcal{A}$ happening, then $|C_{Z'}| = O(kL^2/\rho)$. For each level $i$, since each point from cells $\mathcal{C} \notin C_{Z'}$ contributes at least $\Delta/(2^{i+1}d)$ to the bi-criterion solution, there are at most $O(2^i\mathsf{OPT}d/\Delta)$ points in cells not in $C_{Z'}$. By a Chernoff bound, with probability at least $1 - \rho/(L+1)$, the number of points sampled from cells $\mathcal{C} \notin C_{Z'}$ of level $i$ is upper bounded by $O(d^4kL^3\log\frac{1}{\rho}/\epsilon^2)$. Thus for all levels, with probability at least $1-\rho$, the number of points sampled is upper bounded by $O(d^4kL^4\log\frac{1}{\rho}/\epsilon^2)$, which is also an upper bound of the number of cells occupied by sampled points. Now we bound the number of non-zero weighted centers. In the coreset construction, if a cell center has non-zero weight, then either itself or one of its children cells has non-zero assigned value $|\widehat{\mathcal{C}}|$. Thus the number of non-zero weigted centers is upper bound by 2 times the number of non-zero valued cells. Thus $|\mathcal{S}| = O(d^4kL^4\log\frac{1}{\rho}/\epsilon^2 + \frac{kL^2}{\rho})$. ☐

**Lemma 3.4.** *Fix $\epsilon, \rho \in (0, 1/2)$, if a set of cells $C$ from grid $\mathcal{G}_i$ satisfies $\sum_{\mathcal{C}\in C} |\mathcal{C}|\mathrm{diam}(\mathcal{C}) \le \beta\mathit{OPT}$ for some $\beta \ge 2\epsilon/(3(L+1))$, let $S$ be a set of independent samples from the point set $\cup\{\mathcal{C} \in C\}$ with probability*

$$\pi_i \ge \min\left(\frac{3a(L+1)^2\Delta\sqrt{d}\beta}{2^i\epsilon^2 o} \ln\frac{2\Delta^{kd}(L+1)}{\rho}, 1\right)$$

*where $0 < o \le a\mathsf{OPT}$ for some $a > 0$, then for a fixed set $Z \subset [\Delta]^d$, with probability at least $1 - \rho/((L+1)\Delta^{kd})$,*

$$|\sum_{p\in S}(d(c_p^i, Z) - d(c_p^{i-1}, Z))/\pi_i - \sum_{p\in\cup\{\mathcal{C}\in C\}}(d(c_p^i, Z) - d(c_p^{i-1}, Z))| \le \frac{\epsilon OPT}{L+1}.$$

The proof is a straightforward application of Bernstein inequality. It is presented in Section A.

## 3.3 The Streaming Algorithm

For the streaming algorithm, the first challenge is that we do not know the actual value of OPT, neither do we have an $(\alpha, \beta)$-bi-criterion solution. To handle this, we will show that we do not need an actual set of centers of an approximate solution, and that a conceptual optimal solution suffices. We will guess logarithmically many values for OPT to do the sampling. We re-run the algorithm in parallel for each guess of OPT.

The second challenge is that we cannot guarantee the sum $\sum_{\mathcal{C}\in\mathcal{G}_i} \mathrm{diam}(\mathcal{C})$ to be upper bounded by $\beta\mathsf{OPT}$ as required in Lemma 3.4. We will show that we can split the set of cells into two parts. The first part satisfies the property that $\sum_{\mathcal{C}} |\mathcal{C}|\mathrm{diam}(\mathcal{C}) \le \beta\mathsf{OPT}$ for some parameter $\beta$. The second part satisfies that $|\mathcal{C}|\mathrm{diam}(\mathcal{C}) \ge a\mathsf{OPT}/k$ for some constant $a$.

For the first part, we use a similar sampling procedure as we did in the offline case. The challenge here is that there might be too many points sampled when the algorithm is midway through the stream, and these points may be deleted later in the stream. To handle this case, we use a data structure called `K-Set` structure with parameter $k$ (Ganguly, 2005). We will insert (with deletions) a multiset of points $M \subset [N]$ into the `K-Set`. The data structure processes each stream operation in $O(\log(k/\delta))$ time. At each point of time, it supports an operation `RETRSET`, that with probability at least $1-\delta$ either returns the set of items of $M$ or returns `Fail`. Further, if the number of distinct items $|M|$ is at most $k$, then `RETRSET` returns $M$ with probability at least $1-\delta$. The space used by the `K-Set` data structure is $O(k(\log|M| + \log N)\log(k/\delta)$. The `K-Set` construction also returns the frequency of each stored points upon the `RETRSET` operation.

For the second part, we call these cells *heavy*. We first upper bound the number of heavy cells by $\alpha k$ for some $\alpha > 1$. We use a heavy hitter algorithm `HEAVY-HITTER` to retrieve an approximation to the number of points in these cells. The guarantee is given in the following theorem. In an insertion-deletion stream, it may that although the stream has arbitrary large length, at any moment a much smaller number of elements are active (that is, inserted and not yet deleted). We define the size of a stream to be the maximum number of active elements at any point of the stream.

**Theorem 3.5** ((Larsen et al., 2016) Theorem 2)**.** *Fix $\epsilon, \delta \in (0, 1/2)$. Given a stream (of insertions and deletions) of size $m$ consisting of items from universe $[n]$, there exists an algorithm `HEAVY-HITTER`$(n, k, \epsilon, \delta)$ that makes a single pass over the stream and outputs a set of pairs $H$. With probability at least $1-\delta$, the following holds,*
*(1) for each $(i, \hat{f}_i) \in H$, $f_i^2 \ge \sum_{j=1}^n f_j^2/k - \epsilon^2\sum_{j=k+1}^n f_j^2$;*
*(2) if for any $i \in [n]$ and $f_i^2 \ge \sum_{j=1}^n f_j^2/k + \epsilon^2\sum_{j=k+1}^n f_j^2$, then $(i, \hat{f}_i) \in H$;*
*(3) for each $(i, \hat{f}_i) \in H$, $|\hat{f}_i - f_i| \le \epsilon\sqrt{\sum_{j=k+1}^n f_j^2}$.*
*The algorithm uses $O\left((k + \frac{1}{\epsilon^2})\log\frac{n}{\delta}\log m\right)$ bits of space, $O(\log n)$ update time and $O(k + 1/\epsilon^2)\mathrm{polylog}(n)$ query time.*

Thus, using `HEAVY-HITTER`, we are guaranteed that the error of the number of points in heavy cells is upper bounded by $\epsilon$ times the number of points in the non-heavy cells. The first heavy hitter algorithm that achieves an $l_2$ guarantee is by (Charikar et al., 2002), who has the same space and update time as that of the above algorithm. However the update time is slow, i.e. $O(n\log n)$ time to output the set of heavy hitters.

Lastly, we will use fully independent random hash function to sample the points. We will use Nissan's pseudorandom generator to de-randomize the hash functions by the method of (Indyk, 2000b). Our main theorem for this section is as follows. The formal proof of this theorem is postponed to Section B.

**Theorem 3.6** (Main Theorem)**.** *Fix $\epsilon, \rho \in (0, 1/2)$, positive integers $k$ and $\Delta$, Algorithm 1 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set $S$, such that with probability at least $1-\rho$, $S$ is an $\epsilon$-coreset for $k$-median of size $O\left(\frac{d^4L^4k}{\epsilon^2} + \frac{L^2k}{\rho}\right)$, where $L = \log\Delta$. The algorithm uses*

$$O\left[k\left(\left(\frac{d^7L^7}{\epsilon^2} + \frac{d^3L^5}{\rho}\right)\log\frac{dkL}{\rho\epsilon} + \frac{d^5L^6}{\epsilon^2\rho}\right)\right]$$

*bits in the worst case, processes each update in time $O\left(dL^2\log\frac{dkL}{\rho\epsilon}\right)$ and outputs the coreset in time $\mathrm{poly}(d, k, L, 1/\epsilon)$ after one pass of the stream.*

## 4 Positively Weighted Coreset

In this section, we will introduce a modification to our previous coreset construction, which leads to a coreset with all positively weighted points. The full algorithm and proofs are postponed to Section C. We present the main steps in this section.

The high level idea is as follows. When considering the estimate of the number of points in a cell, the estimate is only accurate when it truly contains a large number of points. However, in the construction of the previous section, we sample from each cell of each level, even though some of the cells contain a single point. For those cells, we cannot adjust their weights from negative to positive, since doing so would introduce large error. In this section, we introduce an ending level to each point. In other words, the number of

**Algorithm 1** $\texttt{CoreSet}(S, k, \rho, \epsilon)$: construct a $\epsilon$-coreset for dynamic stream $S$.

```
Initization:
Initialize a grid structure;
$O \leftarrow \{1, 2, 4, \ldots, \sqrt{d}\Delta^{d+1}\}$;
$L \leftarrow \lceil \log \Delta \rceil$;
$\pi_i(o) \leftarrow \min\left(\frac{3(L+1)^2\Delta d^2}{2^i\epsilon^2 o} \ln \frac{2\Delta^{kd}(L+1)}{\rho}, 1\right)$;
$K \leftarrow \frac{(2+e)(L+1)k}{\rho} + \frac{24d^4(L+1)^3k}{\epsilon^2} \ln \frac{1}{\rho}$,
$\epsilon' \leftarrow \left(\epsilon\sqrt{\frac{\rho}{8(2+e)^2kd^3(L+1)^3}}\right)$; $m \leftarrow 0$;
For each $o \in O$ and $i \in [0, L]$, construct fully independent hash function $h_{o,i} : [\Delta]^d \rightarrow \{0,1\}$ with $Pr_{h_{o,i}}(h_{o,i}[q] = 1) = \pi_i(o)$;
Initialize K-Set instances $\texttt{KS}_{o,i}$ with error probability $\rho/(L+1)$, size parameter $K$;
Initialize HEAVY-HITTER($\Delta^d$, $(e+2)(L+1)k/\rho$, $\epsilon', \rho/(L+1)$) instances, $\texttt{HH}_0, \texttt{HH}_1, \ldots, \texttt{HH}_L$, one for a level;
Update ($S$):
for each update $(op, q) \in S$:
    /*$op \in \{\texttt{Insert}, \texttt{Delete}\}$*/
    $m \leftarrow m \pm 1$;   /*Insert: +1, Delete: −1*/
    for each $i \in [0, L]$:
        $c_q^i \leftarrow$ the center of the cell contains $q$ at level $i$;
        $\texttt{HH}_i$.update($op, c_q^i$);
        for each $o \in [O]$:
            if $h_{o,i}(q) == 1$:
                $\texttt{KS}_{o,i}$.update($op, c_q^i$);
Query:
Let $o^*$ be the smallest $o$ such that no instance of $\texttt{KS}_{o,0}, \texttt{KS}_{o,1}, \ldots, \texttt{KS}_{o,L}$ returns Fail;
$R \leftarrow \{\}$;
for $i = -1$ to $L$:
    for each cell center $c$ in level $i$:
        Let $\mathcal{C}$ be the cell containing $c$;
        if $i$ = -1:
            $f \leftarrow m$;
        else:
            $f \leftarrow \texttt{GetFreq}(c, \texttt{HH}_i, \texttt{KS}_{o^*,i}, \pi_i(o^*))$;
        if $i < L$:
            $g \leftarrow \sum_{\mathcal{C}' \subset \mathcal{C}: \mathcal{C}' \in \mathcal{G}^{i+1}}$
              $\texttt{GetFreq}(c(\mathcal{C}'), \texttt{HH}_{i+1}, \texttt{KS}_{o^*i+1}, \pi_i(o^*))$;
            Assign weight $f - g$ to $c$;
            if $f - g \neq 0$:
                $R \leftarrow R \cup \{c\}$;
        else:
            Assign weight $f$ to $c$;
            if $f \neq 0$:
                $R \leftarrow R \cup \{c\}$;
return $R$.
```

points of a cell is estimated by sampling only if it contains many points. Thus, the estimates will be accurate enough and allow us to rectify the weights to be all positive.

**Algorithm 2** $\texttt{GetFreq}(e, \texttt{HH}, \texttt{KS}, \pi_i)$: retrieve the correct freuquency of cell center $e$, given the instance of HEAVY-HITTER and K-set.

```
$f_S(e) \leftarrow$ the frequency of $e$ returned by HH;
$f_K(e) \leftarrow$ the frequency of $e$ returned by KS;
$k' \leftarrow (e+2)(L+1)k/\rho$;
$F \leftarrow$ the set of top-$k'$ heavy hitters returned by HEAVY-HITTER;
if $e \in F$:
    return $f_S(e)$;
else:
    return $f_K(e)/\pi_i$.
```

## 4.1 Reformulation of the Telescope Sum

**Definition 4.1.** *A heavy cell identification scheme $\mathcal{H}$ is a map $\mathcal{H} : \mathcal{G} \rightarrow$ {heavy, non-heavy} such that, $h(\mathcal{C}_{-1})$ =heavy and for cell $\mathcal{C} \in \mathcal{G}_i$ for $i \in [0, L]$*

*1. if $|\mathcal{C}| \geq \frac{2^i\rho d\mathsf{OPT}}{k(L+1)\Delta}$ then $\mathcal{H}(\mathcal{C})$ = heavy;*

*2. If $\mathcal{H}(\mathcal{C})$ = non-heavy, then $\mathcal{H}(\mathcal{C}')$ = non-heavy for every subsell $\mathcal{C}'$ of $\mathcal{C}$.*

*3. For every cell $\mathcal{C}$ in level $L$, $\mathcal{H}(\mathcal{C})$ = non-heavy.*

*4. For each $i \in [0, L]$, $|\{\mathcal{C} \in \mathcal{G}_i : \mathcal{H}(\mathcal{C}) = \text{heavy}\}| \leq \frac{\lambda_1 kL}{\rho}$, where $\lambda_1 \leq 10$ is a positive universal constant.*

*The output for a cell not specified by the above conditions can be arbitrary. We call a cell* heavy *if it is identified heavy by $\mathcal{H}$. Note that a heavy cell does not necessarily contain a large number of points, but the total number of these cells is always bounded.*

In the sequel, heavy cells are defined by an arbitrary fixed identification scheme unless otherwise specified.

**Definition 4.2.** *Fix a heavy cell identification scheme $\mathcal{H}$. For level $i \in [-1, L]$, let $\mathcal{C}(p, i) \in \mathcal{G}_i$ be the cell in $\mathcal{G}_i$ containing $p$. The* ending level *$l(p)$ of a point $p \in P$ is the largest level $i$ such that $\mathcal{H}(\mathcal{C}(p, i))$ =heavy, and $\mathcal{H}(\mathcal{C}(p, i+1))$ =non-heavy.*

Note that the ending level is uniquely defined if a heavy cell identification scheme is fixed. We now rewrite the telescope sum for $p$ as follows,

$$p = \sum_{i=0}^{l(p)} \left(c_p^i - c_p^{i-1}\right) + c_p^L - c_p^{l(p)},$$

where $c_p^{-1} = \mathbf{0}$ and $c_p^L = p$. For arbitrary $k$-centers $Z \subset [\Delta]^d$, we write, $d(p, Z) = \sum_{i=0}^{l(p)} \left(d(c_p^i, Z) - d(c_p^{i-1}, Z)\right) + d(c_p^L, Z) - d(c_p^{l(p)}, Z) + d(\mathbf{0}, Z)$.

## 4.2 The New Construction (with arbitrary weights)

For these heavy cells, we use HEAVY-HITTER algorithms to obtain accurate estimates of the number of points in these cells, thus providing a *heavy cell identification scheme*. For the non-heavy cells, we only need to sample points from the bottom level, $\mathcal{G}_L$, but with a different probability for points with different ending levels.

We now describe the new construction. This essentially has the same gaurantee as the simpler construction from the previous section, however the benefit here is that (as shown in the next subsection) it can be modified to output only positive weights. In the following paragraph, the estimations $\widehat{|\mathcal{C}|}$ are given as a blackbox. In proposition C.9 we specify the conditions these estimations must satisfy.

**Non-Negatively Weighted Construction** Fix an arbitrary heavy cell identification scheme $\mathcal{H}$. Let $P_l$ be all the points with ending level $l(p) = l$. For each heavy cell $\mathcal{C}$, let $\widehat{|\mathcal{C}|}$ be an estimation of number of points of $|\mathcal{C}|$, we also call $\widehat{|\mathcal{C}|}$ the *value* of cell $\mathcal{C}$. For each non-heavy cell $\mathcal{C}'$, let $\widehat{|\mathcal{C}'|} = 0$. Let $S$ be a set samples of $P$ constructed as follows: $S = S_{-1} \cup S_0 \cup S_1, \cup \ldots \cup S_L$, where $S_l$ is a set of i.i.d samples from $P_l$ with probability $\pi_l$. Here $\pi_l$ for $l \in [-1, L]$ is redefined as $\pi_l =$

$$\min\left(\frac{\lambda_3 d^2 \Delta L^2}{2^l \epsilon^2 o} \log\left(\frac{2L\Delta^{dk}}{\rho}\right) + \frac{\lambda_4 d^2 k L^3 \Delta}{2^i \epsilon^2 \rho o} \log\frac{30kL^2}{\rho^2}, 1\right)$$

where $\lambda_3 > 0$ and $\lambda_4 > 0$ are universal constants. Our coreset $\mathcal{S}$ is composed by all the sampled points in $S$ and the cell centers of heavy cells, with each point $p$ assigned a weight $1/\pi_{l(p)}$ and for each cell center $c$ of a heavy cell $\mathcal{C} \in \mathcal{G}_i$, the weight is,

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}' : \mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i}. \tag{6}$$

For each non-heavy cell $\mathcal{C}$ except for those in the bottom level, $\mathrm{wt}(c(\mathcal{C})) = 0$. The weight of each point from $S$ is the value of the corresponding cell in the bottom level.

### 4.3 Ensuring Non-Negative Weights

We now provide a procedure to rectify all the weights for the coreset constructed in the last sub-section. The idea is similar to the method used in (Indyk & Price, 2011). The procedure is shown in Algorithm 4.3. After this procedure, there will be no negative weights in the coreset outputs.

**Algorithm 3** `RectifyWeights`$\left(\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|} \ldots, \widehat{|\mathcal{C}_{k'}|}, S\right)$: input the estimates of number of points in each cell and the weighted sampled points, output a weighted coreset with non-negative weights.

```
for $i = -1$ to $L$:
    for each heavy cell $\mathcal{C}$ center in $\mathcal{G}_i$:
        if $\mathrm{wt}(\mathcal{C}) < 0$:
            Decrease the value of the children heavy cells in level $\mathcal{G}_{i+1}$ and sampled points $S_i$ arbitrarily by total $|\mathrm{wt}(\mathcal{C})|$ amount, such that for each children cell $\mathcal{C}' \in \mathcal{G}_{i+1}$, $\widehat{|\mathcal{C}'|}$ is non-negative, and for each sampled point $p \in S_i$, the weight is non-negative.
return Rectified Coreset
```

**Theorem 4.3.** *Fix $\epsilon, \rho \in (0, 1/2)$, positive integers $k$ and $\Delta$, Algorithm 6 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set $S$ with* non-negative weights *for each point, such that with probability at least $0.99$, $S$ is an $\epsilon$-coreset for $k$-median of size*

$$O\left[\frac{d^3 L^4 k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$$

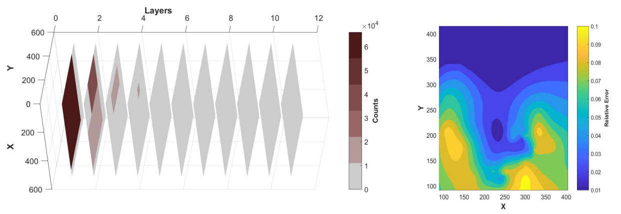
*where $L = \log\Delta$. The algorithm uses*

$$O\left[\frac{d^7 L^7 k}{\epsilon^2}\left(\rho d L + \frac{L}{\rho}\log^2\frac{dkL}{\rho\epsilon}\right)\log^2\frac{dkL}{\rho\epsilon}\right]$$

*bits in the worst case. For each update of the input, the algorithm needs* $\mathrm{poly}(d, 1/\epsilon, L, \log k)$ *time to process and outputs the coreset in time* $\mathrm{poly}(d, k, L, 1/\epsilon, 1/\rho, \log k)$ *after one pass of the stream.*

## 5 Experiments

We illustrate our construction using an offline construction on Gaussian mixture data in $\mathbb{R}^2$. As shown in Figure 2 in Section D, we randomly generated 65536 points from $\mathbb{R}^2$, then rounded the points to a grid of size $\Delta = 512$. Our coreset uses $\log_2\Delta + 2 = 11$ levels of grids. The storage in each level is very sparse. As shown in Figure 1(a), only 90 points are stored in total. We compared the 1-median costs estimated using the coreset and the dataset, the resulting difference is very small, as illustrated in Figure 1(b).



(a) (b)

*Figure 1.* (a) The layer structure of the coreset. Cells with more weight are shaded darker. (b) The relative error of a 1-median cost function. Using only 90 points, the global maximum error was under 10%.

## 6 Concluding Remark

We develop algorithms that make a single pass over the dynamic stream and output, with high probability, a coreset for the original $k$-median problem. Both the space complexity and the size of the coreset are polynomially dependent on $d$, whereas the only previous known bounds are exponential in $d$. We constructed our coreset for the possible solutions in discrete space $[\Delta]^d$, but it is easy to modify the coreset to be a coreset in continuous space $[0, \Delta]^d$ (note that we still require the input dataset to be from a discrete space). The way to do this is by modifying the sampling probability $\pi_i$ in the algorithm, i.e. replacing the factor of $\ln(\Omega(\Delta^{kd}L/\rho))$ to $\ln(\Omega((\Delta/\epsilon)^{kd}L/\rho))$. Then any $k$-set from $[0, \Delta]^d$ can be rounded to the closest $k$-set in $[\Delta/\epsilon]^d$ and the cost only differs by a $(1 \pm \epsilon)$ factor while the space bound changes only by a $\mathrm{polylog}(1/\epsilon)$ factor. Lastly, we remark that the coreset scheme can be easily modified to other metric spaces, e.g. the $l_p$ metric. The space bound depends on the doubling dimension of the metric.

As shown in our experiments, a 2D implementation using our framework is very efficient. We believe that a high-dimensional implementation will be efficient as well. We leave the full implementation as a future project.

## Acknowledgment

V. Braverman is supported by the NSF Grants IIS-1447639, EAGER CCF- 1650041, and CAREER CCF-1652257. H. Lang is supported by the Franco-American Fulbright Commission. H. Lang thanks INRIA (l'Institut national de recherche en informatique et en automatique) for hosting him during the writing of this paper. C. Sohler is Supported by DFG within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A2. L. Yang is supported by the NSF Grant IIS-1447639.

## References

Bagchi, Amitabha, Chaudhary, Amitabh, Eppstein, David, and Goodrich, Michael T. Deterministic sampling and range counting in geometric data streams. *ACM Transactions on Algorithms (TALG)*, 3(2):16, 2007.

Chan, Timothy M. Faster core-set constructions and data stream algorithms in fixed dimensions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 152–159. ACM, 2004.

Chan, Timothy M and Sadjad, Bashir S. Geometric optimization problems over sliding windows. *International Journal of Computational Geometry & Applications*, 16 (02n03):145–157, 2006.

Charikar, Moses, Chekuri, Chandra, Feder, Tomás, and Motwani, Rajeev. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 626–635. ACM, 1997.

Charikar, Moses, Chen, Kevin, and Farach-Colton, Martin. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pp. 693–703. Springer, 2002.

Charikar, Moses, O'Callaghan, Liadan, and Panigrahy, Rina. Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pp. 30–39. ACM, 2003.

Chen, Ke. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009. doi: 10.1137/070699007. URL http://dx.doi.org/10.1137/070699007.

Cormode, Graham and Muthukrishnan, S. Radial histograms for spatial streams. *DIMACS Technical Report*, 11, 2003.

Feigenbaum, Joan, Kannan, Sampath, and Zhang, Jian. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41(1):25–41, 2005.

Feldman, Dan and Langberg, Michael. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pp. 569–578, 2011. doi: 10.1145/1993636.1993712. URL http://doi.acm.org/10.1145/1993636.1993712.

Feldman, Dan, Monemizadeh, Morteza, and Sohler, Christian. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pp. 11–18, 2007. doi: 10.1145/1247069.1247072. URL http://doi.acm.org/10.1145/1247069.1247072.

Feldman, Dan, Schmidt, Melanie, and Sohler, Christian. Turning big data into tiny data: Constant-size coresets for *k*-means, PCA and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pp. 1434–1453, 2013. doi: 10.1137/1.9781611973105.103. URL http://dx.doi.org/10.1137/1.9781611973105.103.

Frahling, Gereon and Sohler, Christian. Coresets in dynamic geometric data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp. 209–217. ACM, 2005.

Ganguly, Sumit. Counting distinct items over update streams. In *International Symposium on Algorithms and Computation*, pp. 505–514. Springer, 2005.

Guha, Sudipto, Mishra, Nina, Motwani, Rajeev, and O'Callaghan, Liadan. Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on*, pp. 359–366. IEEE, 2000.

Har-Peled, Sariel and Kushal, Akash. Smaller coresets for k-median and k-means clustering. In *Proceedings of the 21st ACM Symposium on Computational Geometry, Pisa, Italy, June 6-8, 2005*, pp. 126–134, 2005. doi: 10.1145/1064092.1064114. URL http://doi.acm.org/10.1145/1064092.1064114.

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

Hershberger, John and Suri, Subhash. Adaptive sampling for geometric problems over data streams. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 252–262. ACM, 2004.

Indyk, Piotr. *High-dimensional computational geometry*. PhD thesis, Citeseer, 2000a.

Indyk, Piotr. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pp. 189–197. IEEE, 2000b.

Indyk, Piotr. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 539–545. Society for Industrial and Applied Mathematics, 2003.

Indyk, Piotr. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 373–380. ACM, 2004.

Indyk, Piotr and Price, Eric. K-median clustering, model-based compressive sensing, and sparse recovery for earth mover distance. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 627–636. ACM, 2011.

Larsen, Kasper Green, Nelson, Jelani, Nguyên, Huy L, and Thorup, Mikkel. Heavy hitters via cluster-preserving clustering. *arXiv preprint arXiv:1604.01357*, 2016.

Liu, Zaoxing, Ivkin, Nikita, Yang, Lin, Neyrinck, Mark, Lemson, Gerard, Szalay, Alexander, Braverman, Vladimir, Budavari, Tamas, Burns, Randal, and Wang, Xin. Streaming algorithms for halo finders. In *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pp. 342–351. IEEE, 2015.

Meyerson, Adam. Online facility location. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pp. 426–431. IEEE, 2001.

Muthukrishnan, Shanmugavelayutham. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

Nisan, Noam. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.