STREAMING CORESETS FOR HIGH-DIMENSIONAL GEOMETRY

by
Harry Lang

A dissertation submitted to Johns Hopkins University in conformity
with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
May 2018

Abstract

This thesis studies clustering problems on data streams, specifically with applications to metric spaces that are either non-Euclidean or of high-dimensionality. The algorithms are introduced to approximate minima of the k-median function, although they can be used to find approximate minima of many other functions are shown in Part I.

In Part I, we introduce the first analysis of the effect of stream order on the space required to solve k-median. We use this to forge a connection to random-order streams and provide an optimal $O(nk)$-time algorithm for k-median in the RAM model. In Part II, we provide a nearly optimal algorithm to maintain a coreset for k-median. Specifically for the case of k-means, we introduce the first coreset to be simultaneously independent of both the dimension and the input size. This gives the first sublinear size coreset for high-dimensional sparse data. In Part III, we apply our techniques to streams where points may be deleted as well as inserted, and construct the first coreset for high-dimensional spaces.

List of Readers/Advisors in Alphabetical Order

Raman Aurora

Vladimir Braverman

Fei Lu

Mauro Maggioni

Alex Szalay

Acknowledgements

Research in this thesis includes work conducted jointly with Vladimir Braverman[1] and Dan Feldman.

This research is supported by the Franco-American Fulbright Commission. The author thanks INRIA (l'Institut national de recherche en informatique et en automatique) for hosting him during the writing of many of the results in this thesis.

[1]This material is based upon work supported in part by the National Science Foundation under Grants No. 1447639, 1650041 and 1652257, Cisco faculty award, and by the ONR Award N00014-18-1-2364.

Table of Contents

List of Tables

List of Figures

Introduction

This thesis is comprised of three main parts/chapters, which are now summarized.

Part I: In the streaming model, the order of the stream can significantly affect the difficulty of a problem. A t-semirandom stream was introduced as an interpolation between random-order ($t = 1$) and adversarial-order ($t = n$) streams where an adversary intercepts a random-order stream and can delay up to t elements at a time. IITK Sublinear Open Problem #15 asks to find algorithms whose performance degrades smoothly as t increases. We show that the celebrated online facility location algorithm achieves an expected competitive ratio of $O(\frac{\log t}{\log\log t})$. We present a matching lower bound that any randomized algorithm has an expected competitive ratio of $\Omega(\frac{\log t}{\log\log t})$.

We use this result to construct an $O(1)$-approximate streaming algorithm for k-median clustering that stores $O(k \log t)$ points and has $O(k \log t)$ worst-case update time. Our technique generalizes to any dissimilarity measure that satisfies a weak triangle inequality, including k-means, M-estimators, and ℓ_p norms. The special case $t = 1$ yields an optimal $O(k)$ space algorithm for random-order streams as well as an optimal $O(nk)$ time algorithm in the RAM model, closing a long line of research on this problem.

Part II: Let P be a set (called points), Q be a set (called queries) and a function $f : P \times Q \to [0, \infty)$ (called cost). For an error parameter $\epsilon > 0$, a set $S \subseteq P$ with a *weight function* $w : P \to [0, \infty)$ is an ϵ-coreset if $\sum_{s\in S} w(s)f(s, q)$ approximates $\sum_{p\in P} f(p, q)$ up to a multiplicative factor of $1 \pm \epsilon$ for every given query $q \in Q$. Coresets are used to solve fundamental problems in machine learning of streaming and distributed data.

We construct coresets for the k-means clustering of n input points, both in an arbitrary metric space and d-dimensional Euclidean space. For Euclidean space, we present the first coreset whose size is simultaneously independent of both d and n. In particular, this is the first coreset of size $o(n)$ for a stream of n sparse points in a $d \geq n$ dimensional space (e.g. adjacency matrices of graphs). We also provide the first generalizations of such coresets for handling outliers. For arbitrary metric spaces, we improve the dependence on k to $k \log k$ and present a matching lower bound.

For M-estimator clustering (special cases include the well-known k-median and k-

means clustering), we introduce a new technique for converting an offline coreset construction to the streaming setting. Our method yields streaming coreset algorithms requiring the storage of $O(S + k \log n)$ points, where S is the size of the offline coreset. In comparison, the previous state-of-the-art was the merge-and-reduce technique that required $O(S \log^{2a+1} n)$ points, where a is the exponent in the offline construction's dependence on ϵ^{-1}. For example, combining our offline and streaming results, we produce a streaming metric k-means coreset algorithm using $O(\epsilon^{-2} k \log k \log n)$ points of storage. The previous state-of-the-art required $O(\epsilon^{-4} k \log k \log^6 n)$ points.

Part III: 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 $\{1, 2, \dots \Delta\}^d$. Our algorithms use $k\epsilon^{-2}\text{poly}(d \log \Delta)$ 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.

Part I

Clustering Under Different Stream Orders

1.1 Introduction

One of the fundamental theoretical questions in the streaming model is to understand how the stream order impacts computation. In the adversarial-order model, results must hold under any order, whereas in the random-order model the order is selected uniformly at random. The order of the stream can strongly affect the resources required to solve a problem. For example, for streams of n integers where the stream may be read sequentially multiple times, determining the median using polylogarithmic space requires $\Theta(\frac{\log n}{\log\log n})$ passes in adversarial-order [32, 17] but only $\Theta(\log\log n)$ passes in random-order [16, 7].

As demonstrated by the median problem, there can be an exponential gap in the resources required for random-order and adversarial-order streams. To interpolate between these two extremes, Guha and McGregor introduced two notions of *semirandom-order* where the adversary has limited power.

Definition 1 (t-semirandom order, [16])**.** *A t-bounded adversary*[1] *can permute a stream $p_1, \dots, p_n$ to the stream $p_{\sigma(1)}, \dots, p_{\sigma(n)}$ with any permutation σ that satisfies $|\{j \in [n] : j < i \text{ and } \sigma(j) > \sigma(i)\}| < t$ for every $i \in [n]$. A stream is in t-semirandom order if it is generated by a t-bounded adversary acting on a random-order stream.*

Definition 2 (ϵ-generated random order, [17])**.** *Let μ be the uniform distribution over all permutations of n elements. A stream of n points arrives in ϵ-generated random-order if the permutation is drawn from a distribution ν such that $||\mu - \nu||_1 \leq 2\epsilon$.*

These models capture the notion of an adversary with limited power and establish a spectrum of semirandom orders to intermediate between the fully random and fully adversarial cases. IITK Sublinear Open Problem #15 [26] asks:

> How do these notions relate to each other? Can we develop algorithms whose performance degrades smoothly as the stream ordering becomes "less-

[1]Our definition of t-bounded actually corresponds to $(t-1)$-bounded as introduced in [16]. This turns out to be more natural and avoids writing $t+1$ in all our bounds.

random" using either definition? For a given application, which notion is more appropriate?

We respond to the first question by showing that no non-trivial relations hold between these models. One can verify that $\epsilon = 0$ and $t = 0$ correspond to random-order, and that $\epsilon = 1$ and $t = n$ correspond to adversarial-order. However, we show that these models are incomparable in the sense that an ϵ-generated adversary requires $\epsilon > 1 - 2^{-\Omega(n)}$ to simulate the action of a t-bounded adversary for any $t > 1$, and that a t-bounded adversary requires $t > n/2$ to simulate the action of an ϵ-generated adversary for any $\epsilon \geq 2^{-\Omega(n)}$.

We answer the second question by proving matching bounds for the online facility location problem that show the performance degrades smoothly as t increases. These are the first bounds for semirandom streams that match at all values of t. Previous results matched only for t sufficiently small. For example, the result of [16] shows how to return the median in $O(\log \log n)$ passes when $t = O(\sqrt{n})$. However, for a constant $c < 1$ the algorithm is only guaranteed to terminate in $O(n)$ passes when $t = \Omega(n^c)$. In comparison, even at $t = n$ there are polylogarithmic-space algorithms that return the median in only $O(\log n / \log \log n)$ passes [32].

Our results provide evidence that t-bounded adversarial-order is a viable model of semirandomness. We address the third question by complementing our positive results for the t-semirandom model with an argument showing that the ϵ-generated random order model is uninteresting for a wide class of problems. A more complete discussion of IITK Open Question #15 is included in Section 1.6.

1.1.1 Our Contributions

We present results for online facility location and a large class of clustering problems. In Section 1.3, we provide a novel analysis that the online facility location algorithm of [30] is $O(\frac{\log t}{\log \log t})$-competitive in expectation. Adapting Meyerson's original argument for a t-bounded adversary is possible but results in an $O(t)$ expected competitive ratio. We introduce a different analysis that permits this exponential improvement. We complement this result by presenting a matching lower bound in Section 1.4 that any randomized algorithm for online facility location is $\Omega(\frac{\log t}{\log \log t})$-competitive in expectation. See Table 1.1 for a comparison with existing results.

In Section 1.5, we present a streaming algorithm for clustering using any function that satisfies a weak triangle inequality (this includes k-median, k-means, M-estimators, and ℓ_p norms). Our algorithm stores $O(k \log t)$ points and has $O(k \log t)$ worst-case update time. As shown in Table 1.2, we match the state-of-the-art for adversarial-order streams and provide the first results for all $t < n$. We remark that our algorithm respects sparsity by only storing a weighted subset of the input. Another notable property of our clustering algorithm is that it is oblivious to the actual values of k and t. The algorithm takes an input value m, and the output is valid as long as $m = \Omega(k \log t)$. This may be useful for practical applications where the number of clusters or power of the adversary is unknown. For example, if the data exhibits a hierarchical structure than the resolution of the result (measured by k) degrades smoothly as the power of the adversary increases.

The special case $t = 1$ yields the first result for clustering on random-order streams. In the `RAM` model where we can shuffle the input into random order in linear time, this implies an optimal $O(nk)$ time algorithm and closes a long line of research on the problem.

As a blackbox used by our clustering algorithm, we present a method to compress a weighted set of n distinct points to a weighted set of $\frac{n+k}{2}$ distinct points in linear time while incurring less than twice the optimal cost of clustering to k points. Our algorithm, based on 2-coloring a nearest neighbor graph, is presented in Section 1.5.2 as it may be of independent interest.

Regime	Upper Bound	Source	Lower Bound	Source
$t = 1$	$O(1)$	[30]	$\Omega(1)$	Trivial
$1 \leq t \leq n$	$O\left(\frac{\log t}{\log \log t}\right)$	$\star\star$	$\Omega\left(\frac{\log t}{\log \log t}\right)$	$\star\star$
$t = n$	$O\left(\frac{\log n}{\log \log n}\right)$	[14]	$\Omega\left(\frac{\log n}{\log \log n}\right)$	[14]

Table 1.1: Expected competitive ratio for online facility location. Upper bounds apply to the algorithm of [30]. Lower bounds apply to any randomized algorithm.

Regime	Space	Source
$t = 1$	$O(k)$	$\star\star$
$1 \leq t \leq n$	$O(k \log t)$	$\star\star$
$t = n$	$O(k \log n)$	[14]

Table 1.2: Space complexity (measured in weighted points) of algorithms for metric k-median and k-means in the streaming model.

1.1.2 Prior Work

Random-Order Streams: There has been an increasing interest to design algorithms for data streams that arrive in random-order, and in recent years the model has become quite popular. Random-order streams have been considered for problems including rank selection [32, 19, 27], frequency moments [3], entropy [20], submodular maximization [31], and graph matching [25, 24, 12]. Lower bounds that hold even under the assumption of random-order have been developed using multi-party communication complexity [8, 6, 7, 15]. Semirandom-order streams, in both the t-bounded and ϵ-generated models, have been considered for rank selection [16, 17]. The stochastic streaming model, which takes the random-order assumption a step further by assuming that stream elements are independent samples from an unknown distribution, has also attracted attention [18, 33, 11]. The stochastic streaming model is strictly easier than the random-order model since any stochastic stream is automatically in random-order.

Online Facility Location: The study of online facility location was initiated by Meyerson [30]. He provided a simple randomized algorithm and proved that for random-order streams it is $O(1)$-competitive in expectation. Later, Fotakis [14] showed that for adversarial-order streams any randomized algorithm has an expected competitive ratio of $\Omega(\log n / \log \log n)$ and proved that Meyerson's randomized algorithm achieves this bound; he also presented a novel deterministic algorithm that achieves this bound. For Euclidean space, a simple and practical deterministic algorithm was provided by [2].

Streaming Metric k-median and k-means Clustering: The streaming k-median and k-means problems have only been considered in the adversarial-order model. These problems are well-studied; here we mention only the metric space results that achieved an improvement in the space bound over the previous state-of-the-art. The first streaming solution computed a $2^{O(1/r)}$-approximation for any $r \in (0,1)$ and stored $O(n^r/r)$ points [21]. Later, an algorithm storing only $O(k \log^2 n)$ points was provided [9]. The current state-of-the-art $O(1)$-approximation stores $O(k \log n)$ points [5]. A variety of other results are known for Euclidean space.

Runtime	Source
$O(n^2 \log n)$	[23]
$O(nk \log k)$	[22]
$O(n^2)$	[28]
$O(nk + n \log n + k^2 \log^2 n)$	[29]
$O(nk + n^{1/2} k^{3/2} \log^2 n \log^{3/2} k)$	[10]
$O(nk)$	$\star\star$
$\Omega(nk)$	[29]

Table 1.3: Results for metric k-median in the RAM model

RAM-Model Metric k-median and k-means Clustering: The history of fast $O(1)$-approximations[2] in the RAM model is summarized in Table 1.3, omitting results that do not improve the runtime for any value of k. These results for k-median generalize to k-means with a larger constant in the approximation ratio. We conclude this line of research with an optimal $O(nk)$ time algorithm, matching the $\Omega(nk)$ time lower bound for any randomized algorithm [29]. We remark that there exists an $O(nk)$ time algorithm for k-means in Euclidean space [1], but it relies on the principal axis theorem and therefore does not generalize to k-median or to other metric spaces.

[2]Most of the results shown in Table 1.3 actually output $O(k)$ centers instead of exactly k. However, we observe that the result of [28] implies that any solution of $O(k)$ centers can be converted to a solution of exactly k centers in $O(k^2)$ time.

1.2 Preliminaries

Let $(\mathcal{X}, d)$ be a metric space. In the facility location problem with parameter $f > 0$ (called the facility cost), we are given a set[3] $A \subset \mathcal{X}$ called *demands*. The problem is to compute a set $B \subset \mathcal{X}$ called *facilities* and to connect each demand to a facility. To connect demand a to facility b, we incur cost $d(a, b)$. We also incur cost f for each facility opened. The objective is to compute B such that the total cost is minimized. Defining $\mathsf{COST}(A, B) = \sum_{a \in A} \min_{b \in B} d(a, b)$, the total cost is $|B|f + \mathsf{COST}(A, B)$ by connecting each demand to the nearest facility.

In online facility location, we receive A as a stream of points. When point p arrives, we may open a facility (incurring facility cost f) and then must connect p to a facility (incurring connection cost). Observe that if we open a facility at the location of p, there is no connection cost. The problem is online because the decisions to open a facility and connect p are irrevocable, meaning that a facility can never be closed and that p cannot be reconnected if a closer facility opens later.

For the k-median problem, there is no facility cost but the number of facilities (here called *centers*) is fixed at k. The goal is to compute a set B that minimizes the total cost $\mathsf{COST}(A, B)$. When the input arrives as a stream, we seek to design algorithms that require a minimal amount of memory.

Definition 3 (Optimal Cost)**.** *Let A be a set and let $k \geq 1$. $\mathsf{OPT}_k(A)$ is defined as the minimum of $\mathsf{COST}(A, B)$ where $B \subset \mathcal{X}$ ranges over all sets of k points.*

The optimal cost for k-median is $\mathsf{OPT}_k(A)$. For facility location, the optimal cost is the minimum $kf + \mathsf{OPT}_k(A)$ where k ranges over all positive integers. An α-approximation is a solution with cost at most α times the optimum.

A t-semirandom stream is the result of a random-order stream that has been intercepted by a t-bounded adversary (see Definition 1). Imagine that the stream of elements is a deck of cards, initially shuffled into random order. The adversary draws cards into his hand

[3]We use the word "set" to actually mean "multiset". Multisets may contain multiple copies of the same element.

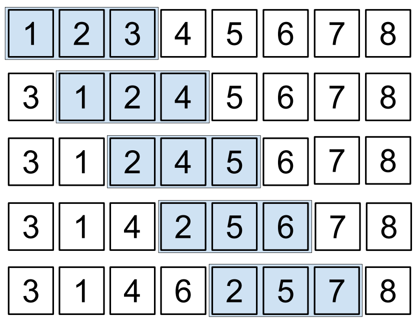


Figure 1.1: Example of a 3-bounded adversary acting on a stream. Each step is shown on a different row. The shaded box represents the memory of the adversary. The adversary must send an element to the algorithm before receiving the next element.

from the deck. He may give any card from his hand to the algorithm. The restriction is that he can have at most t cards in his hand at any time. This means that if he has a full hand of t cards, he cannot draw a new card until giving one to the algorithm. See Figure 1.1 for an example.

1.3 Online Facility Location

The algorithm of Meyerson [30] is simple and elegant. Let $f > 0$ be the facility cost parameter. When a point p arrives, let $\delta(p)$ be the distance between p and the nearest facility. With probability $\min(1, \delta(p)/f)$, we open a facility at p and pay facility cost f. Otherwise, we connect p to the nearest facility and pay connection cost $\delta(p)$. We write $\mathtt{OFL}$ to refer to this algorithm.

Theorem 1. *Let $r > 1$ and $h \in \mathbb{N}$ such that $r^h \geq 4t$, and let k be a positive integer. If $\mathtt{OFL}(f)$ runs on a t-semirandom stream S, the facility cost and connection cost incurred by $\mathtt{OFL}(f)$ are each less than $(r+3)\mathsf{OPT}_k(S) + (h+2)fk$ in expectation.*

Proof. Partition S into k optimal clusters $\{X_\ell\}_{\ell\in[k]}$ so that $\mathsf{OPT}_k(S) = \sum_{\ell\in[k]} \mathsf{OPT}_1(X_\ell)$. Consider the centers $\{c_\ell\}_{\ell\in[k]}$ such that $\mathsf{COST}(X_\ell, c_\ell) = \mathsf{OPT}_1(X_\ell)$. Lemmas 1, 2, and 6 bound the expected connection cost and expected facility cost on each X_ℓ by $(r + 3)\mathsf{OPT}_1(X_\ell) + (h+2)f$. We obtain the result by summing over all $\ell \in [k]$. $\square$

We now provide the results used in the proof of Theorem 1. Let $X \subset S$ be a set of n points. Given a center point c, define $A = \mathsf{COST}(X, c)$ and $a = A/n$. We partition X into three pieces as follows:

$$G = \{x \in X : d(x,c) \leq 2a\}$$

$$M = \{x \in X : 2a < d(x,c) \leq 2r^h a\}$$

$$B = \{x \in X : d(x,c) > 2r^h a\}$$

Defining $A_Z = \texttt{COST}(Z,c)$ for any set Z, we decompose $A = A_G + A_M + A_B$.

Fact 1. *In any order, the expected connection cost of a set Z incurred before a facility opens in Z is less than f.*

Proof. For convenience we normalize to $f = 1$. Let $(z_1, \ldots, z_m)$ be the order of Z. Define $x_j = \min(1, \delta(z_j))$. There is a probability of x_j that point z_j opens as a facility; otherwise z_j incurs connection cost x_j.

Let E_i denote the expected connection cost before a facility is opened when `OFL` is run on the suffix $(z_i, z_{i+1}, \ldots, z_m)$. For $1 \leq i \leq j \leq m$ define $P_i^j = \prod_{\ell=i}^{j}(1 - x_\ell)$. Observe that $E_i = \sum_{j=i}^{m} x_j P_i^j$. We seek to prove that $E_1 < 1$.

We write the recursive formula $E_i = (1 - x_i)(x_i + E_{i+1})$. Observe that $E_m \leq 1/4$, with the maximum occurring when $x_m = 1/2$. Assuming inductively that $E_{i+1} < 1$, observe that $E_i = (1 - x_i)(x_i + E_{i+1}) < 1 - x_i^2 \leq 1$. We conclude that $E_1 < 1$. □

Observe that $\delta(p)$ can be used to simultaneously bound both the expected connection cost and the expected facility cost. We use this in Lemmas 1-6 to bound both types of cost with the same argument.

Lemma 1. *In any order, the expected connection cost and expected facility cost of G are each at most $f + A_G + 2a|G|$.*

Proof. By Fact 1, the expected connection cost of points in G before a facility opens in G is less than f. The facility cost is exactly f when the first facility opens in G. After a facility has opened in G, we may bound $\delta(g) \leq d(g,c) + 2a$ for any $g \in G$ by the triangle inequality. The result follows by summing over all $g \in G$. □

The proof of the next lemma is similar to the previous.

Lemma 2. *In any order, the expected connection cost and expected facility cost of M are each at most $hf + (r+1)A_M$.*

Proof. We partition M into h parts, defining $M_j = \{x \in X : 2r^{j-1}a < d(x,c) \leq 2r^j a\}$ for $j \in \{1, \ldots, h\}$. By Fact 1, the expected connection cost of points in M_j before a facility opens in M_j is less than f. The facility cost of M_j is exactly f when the first facility opens in M_j. After a facility opens in M_j, we may bound $\delta(m) \leq d(m,c) + 2r^j a \leq (r+1)d(m,c)$ for each $m \in M_j$ by the triangle inequality. The result follows by summing over all $m \in M$. □

Fixing an order, let α'_i be the expected connection cost of the ith point of G. For $i \in \{1, \ldots, |G|\}$, define $\alpha_i = \min_{1 \leq j \leq i} \alpha'_j$. For $i \in \{0, \ldots, |G|\}$, let β_i be the number of points in B that arrive after exactly i points of G have arrived.

Lemma 3. *In any order, the expected connection cost and expected facility cost of B are each at most $A_B + 2a|B| + f\beta_0 + \sum_{i=1}^{|G|} \alpha_i \beta_i$.*

Proof. The first β_0 points of B incur cost at most $f\beta_0$ trivially. If a point b arrives after a point g, then $\delta(b) \leq d(b,c) + 2a + \delta(g)$ by the triangle inequality. Hence for a point b that arrives after i points of G, taking expectations and then the minimum over the i preceding points of G shows that $E[\delta(b)] \leq d(b,c) + 2a + \alpha_i$. The result follows by summing over all $b \in B$. □

The remainder of the proof will depend on the assumption of t-semirandom order. For precision, we continue to use $E[\cdot]$ for expectation over the randomness used by `OFL` and introduce $\mathcal{E}[\cdot]$ for expectation over the randomness of the stream order. Let β'_i be the number of points in B that occur after exactly i points of G in the initial random-order stream before being intercepted by the adversary. Since the adversary may hold at most t points, if the adversary has received $t+i$ points of G then the algorithm has received at least i points of G. This provides the relation $\beta_0 + \ldots + \beta_i \leq \beta'_0 + \ldots + \beta'_{i+t-1}$ for every $i \geq 0$. We can view β'_i as the number of balls in the ith bin when we randomly drop $|B|$ balls into $|G|+1$ bins.

Observe that $f > \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_{|G|}$. The adversary's optimal strategy against our bound in Lemma 3 is to delay points in G as long as possible. We therefore identify the worst-case bound $\beta_0 = \beta'_0 + \ldots + \beta'_{t-1}$ and $\beta_i = \beta'_{i+t-1}$ for every $i \geq 1$. We rewrite $f\beta_0 + \sum_{i=1}^{|G|} \alpha_i \beta_i = f \sum_{j=0}^{t-1} \beta'_j + \sum_{i=1}^{|G|} \alpha_i \beta'_{i+t-1}$.

The difficulty is that α_i and β_i are dependent random variables. Although they cannot affect each other directly, both α_i and β_i depend on the prefix of the stream ending on the i^{th} point of G. To overcome this, we split the sum into two pieces and for each piece we find an upper bound on either α_i or β_i that holds independently of the prefix.

The next lemma quantifies the intuition that if many points of G have arrived then there must be a facility very close to G. This bound suffices after a constant fraction of G has arrived.

Lemma 4. *In t-semirandom order,* $\sum_{i=\lceil |G|/2 \rceil + 1}^{|G|} \mathcal{E}[\alpha_i \beta_i] < f \ln(2)/4t$.

Proof. Let $\mu(g)$ be the indicator random variable that no facility is open in G after g is processed. Let G' be the set of i points of G that have arrived. Fact 1 implies that $\sum_{g \in G} E[\delta(g)\mu(g)] < f$. Thus there must be some $g \in G'$ such that $E[\delta(g)\mu(g)] < f/i$. Since $\alpha_i = \min_{g \in G'} E[\delta(g)\mu(g)]$, we bound $\alpha_i < f/i$.

By Markov's inequality $|B| < \frac{n}{2r^h} \leq \frac{n}{8t}$ and $|G| \geq \lceil \frac{n}{2} \rceil$. Then $\mathcal{E}[\beta_i] = \mathcal{E}[\beta'_{i+t-1}] = \frac{|B|}{|G|+1} < \frac{1}{4t}$ which implies $\mathcal{E}[\alpha_i \beta_i] < \mathcal{E}[(f/i)\beta_i] < \frac{f}{4ti}$. The result follows since $\sum_{i=\lceil |G|/2 \rceil + 1}^{|G|} \frac{1}{i} < \ln(2)$. $\square$

If we drop $|B|$ balls into $|G|+1$ bins and condition on the number of balls in i of the bins, the expected number of balls in any other bin is at most that of dropping $|B|$ balls into $|G| - i + 1$ bins. This bound blows up towards the end of the stream but suffices for the first half.

Lemma 5. *In t-semirandom order,* $\sum_{i=1}^{\lceil |G|/2 \rceil} \mathcal{E}[\alpha_i \beta_i] < f \ln(3)/2t$

Proof. Let $P(i)$ denote the order of the prefix of the stream ending on the i^{th} point of G. Observe that $\mathcal{E}[\beta'_i | P(i)] \leq \frac{|B|}{|G|-i+1}$ with the maximum occurring when $\beta'_j = 0$ for all $j \in \{0, \ldots, i-1\}$. This implies $h(i) := \max_{P(i)} \mathcal{E}[\beta_i | P(i)] \leq \frac{|B|}{|G|-t-i+2}$.

By causality, points that arrive after the i^{th} point of G do not affect α_i. Therefore α_i is a constant when $P(i)$ is fixed. We can now separate

$$\begin{aligned}\mathcal{E}[\alpha_i\beta_i] &= \mathcal{E}_{P(i)}[\mathcal{E}[\alpha_i\beta_i|P(i)]]\\ &= \mathcal{E}_{P(i)}[\mathcal{E}[\alpha_i|P(i)]\cdot\mathcal{E}[\beta_i|P(i)]]\\ &\le \mathcal{E}_{P(i)}[\mathcal{E}[\alpha_i|P(i)]\cdot h(i)]\\ &= h(i)\mathcal{E}_{P(i)}[\mathcal{E}[\alpha_i|P(i)]]\\ &= h(i)\mathcal{E}[\alpha_i]\end{aligned}$$

We have the restraint $\sum_{i=1}^{|G|}\mathcal{E}[\alpha_i] < f$ by Fact 1. Since $h(i)$ is increasing, the sum $\sum_{i=1}^{\lceil|G|/2\rceil} h(i)\mathcal{E}[\alpha_i]$ is maximized when $\alpha_i = \frac{f}{\lceil|G|/2\rceil}$ for $i \le \lceil|G|/2\rceil$. We bound $\sum_{i=1}^{\lceil|G|/2\rceil}\mathcal{E}[\alpha_i\beta_i] \le \sum_{i=1}^{\lceil|G|/2\rceil}\frac{f}{\lceil|G|/2\rceil}\frac{|B|}{|G|-t-i+2}$. By Markov's inequality $\frac{|B|}{\lceil|G|/2\rceil} < \frac{1}{2t}$. We may assume that $t < \frac{n}{8}$ since otherwise $B = \varnothing$ and there is nothing to show. We conclude by $\sum_{i=1}^{\lceil|G|/2\rceil}\frac{1}{|G|-t-i+2} = \sum_{y=\lfloor|G|/2\rfloor-t+2}^{|G|-t+1}\frac{1}{y} < \ln\left(\frac{|G|-t+1}{\lfloor|G|/2\rfloor-t+1}\right) < \ln(3)$. □

We can now provide a bound for B that holds for t-semirandom order streams.

Lemma 6. *In t-semirandom order, the expected connection cost and expected facility cost of B are each less than $A_B + 2a|B| + f$.*

Proof. We substitute the bounds of Lemmas 4 and 5 into Lemma 3. The last part is to bound for the worst-case adversary $\mathcal{E}[\beta_0] = \sum_{k=0}^{t-1}\mathcal{E}[\beta'_j] \le \frac{1}{4}$ and observe that $\ln(18) < 3$. □

1.3.1 Application to Online Facility Location

Our main result for online facility location is now a simple corollary of Theorem 1. As shown in the Section 1.5, Theorem 1 yields results for both online facility location and k-median clustering by applying the theorem with different choices of r and h.

Corollary 1. *On a t-semirandom order stream,* `OFL` *is $(2+o(1))\left(\frac{\log_2 t}{\log_2\log_2 t}\right)$-competitive in expectation.*

Proof. Let C be an optimal facility set for S. Define $k = |C|$ and observe that $\mathsf{OPT}_k(S)$ is the connection cost associated with the optimal solution with facility set C. The optimal cost for the facility location problem is then $kf + \mathsf{OPT}_k(S)$. For any $\epsilon > 0$, set $r = (1+\epsilon)\frac{\log_2 t}{\log_2 \log_2 t}$ and $h = \lceil r \rceil$. Observe that $r^h \geq 4t$ for all t greater than some function of ϵ. Applying Theorem 1 to S with these values of r, h, and k shows that the total expected cost is at most $2(r+3)(kf + \mathsf{OPT}_k(S))$. This implies the result since $3 < \epsilon r$ by taking t sufficiently large. $\square$

Remark on Aspect Ratio: Suppose that we are working in a metric space of aspect ratio[4] Δ. Setting $r^h \geq \Delta$ instead of $r^h \geq 4t$, observe in the proof of Theorem 1 that $B = \varnothing$. This yields a bound of $(r+3)\mathsf{OPT}_k(S) + (h+1)kf$ on both the expected facility cost and expected connection cost. We may therefore replace t with $\min(t, \Delta)$ in our results. This justifies the lower bound in the following section being constructed in a metric space of aspect ratio t. Our upper and lower bounds match for all choices of t and Δ by substituting $\min(t, \Delta)$ for t.

1.4 Lower Bound

We present a lower bound on the expected competitive ratio of any randomized algorithm for the online facility location problem. The bound holds even when the algorithm can open a facility at any location in the metric space. The proof works by constructing a sequence of points that converge to the location of an optimal facility. At each step, there are enough possible locations of the optimal facility that no algorithm can guess (except with negligible probability) the correct location until it is too late.

Theorem 2. *On a t-semirandom order stream for $t \geq 4$, every randomized algorithm for online facility location has an expected competitive ratio of at least* $\frac{1}{3}\left\lceil \frac{\log_2 t}{\log_2 \log_2 t} \right\rceil$.

[4]The aspect ratio of a metric space is the ratio between the maximum and minimum non-zero distance between points.

We will construct a family of inputs and show that any deterministic algorithm has at least a certain competitive ratio when run on an input selected uniformly at random from this family. The result immediately extends to randomized algorithms by Yao's principle.

The Metric Space: Define $m := \lceil \frac{\log_2 t}{\log_2 \log_2 t} \rceil$, $h := m - 1$, and $D := f/h$. Let z be a positive integer. The points of the metric space are the nodes of a complete z-ary tree of depth h. This means that the root is at depth 0 and leaves are at depth h. The distance between a node at depth i and any of its z children is $Dm^{-i} - Dm^{-i-1}$. The distance between other nodes is obtained by summing the distances of the shortest path between them.

The Family of Inputs: The family of inputs is enumerated by the h^z possible strings of h numbers in $\{1, \ldots, z\}$. We now describe how to construct the input associated with $(b_1, \ldots, b_h) \in \{1, \ldots, z\}^h$. Define x_0 to be the root. Recursively define x_i to be the b_i^{th} child of x_{i-1}. For $i \in \{1, \ldots, h\}$, place m^i points at node x_i. We let the input size be n for any $n \geq t$. Since $\sum_{i=1}^{h} m^i < t \leq n$, there will be some remaining points. Place all remaining points at the root. The randomized input is to select a member from this family of z^h inputs uniformly at random.

The Optimal Cost: The optimal cost of the input associated with the string $(b_1, \ldots, b_h)$ is at most the cost of the solution that places facilities at the root and at x_r, connecting all non-root points to x_r. The distance between x_i and x_r is $\sum_{j=i}^{h-1}(Dm^{-j} - Dm^{-j-1}) = Dm^{-i} - Dm^{-h} < Dm^{-i}$. Therefore the optimal cost is less than $2f + \sum_{i=1}^{h} m^i(Dm^{-i}) = 3f$ for every input the the family.

The t-Bounded Adversary's Strategy: There are $\sum_{i=1}^{h} m^i < t$ points not located at the root. This implies that regardless of the order that the points are sent, a t-bounded adversary can deterministically ensure that points arrive in non-decreasing order of depth. The adversary does this by simply sending along any elements at the root while storing the at most t other elements until they are ready to be sent in non-decreasing order of depth. We consider this arrival order.

The Algorithm's Optimal Strategy: We define a cost scheme for the algorithm that is strictly less than the original cost (therefore any lower bound in this easier scheme is valid for the original problem). This greatly simplifies the analysis by allowing us to isolate an optimal strategy.

Suppose that if a facility is open at certain node, then the connection cost of a point at any ancestor[5] node is zero. With this modification, we can define an optimal strategy when a point arrives. If there is an open facility at any descendant node, then connect this point with zero cost. Otherwise, open a facility with cost f at any descendant leaf node and then connect with zero cost.

If there is no open facility at a descendant node, the second option is optimal since the nearest facility cannot be closer than the parent node. The parent node is at distance $Dm^{-(i-1)} - Dm^{-i} = fm^{-i}$. The algorithm, aware of the family of inputs, knows that a total of m^i points are coming at this node. This means that the total connection cost from this node would be at least f. Therefore it is optimal to pay f and open a new facility.

Given that we will open a new facility, placing it at a descendant leaf node is optimal because it minimizes the connection cost of future points in the stream. Without loss of generality, we have our deterministic algorithm always open at the descendant leaf node obtained by moving down the first child of each node in the path from the current node.

The Algorithm's Expected Cost: Using the algorithm's strategy defined above, one can see that the algorithm incurs zero connection cost. For the input associated with $(b_1, \ldots, b_h)$, the number of facilities besides the root is just the number of b_i not equal to 1. The probability that $b_i \neq 1$ is $1 - \frac{1}{z}$. The expected total cost is then $(1 - \frac{1}{z})h + 1$. Using sufficiently large z shows that it is not possible to achieve expected cost below $h + 1$.

[5]If node a is contained in the subtree of node b, we say that a is a descendant of b and that b is an ancestor of a.

The competitive ratio is therefore at least $(h+1)/3 = \frac{1}{3}\lceil \frac{\log_2 t}{\log_2 \log_2 t} \rceil$ as desired. The result extends to randomized algorithms by Yao's principle.

1.5 k-Median Clustering on Streams

In this section, we present a $O(1)$-approximation streaming algorithm for k-median clustering that stores $O(k \log t)$ points and has $O(k \log t)$ worst-case update time. The extension to other functions is sketched in Section 1.5.1. Our algorithm is based on the doubling algorithm of [9]. Among our innovations is a subroutine $\mathcal{B}(X, k)$ that permits us to improve the update time and approximation ratio of the original algorithm. $\mathcal{B}(X, k)$ is described in Section 1.5.2 and comes with the following guarantee:

Theorem 3. *For a weighted set X of n distinct points, suppose that the nearest neighbor in X has already been computed for each $x \in X$. Given an integer $k \geq 1$, the algorithm $\mathcal{B}(X, k)$ terminates in $O(n)$ time and outputs a pair (Z, λ) such that Z is a weighted set of at most $\lfloor \frac{n+k}{2} \rfloor$ points and $\lambda = \textsf{COST}(X, Z) < 2\textsf{OPT}_k(X)$.*

Throughout this section, as in Theorem 3, we overload notation by writing $\textsf{COST}(A, B)$ where B is a weighted set (with total weight equal to that of A). Recall that for an unweighted set B, the function $\textsf{COST}(A, B)$ denotes the minimum connection cost of connecting the demand set A to the facility set B where each facility can service an unlimited number of demands. When B is a weighted set, we let $\textsf{COST}(A, B)$ denote the minimum connection cost under the restraint that each facility must service exactly its weight in demands.

We now present Algorithm 1 to maintain a set Ψ which we show can determine a $O(1)$-approximation to the k-median clustering of the stream. Observe that the main loop of Lines 3-15 always begins with $|\Psi| \leq 29m$, which ensures by Theorem 3 that the while-loop of Lines 7-15 always begins with $|\Psi| \leq 15m$.

Lemma 7. *Except between Lines 4 and 5, $\textsf{COST}(P, \Psi) < 20L$.*

Algorithm 1 Input: integer $m \geq 1$, a stream of points P

```
1: $L \leftarrow 0$
2: $\Psi \leftarrow$ the first $29m$ points of $P$
3: loop
4:     $(\Psi, \lambda) \leftarrow \mathcal{B}(\Psi, m)$
5:     $L \leftarrow \max(10L, \lambda/3)$
6:     $\mathsf{COST} \leftarrow 0$
7:     while $|\Psi| < 29m$ and $\mathsf{COST} < 14L$ do
8:         $p \leftarrow$ next point of $P$
9:         $y \leftarrow \arg\min_{y' \in \Psi} d(p, y')$
10:        $u \leftarrow$ uniform random in $(0, 1)$
11:        if $uL < m\, d(p, y)$ then
12:            $\Psi \leftarrow \Psi \cup \{p\}$
13:        else
14:            $w(y) \leftarrow w(y) + 1$
15:            $\mathsf{COST} \leftarrow \mathsf{COST} + d(p, y)$
```

Proof. The variable COST is an upper bound on the increase of $\mathsf{COST}(P, \Psi)$ during the current instance of the while-loop. Since COST increases by at most $\frac{L}{m}$ in each iteration, the termination condition of Line 7 ensures that $\mathsf{COST} < 15L$. It remains to show that the cost when the while-loop began was at most $5L$. Let us recursively assume that when the previous while-loop began, the cost was at most $5L'$ where L' was the previous value of L'. During that instance, less than $15L'$ cost was incurred. On Line 4, exactly λ cost was incurred. We may bound $\lambda \leq 3L$ and $L' \leq L/10$ by Line 5. Therefore the total cost is $5L' + 15L' + \lambda \leq 5L$ as desired. $\square$

The next lemma addresses a subtle issue that only arises for $1 < t < n$. Observe that any segment of a random-order ($t = 1$) stream is in random-order, and that any segment of an adversarial-order ($t = n$) stream is in adversarial-order. However, a segment of a t-semirandom order stream for $1 < t < n$ is not necessarily in t-semirandom order (or even in $2t$-semirandom order) because the adversary may have as many as t points in storage when the segment begins. Instead, we analyze a segment as two separate t-semirandom streams, one coming from the adversary's storage and the other as those points that the adversary has not yet received.

Lemma 8. *Assume that $m \geq k(4 + \lceil \log_2 t \rceil)$. Whenever the while-loop of Lines 7-15* *terminates,* $\mathsf{OPT}_k(P) > L$ *with probability* $\frac{1}{2}$.

Proof. Observe that the while-loop runs `OFL` with facility cost $f = \frac{L}{m}$. Setting $r = 2$ and $h = \lceil \log_2(4t) \rceil$, apply Theorem 1 and plug in the value for f. This shows that on a t-semirandom stream, `OFL` incurs less than $5\mathsf{OPT}_k(P) + L$ in expected connection cost and opens less than $m(1 + 5\mathsf{OPT}_k(P)/L)$ facilities in expectation. Observe that this bound holds for the points of P even when P is interlaced with points from another stream.

For a segment P of the t-semirandom stream, let P_1 be the adversary's storage at the beginning of the segment and let P_2 be all other points. Applying the previous argument twice, we see that on this segment `OFL` incurs less than $5\mathsf{OPT}_k(P) + 2L$ in expected connection cost and opens less than $m(2 + 5\mathsf{OPT}_k(P)/L)$ facilities in expectation. The terms involving $\mathsf{OPT}_k(P)$ did not double since $\mathsf{OPT}_k(P_1) + \mathsf{OPT}_k(P_2) \le \mathsf{OPT}_k(P)$. With probability at least $\frac{1}{2}$, the cost and number of facilities are most twice these bounds by Markov's inequality.

Suppose that $\mathsf{OPT}_k(P) \le L$. Then with probability at least $\frac{1}{2}$, the while-loop incurs less than $14L$ cost and opens less than $14m$ facilities. Since the while-loop begins with $|\Psi| \le 15m$, the termination condition means that either $14L$ cost was incurred or $14m$ facilities were opened. We conclude with probability at least $\frac{1}{2}$ that $\mathsf{OPT}_k(P) > L$ when the while-loop terminates. □

For correctness of the algorithm, the result of the preceding lemma only needs to hold for the most recent termination of the while-loop. We therefore apply it only once, avoiding a factor of $O(\log n)$ in the space bound that would result from applying the lemma at each loop iteration.

After processing a point, Algorithm 1 waits on Line 8 for the next point. We now extend the previous lemma to hold on this line, and therefore after each point has been processed.

Lemma 9. *Assume that* $m \ge k(4 + \lceil \log_2 t \rceil)$. *On Line 8,* $L < 14\,\mathsf{OPT}_k(P)$ *with probability* $\frac{1}{2}$.

Proof. Let L' and Ψ' be the states of L and Ψ at the beginning of the current iteration of the main loop. We condition upon the event that $\mathsf{OPT}_k(P) > L'$, which occurs with probability at least $\frac{1}{2}$ by Lemma 8. From Line 5 we infer that either $L = 10L'$ or $3L = \lambda = \mathsf{COST}(\Psi', \Psi)$.

In the case that $L = 10L'$, then the result is immediate. Otherwise, $3L = \mathsf{COST}(\Psi', \Psi) < 2\mathsf{OPT}_m(\Psi')$ by the guarantee of $\mathcal{B}(X, k)$. We have $\mathsf{OPT}_m(\Psi') \leq \mathsf{OPT}_k(\Psi')$ since $m \geq k$ by assumption. Applying the triangle inequality to each point of Ψ', we get $\mathsf{OPT}_k(\Psi') \leq \mathsf{OPT}_k(P) + \mathsf{COST}(P, \Psi')$. We have that $\mathsf{COST}(P, \Psi') < 20L' < 20\mathsf{OPT}_k(P)$, where the first inequality is by Lemma 7 and the second inequality is the event we have conditioned upon. Therefore $L < 14\mathsf{OPT}_k(P)$. This result holds regardless of how many iterations of the while-loop have occurred since $\mathsf{OPT}_k(P)$ is non-decreasing as points are added to P. □

We now state our main theorem for clustering. Amplifying the probability of success is simple; run $\lceil \log_2 \frac{1}{\delta} \rceil$ independent instances of Algorithm 1 in parallel and return the Ψ from an instance with minimal L.

Theorem 4. *Algorithm 1 with parameter m can be implemented to run in $O(m)$ worst-case update time and store $O(m)$ points. Suppose that the input stream P of n points arrives in t-semirandom order. If $m \geq k(4 + \lceil \log_2 t \rceil)$, then with probability at least $\frac{1}{2}$ the set Ψ maintained by the algorithm satisfies $\mathsf{COST}(P, \Psi) \leq O(1) \cdot \mathsf{OPT}_k(P)$.*

Proof. Combining Lemmas 7 and 9 shows that $\mathsf{COST}(P, \Psi) < 280\, \mathsf{OPT}_k(P)$. It is immediate from the pseudocode that the storage is less than $29m$ points. As for the update time, each iteration of the while-loop requires $O(m)$ time. By Theorem 3, if the nearest neighbor function for Ψ has been computed then Line 4 terminates in $O(m)$ time. We must show how to ensure with $O(m)$ worst-case update time that the nearest neighbor function for Ψ has been computed before each time that Line 4 is executed.

Given the nearest neighbor function π for a set of n points, observe that we can insert a point into the set and update π in $O(n)$ time. Beginning with the nearest neighbor function π of the first two points of Ψ, we simply update π with the next three points of Ψ each time one point is received from the stream. The while-loop runs at least $14m$ times before each time that Line 4 executes. Therefore the nearest neighbor function for all of Ψ is guaranteed to have been computed by the time the while-loop terminates. $\square$

By tweaking parameters and refining the analysis, one can improve the guarantee to $\mathsf{COST}(P, \Psi) < 3\mathsf{OPT}_k(P)$ (in particular, the space blows up as the constant approaches 2). It is well-known that if $\mathsf{COST}(P, \Psi) \leq \alpha\mathsf{OPT}_k(P)$ then any γ-approximation of Ψ is a $(\alpha(\gamma + 1) + \alpha)$-approximation of P [9]. Therefore Theorem 4 implies that Ψ carries enough information to determine a $O(1)$-approximation of P. Our constant $\alpha \leq 3$ does not guarantee a particularly low approximation ratio. However, Algorithm 1 can be used as a building block for a more accurate solution. Using the technique introduced in Part II, we can use Algorithm 1 to maintain an ϵ-coreset which carries enough information to determine a $(1 + \epsilon)$-approximation[6] for any $\epsilon > 0$. This technique essentially converts the constant in the approximation factor into a constant in the size of the coreset. The only space required in addition to Algorithm 1 is the space needed to store the ϵ-coreset. As an example, for k-median in $\mathbb{R}^d$, coresets of size $O(\epsilon^{-2}kd)$ are known [13], implying that our result can be used to determine a $(1 + \epsilon)$-approximation using $O(\epsilon^{-2}kd + k \log t)$ space.

As a corollary to Theorem 4, we obtain an $O(nk)$-time approximation algorithm for the RAM model. In light of the $\Omega(nk)$ time lower bound of [29], the runtime is optimal.

Corollary 2. *Given a set P of n points, there exists an algorithm that outputs a $O(1)$-approximation to the k-median clustering of P with probability $1 - \delta$ in time $O(nk \log \frac{1}{\delta})$.*

Proof. Shuffle P in $O(n)$ time. Set $m = 4k$ and run Algorithm 1 in $O(nk)$ time followed by the offline algorithm of [29] in $O(k^2)$ time to obtain by Theorem 4 a set C of k points such that $\mathsf{COST}(P, C) \leq O(1) \cdot \mathsf{OPT}_k(P)$ with probability at least $\frac{1}{2}$. Repeat this $\lceil \log_2 \frac{1}{\delta} \rceil$ times and output the solution of minimal cost. $\square$

[6]An efficiently computed solution will have a larger approximation factor. Both the k-median and k-means problem are `MAX-SNP` Hard for all $k \geq 2$. See the related work section of [4] for a survey of hardness results.

1.5.1 Extension to Other Functions

We have assumed that we are in a metric space $(\mathcal{X}, d)$ but we can weaken this assumption. Suppose that throughout our results we replace the metric with an arbitrary symmetric positive-definite function $D : \mathcal{X} \times \mathcal{X} \to [0, \infty)$. If D satisfies the triangle inequality, then D is a metric and our result for k-median applies directly. However, suppose that D just satisfies a weak triangle inequality for some $\beta \geq 1$:

$$D(a,c) \leq \beta(D(a,b) + D(b,c)) \text{ for all } a,b,c \in \mathcal{X}$$

All of our proofs go through with larger constants. The bound in Theorem 4 generalizes to $(h+1+\beta)fk + \beta(r+3\beta)\mathsf{OPT}_k(S)$ and the guarantee of the $\mathcal{B}(X,k)$ routine of the next subsection generalizes to $\mathsf{COST}(X,Z) < 2\beta\mathsf{OPT}_k(X)$. For any function D satisfying a constant β value, our results for both online facility location and clustering carry through with larger constants. An example application is that our results generalize to ℓ_p norms. Another important case is k-means that corresponds to $D(x,y) = d(x,y)^2$.

Recall that the maximum likelihood estimator for the mean μ of Gaussian data Q is the $\hat{\mu}$ that minimizes $\sum_{q\in Q} d(q,\hat{\mu})^2$. To handle outliers more robustly, the statistics community introduced M-estimators which generalize maximum likelihood estimation by minimizing $\sum_{q\in Q} \rho(d(q,\hat{\mu}))$ for some positive-definite function $\rho : [0,\infty) \to [0,\infty)$. An M-estimator along with a positive integer k defines a clustering problem to find a set C of k points that minimizes $\sum_{q\in Q} \min_{c\in C} \rho(d(q,c))$. Observe that we recover k-means for $\rho(x) = x^2$ and k-median for $\rho(x) = x$. The convergence and robustness properties of M-estimators have been well-studied, but we also observe that the function $D = \rho \circ d$ usually satisfies a weak triangle inequality for a very low β value. Evidently we can let β be any value such that $\rho(c) \leq \beta(\rho(a) + \rho(b))$ for all $a,b,c \geq 0$ such that $c \leq a + b$. In Table 1.4 we have calculated tight β values for the most commonly used M-estimators .

These results imply that $O(k \log t)$ space suffices to approximate M-estimators on a data stream. We remind the reader that we can use the technique that will be introduced

Estimator	ρ function	β
Linear	$\rho(x) = x$	1
Gaussian	$\rho(x) = x^2$	2
Huber	$\rho(x) = \begin{cases} x^2 & \text{if } x < 1 \\ 2x - 1 & \text{if x} \geq 1 \end{cases}$	2
Cauchy	$\rho(x) = \log(1 + x^2)$	2
Tukey	$\rho(x) = \begin{cases} 1 - (1 - x^2)^3 & \text{if } x < 1 \\ 1 & \text{if x} \geq 1 \end{cases}$	2

Table 1.4: List of the most common M-estimators. The last column is redundant since the β value can be directly calculated from the ρ function. All of these estimators are actually a parameterized family by scaling $\rho(x)$ to $c_1\rho(c_2x)$ for $c_1, c_2 > 0$, but we choose only a single representative since the β value is unchanged.

in Part II to maintain an ϵ-coreset which permits a $(1 + \epsilon)$-approximation to the optimal M-estimator.

1.5.2 The $\mathcal{B}(X, k)$ Routine

We present a deterministic algorithm $\mathcal{B}$ that accepts a weighted set A of n distinct points along with a positive integer k and returns a weighted set Z of $\lfloor \frac{n+k}{2} \rfloor$ distinct points such that $\mathsf{COST}(A, Z) < 2\mathsf{OPT}_k(A)$. If the nearest neighbor graph on A has been computed, then $\mathcal{B}$ terminates in $O(n)$ time.

In what follows, there must be a way to order the points of A. This is necessary for a technical detail that comes up in Lemma 10; we need a consistent way to break ties. In practice, this can simply be the order that the algorithm loops through the points of A.

Definition 4. *The function $\pi : A \to A$ is defined such that $\pi(a) = \arg\min_{x \in A \setminus \{a\}} d(a, x)$. If there is more than one point x that minimizes $d(x, a)$, break ties by selecting the x that is greatest according to the order of A.*

We use π to define a graph as follows:

Definition 5. *Let A be a weighted set. The directed graph $G(A, \pi)$ has the points of A*

as vertices. For each vertex a, there is exactly one directed edge leaving a and pointing to $\pi(a)$.

$G(A,\pi)$ possesses a special structure of not containing any cycles of length greater than 2.

Lemma 10. *The graph $G(A,\pi)$ contains cycles only of length* 2*.*

Proof. Let $\{a_1,\dots,a_s\}$ be a cycle such that for each $1 \le i \le s$ we have $\pi(a_i) = a_{i+1}$ (additions should be interpreted modulo s). We will show that $s = 2$.

By definition of π, it must be that $d(a_i,a_{i+1}) \le d(a_{i-1},a_i)$ since $\pi(a_i) = a_{i+1}$. Then we have $d(a_1,a_2) \le \ldots \le d(a_s,a_1) \le d(a_1,a_2)$ and the chain of inequalities implies equality. Let a_t be the element of the cycle that is greatest according to the ordering of A. Since a_t and a_{t+2} are equidistant from a_{t+1}, the criterion for breaking ties in Definition 4 ensures that $\pi(a_{t+1}) = a_t$. Since $\pi(a_{t+1}) = a_{t+2}$, it must be that $t+2 = t$ and so $s = 2$. □

In light of Lemma 10, let us consider the structure of the directed graph (A,π). Removing the edges in length-2 cycles, we are left with a forest (a collection of trees directed to the root). Considering the full graph along with these 2-cycles, we see that each component is a pair of trees whose roots are coupled. This forest of "bi-trees" can be 2-colored, and the following lemma shows that we can do this efficiently.

Lemma 11. *Given the function π, the graph $G(A,\pi)$ can be 2-colored in $O(m)$ time.*

Proof. We say that $\pi(a)$ is the parent of a, and that a is the child of $\pi(a)$. Each vertex has exactly one parent, and the edges point to the parent. For each point $a \in A$, we store a pointer to its parent as well as a list of pointers to its children. Given the function π, this can be accomplished in $O(n)$ time.

As reasoned above, the graph can be partitioned into bi-tree components. We use the following iterative procedure until all vertices have been colored: (1) Select any uncolored vertex; (2) Walk along the edges until reaching the two roots; (3) Color each root a different color; (4) Recursively color each child vertex the opposite color than its parent.

For Step 2, we will know we have located the roots when we return to the vertex we just left. This process (of moving from a to $\pi(a)$ until reaching the root) terminates in time proportional to the depth of the tree. Therefore a total of $O(n)$ time is spent during Step 2 over all iterations of this procedure.

For Step 4, finding a child takes $O(1)$ time since we have stored a list of children with each vertex. Therefore a total of $O(n)$ time is spent during Step 4 over all iterations of this procedure. $\square$

We will need the following technical lemma to bound $\mathsf{COST}(A, Z)$. Recall that $\mathsf{OPT}_k(A)$ is defined using centers from anywhere in the metric space $\mathcal{X}$. The lemma says that if we restrict to centers from A itself, then the optimal cost increases by less than a factor of two.

Lemma 12. *Let $\overline{\mathsf{OPT}}_k(A)$ denote the minimum of $\mathsf{COST}(A, B)$ where $B \subset A$ ranges over all sets of k points. Then $\overline{\mathsf{OPT}}_k(A) < 2\mathsf{OPT}_k(A)$.*

Proof. Let $C \subset \mathcal{X}$ be a set of k points such that $\mathsf{COST}(A, C) = \mathsf{OPT}_k(A)$. For each $c \in C$, let c' be the closest point of A to c. Any element $a \in A$ that was connected to c can be instead connected to c' with a cost of $d(a, c') \leq d(a, c) + d(c, c') \leq 2d(a, c)$ since $d(c, c') \leq d(c, a)$. Moreover, the cost of this cluster increased by strictly less than factor of two since if $c \notin A$ the cost decreased for $a = c'$ and if $c \in A$ the cost stayed the same. Define $C' = \{c'\}_{c \in C}$. Then $C' \subset A$ is a set of k points such that $\overline{\mathsf{OPT}}_k(A) \leq \mathsf{COST}(A, C') < 2\mathsf{COST}(A, C) = 2\mathsf{OPT}_k(A)$. $\square$

The $\mathcal{B}(X, k)$ algorithm is presented in the next theorem. The basic idea is to 2-color $G(A, \pi)$ and then eliminate one of the colors by relocating those points to their image under π. Since π maps each point to a point of the opposite color, this transformation increases the weights of one color while completely eliminating the other.

Theorem 5. *Let A be a weighted set of n distinct points. Assume that π has been computed for A. In $O(n)$ time, we can compute a weighted set Z of at most $\lfloor \frac{n+k}{2} \rfloor$ distinct points such that $\mathsf{COST}(A, Z) < 2\mathsf{OPT}_k(A)$.*

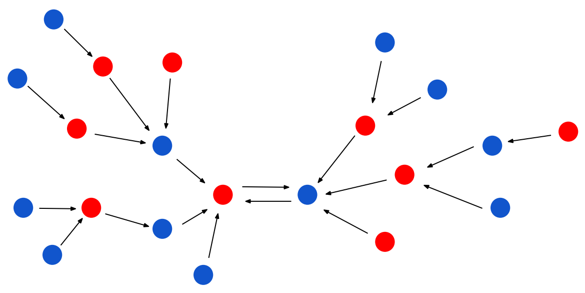


Figure 1.2: A 2-colored bi-tree of the nearest neighbor graph $G(A, \pi)$. Algorithm $\mathcal{B}(A, k)$ removes all blue points by moving the weight of each blue point to the red point to which it points.

Proof. Let the function $w : A \to \mathbb{N}$ map each point of A to its weight. By Lemma 11, we 2-color A in $O(n)$ time. Let A_1 and A_2 be the partition of A into the two colors after removing the k points with the top values of $w(a)d(a, \pi(a))$. Let $|A_t|$ be the larger component (by number of points) and note that $|A_t| \geq \lceil \frac{n-k}{2} \rceil$.

Build Z from A as follows: for each $a \in A_t$, increment $w(\pi(a))$ by $w(a)$ and delete a. This procedure terminates in $O(n)$ time. By definition of a 2-coloring, $\pi(a) \notin A_t$ for every element $a \in A_t$, and so Z contains $n - |A_t| \leq \lfloor \frac{n+k}{2} \rfloor$ weighted points.

Observe that $\mathsf{COST}(A, Z) = \sum_{a \in A_t} w(a)d(a, \pi(a))$. The optimal k-median solution of A using points from A involves moving $n - k$ points of A by at least the distance to the nearest neighbor, so $\sum_{a \in A_1 \cup A_2} w(a)d(a, \pi(a)) \leq \overline{\mathsf{OPT}}_k(A)$ and so $\mathsf{COST}(A, Z) \leq \overline{\mathsf{OPT}}_k(A)$. This completes the proof since Lemma 12 guarantees $\overline{\mathsf{OPT}}_k(A) < 2\mathsf{OPT}_k(A)$. □

To complete the guarantee of Theorem 3, observe that we can return the exact value of $\mathsf{COST}(A, Z)$ in $O(n)$ time.

1.6 Discussion of Models for Semirandom Order

IITK Open Problem #15 addresses computational models of semirandom-order for streams and asks how the models of t-bounded adversarial order and ϵ-generated random order relate to each other [26]. One can verify that $\epsilon = 0$ like $t = 0$ is equivalent to random-order and that $\epsilon = 1$ like $t = n$ is equivalent to adversarial-order. However, we demonstrate in the following two lemmas that no other relations hold between these models.

Lemma 13. *To simulate the t-bounded adversarial order model for any $t > 1$, the*

ϵ-generated order model requires $\epsilon \geq 1 - 2^{-n/2}$.

Proof. Let the stream consist of elements $\{1, \dots, n\}$. Let $\chi(i)$ denote the identity of the i^{th} element in the initial random-order stream. For any $t \geq 2$, a t-bounded adversary can ensure that $\chi(2i-1) < \chi(2i)$ for all $1 \leq i \leq \frac{n}{2}$. Let A_i be the event that $\chi(2i-1) < \chi(2i)$. In random-order, observe that $P(A_i) = \frac{1}{2}$ and that the $\{A_i\}_{i=1}^{n/2}$ are mutually independent. Therefore the uniform distribution μ assigns probability mass $1 - (\frac{1}{2})^{n/2}$ to the orders satisfying $\chi(2i-1) > \chi(2i)$ for some $1 \leq i \leq \frac{n}{2}$. Any distribution ν that assigns probability mass 1 to $\cap_{i=1}^{n/2} A_i$ must satisfy $||\mu - \nu||_1 \geq 2(1 - 2^{-n/2})$ □

Lemma 14. *To simulate the ϵ-generated order model for any $\epsilon \geq 2^{-n/10}$, the t-bounded adversarial order model requires $t > n/2$.*

Proof. Let X be a subset of r elements. In random-order, let A be the event at at least one element of X arrives among the first βn elements. $P_\mu(A) \geq 1 - (1-\beta)^r$, so we can create a distribution ν such that $P_\nu(A) = 1$ and $||\mu - \nu||_1 \leq 2(1-\beta)^r$.

In the t-bounded adversarial order model, elements are *sent* in random-order but intercepted by an adversary who manipulates the order that elements *arrive* for the algorithm. Observe that an element among the last $\frac{n-t}{2}$ to be sent cannot be among the first $\frac{n-t}{2}$ to arrive. If $r \leq \frac{n-t}{2}$ then with positive probability all elements of X are among the last $\frac{n-t}{2}$ to be sent. Therefore if $\beta n \leq \frac{n-t}{2}$ we have $P(A) < 1$. Setting $r = \beta n$, this necessitates that $t > (1 - 2\beta)n$ whenever $\epsilon \geq (1-\beta)^{\beta n}$. The result follows by setting $\beta = \frac{1}{4}$. □

Our matching bounds for online facility location show a non-trivial degradation of performance in the t-bounded adversarial-order model that smooothly interpolates between random-order and adversarial-order. This supports the claim that t-bounded adversarial-order is a viable model of semi-randomness. In contrast, it is trivial to show matching bounds of $\Theta(1 + \frac{\epsilon \log n}{\log \log n})$ in the ϵ-generated model. More generally, any bound in expectation which is $\Theta(f)$ in random-order and $\Theta(g)$ in adversarial-order implies a $\Theta(f + \epsilon g)$ bound in the ϵ-generated model. As a result, the ϵ-generated model is not interesting for a wide class of problems. This class is rather large since any $\Theta(f \log \frac{1}{\delta})$ bound with probability $1 - \delta$ implies a $\Theta(f)$ bound in expectation.

We conclude with an open question:

Open Question: Given an adversarial-order bound of $O(f(n))$, all the results in this paper present a bound of $O(f(t))$ for t-bounded adversarial order, showing a smooth degradation as t increases. However, some problems exhibit a sharp phase transition. For example, the size of the largest component in an Erdős-Rényi graph $ER(n, p)$ jumps from $O(\log n)$ to $\Omega(n)$ around $p \sim 1/n$. In the t-bounded adversarial order model, is it always the case that bounds degrade smoothly as t increases? Alternatively, do problems exist that exhibit a sharp jump in some quantity of interest (i.e. time, space, or approximation factor) when t increases by only a constant factor around some value?

Part II

Nearly Optimal Coresets on Insertion-Only Streams

In the algorithmic field of computer science, we usually have an optimization problem at hand and a state-of-the-art or a straight-forward exhaustive search algorithm that solves it. The challenge is then to suggest a new algorithm with a better running time, storage or other feature. A different and less traditional approach is to use data reduction, which is a compression of the input data in some sense, and to run the (possibly inefficient) existing algorithm on the compressed data. In this case, the problem of solving the problem at hand reduced to the computing a problem-dependent compression such that:

1. an existing algorithm that solves the optimization problem on the original (complete) data, will yield a good approximate solution *for the original data* when applied on the compressed data.

2. The time and space needed for constructing the compression and running the optimization algorithm on the coreset will be better than simply solving the problem on the complete data.

There are many approaches for obtaining such a provable data reduction for different problems and from different fields, such as using uniform sampling, random projections (i.e., the Johnson-Lindenstrauss Lemma), compressed sensing, sketches or PCA.

In this part of the thesis (Part III), we focus on a specific type of a reduced data set, called coreset (or core-set) that was originated in computational geometry, but now applied in other fields such as computer vision and machine learning. This part is organized into basic sections: results for maintaining coresets over data streams (Section 2.2) and results for offline coresets (Sections 2.3-2.5). We briefly introduce both of these topics the remainder of this section. Many of our results, along with comparison to prior works, are summarized in Table 2.5 in Section 2.1.

In the Appendix A (Section 7) we summarize the merge-and-reduce technique that is used in previous approaches [12, 27, 26, 1, 19]. In the Appendix B (Section 8) we provide an alternative framework that generalizes our main result (Theorem 2.2.1), applying to a wide-array of constructions although giving a weaker bound.

2.0.1 Streaming Results

In the streaming model of computation, the input arrives sequentially. This differs from the standard model where the algorithm is given free access to the entire input. Given a memory that is linear in the size of the input, these models are evidently equivalent; therefore the goal of a streaming algorithm is to perform the computation using a sublinear amount of memory.

Our stream consists of n elements $p_1, \ldots, p_n$. In the streaming model (or more specifically the insertion-only streaming model, since points that arrive will never be deleted), we attempt to compute our solution using $o(n)$ memory. Sometimes the algorithm will be allowed to pass over the stream multiple times, resulting in another parameter called the number of passes. All of our algorithms use $\text{polylog}(n)$ memory and require only a single pass.

Prior to the current work, the merge-and-reduce technique due to Har-Peled and Mazumdar [27] and Bentley and Sax [7] was used to maintain a coreset on an insertion-only stream. For a summary of this technique, see Section 2.6 in the Appendix. In this part, we introduce an alternative technique that reduces the multiplicative overhead from $\log^{2a+1} n$ to $\log n$ (here, a is the offline construction's dependence on $1/\epsilon$). While our method is not as general as merge-and-reduce (it requires that the function in question satisfies more than just the "merge" and "reduce" properties, defined in Section 2.6), it is general enough to apply to all M-estimators. For the special case of our coreset offline construction for M-estimators (introduced in Section 2.5), we use a more tailored method that causes this to be $\log n$ additive overhead. Therefore our streaming space complexity matches our offline space complexity, both of which improve upon the state-of-the-art.

The offline coreset construction of [19] has the following structure: first, a bicriterion approximation is computed. Second, points are sampled according a distribution that depends only on the distances between points of the input and their assigned bicriterion centers. This suggests a two-pass streaming algorithm (which we later combine into a single pass): in the first pass, construct a bicriterion using an algorithm such as [9]. In the second pass, sample according to the bicriterion found in the first pass. This provides a two-pass algorithm for a coreset using $O(\epsilon^{-2}k \log k \log n)$-space. Our contribution is showing how these two passes can be combined into a single-pass. Using the algorithm of [9] to output $O(k \log n)$ centers at any time, we show that this is sufficient to carry out the sampling (originally in the second pass) in parallel without re-reading the stream. Our main lemma (Lemma 2.2.7) shows that the bicriterion, rather than just providing "central" points to concentrate the sampling, actually can be thought of as a proof of the importance of points for the coreset (technically, a bound on the "sensitivity" that we define at the beginning of Section 2.2). Moreover, the importance of points is non-increasing as the stream progresses, so we can maintain a sample in the streaming setting without using any additional space.

2.0.2 Offline Results

The name coreset was suggested by Agarwal, Har-Peled, and Varadarajan in [2] as a small subset S of points for a given input set P, such that any shape from a given family that covers S will also cover P, after expanding the shape by a factor of $(1+\varepsilon)$. In particular, the smallest shape that covers S will be a good approximation for the smallest shape that covers P. For approximating different cost functions, e.g. the sum of distances to a given shape, we expect that the total weight of the sample will be similar to the number n of input points. Hence, in their seminal work [27], Har-Peled and Mazumdar used multiplicative weights for each point in S, such that the weighted sum of distances from S to a given shape from the family will approximate its sum of distances from the original data. In [27] each shape in the family was actually a set of k points, and the application was the classic k-means problem.

In this part, we are given an input set P (called *points*), a family (set) Q of items, called *queries* and a function $f : P \to [0, \infty)$ that is called a *cost* function. A coreset is then a *subset* S of P, that is associated with a non-negative weight function $u : S \to [0, \infty)$ such that, for every given query $q \in Q$, the sum of original costs $\sum_{p \in P} f(p, q)$ is approximated by the weighted sum $\sum_{p\in S} u(p)f(p,q)$ of costs in S up to a multiplicative factor, i.e.,

$$(1-\varepsilon)\sum_{p\in P} f(p,q) \leq \sum_{p\in S} u(p)f(p,q) \leq (1+\varepsilon)\sum_{p\in P} f(p,q)$$

While our framework is general we demonstrate it on the k-means problem and its variant. There are at least three reasons for this: (i) This is a fundamental problem in both computer science and machine learning, (ii) This is probably the most common clustering technique that used in practice, (iii) Many other clustering and non-clustering problems can be reduced to k-means; e.g. Mixture of Gaussians, Bregman Clustering, or DP-means [18, 33, 5]. In this context we suggest offline coreset constructions for k-clustering queries, that can be constructed in a streaming fashion using our streaming approach, and are:

- of size linear in k (for $d > \log k$) and arbitrary metric space of dimension d. Current coresets that are subset of the input have size at least cubic in k [30]. This is by reducing the total sensitivity to $O(1)$ without introducing negative weights that might be conditioned on the queries as in [19].

- of size independent of d for the Euclidean case of k-means (squared distances). This is particular useful for sparse input set of points where $d \geq n$, such as in adjacency matrices of graphs, document-term, or image-object matrices. Recent coreset for sparse k-means of [6] is of size exponential in $1/\varepsilon$ and thus turn to $O(n)$ when used when the merge-and-reduce tree (where ε is replaced by $O(\varepsilon/\log(n))$. The result of [20] for k-means is exponential in k/ε and fails with constant probability, so also cannot be used with streaming. Another result of [20] suggests a coreset type set for k-means of size $O(k/\varepsilon)$ but which is based on projections that loss the sparsity of the data. Similar sparsity loss occurs with other projection-type compression methods e.g. in [15]. Nevertheless, we use the technique in [20] to bound the dimension of the k-means problem by $O(k/\varepsilon)$.

- of size independent of d for the Euclidean case, and non-squared distances, using weak coresets. These coresets can be used to approximates the optimal solution, but not every set of k centers. Unlike the weak coresets in [19, 17], we can use any existing heuristic on these coresets, as explained in Section 2.5.1.

- Robust to outliers. This is since the general pseudo-metric definition we used (inspired by [21], support m-estimators which is a tool for handling outliers [38]. Unlike in [21] our coresets are linear (and not exponential) in k, and also independent of n (and not logarithmic in n).

2.1 Related Work

The following table summarizes previous work along with our current results. By far, the most widely-studied problems in this class have been the k-median and k-means functions. In general, the extension to arbitrary M-estimators is non-trivial; the first such result was [21]. Our approach naturally lends itself to this extension. M-estimators are highly important for noisy data or data with outliers. As one example, Huber's estimator is widely used in the statistics community [25, 29]. It was written that "this estimator is so satisfactory that it has been recommended for almost all situations" [40]. Our results work not only for Huber's estimator but for all M-estimators, such as the Cauchy and Tukey biweight functions which are also well-used functions.

Note that in the below table, $\tilde{O}$ notation is used to write in terms of d, ϵ, k, and $\log n$ (therefore hiding factors of $\log \log n$ but not $\log n$).

Framework. A generic framework for coreset construction was suggested in [19]. The main technique is a reduction from coreset to ε-approximations, that can be computed using non-uniform sampling. The distribution of the sampling is based on the importance of each point (in some well defined sense), and the size of the coreset depends on the sum of these importance levels. This term of importance appeared in the literature as leverage

Problem	Offline Size	Streaming Size	Paper
Euclidean k-means	$O(k\epsilon^{-d}\log n)$	$O(k\epsilon^{-d}\log^{d+2} n)$	[27]
Euclidean k-means	$O(k^3\epsilon^{-(d+1)})$	$O(k^3\epsilon^{-(d+1)}\log^{d+2} n)$	[26]
Euclidean k-means	$O(dk^2\epsilon^{-2}\log n)$	$O(dk^2\epsilon^{-2}\log^8 n)$	[12]
Euclidean k-means	$O(dk\log k\epsilon^{-4})$	$O(dk\log k\epsilon^{-4}\log^5 n)$	[19]
Euclidean k-means	$\tilde{O}((d/\epsilon)^{O(d)}k\log n)$	$\tilde{O}((d/\epsilon)^{O(d)}k\log^{O(d)} n)$	[1]
Euclidean k-means	$O(\epsilon^{-2}k\log k\min(k/\epsilon, d))$	$O(\epsilon^{-2}k\log k\min(\frac{k}{\epsilon}, d) + k\log n)$	**
Metric k-means	$O(\epsilon^{-2}k^2\log^2 n)$	$O(\epsilon^{-2}k^2\log^8 n)$	[13]
Metric k-means	$O(\epsilon^{-4}k\log k\log n)$	$O(\epsilon^{-4}k\log k\log^6 n)$	[19]
Metric k-means	$O(\epsilon^{-2}k\log k\log n)$	$O(\epsilon^{-2}k\log k\log n)$	**
Euclidean k-median	$\tilde{O}(dk^2\epsilon^{-2}\log n)$	$O(dk^2\epsilon^{-2}\log^8 n)$	[12]
Euclidean k-median	$O(k\epsilon^{-d}\log n)$	$O(k\epsilon^{-d}\log^{d+2} n)$	[27]
Euclidean k-median	$O(k^2\epsilon^{-d})$	$O(k^2\epsilon^{-(d)}\log^{d+1} n)$	[26]
Euclidean k-median	$O(d\epsilon^{-2}k\log k)$	$O(d\epsilon^{-2}k\log k\log^3 n)$	[19]
Euclidean k-median	$O(d\epsilon^{-2}k\log k)$	$O(d\epsilon^{-2}k\log k + k\log n)$	**
Metric k-median	$O(k^2\epsilon^{-2}\log^2 n)$	$O(k^2\epsilon^{-2}\log^8 n)$	[12]
Metric k-median	$O(\epsilon^{-2}k\log k\log n)$	$O(\epsilon^{-2}k\log k\log^4 n)$	[19]
Metric k-median	$O(\epsilon^{-2}k\log k\log n)$	$O(\epsilon^{-2}k\log k\log n)$	**
Euclidean M-estimator	$O(\epsilon^{-2}k^{O(k)}d^2\log^2 n)$	$O(\epsilon^{-2}k^{O(k)}d^2\log^5 n)$	[21]
Euclidean M-estimator	$O(d\epsilon^{-2}k\log k)$	$O(d\epsilon^{-2}k\log k + k\log n)$	**
Metric M-estimator	$O(\epsilon^{-2}k^{O(k)}\log^4 n)$	$O(\epsilon^{-2}k^{O(k)}\log^7 n)$	[21]
Metric M-estimator	$O(\epsilon^{-2}k\log k\log n)$	$O(\epsilon^{-2}k\log k\log n)$	**

Table 2.5: Summary of Related Work on Coresets

score (in the context of low-rank approximation, see [36] and references therein), or, for the case of k-clustering, sensitivity [30]. The proof of many previous coreset constructions were significantly simplified by using this framework, and maybe more importantly, the size of these coresets was sometimes significantly reduced; see [19] for references. Many of these coresets size can be further improve by our improved framework, as explained below.

The size of the coreset in [19] depends quadratically on the sum of sensitivities, called *total sensitivity* [30]. In this part, we reduce the size of the coreset that are constructed by this framework to be only near-linear ($t\log t$) in the total sensitivity t. In addition, we generalize and significantly simplify the notation and results from this framework.

k-means. In the k-means problem we wish to compute a set k of centers (points) in some metric space, such that the sum of squared distances to the input points is minimized, where each input point is assigned to its nearest center. The corresponding coreset is a positively weighted subset of points that approximates this cost to every given set of k centers. First deterministic coresets of size exponential in d were first suggested by Har-Peled and Mazumdar in [27]. The first coreset construction of size polynomial in d was suggested by Ke-Chen in [12] using several sets of uniform sampling.

The state-of-the-art is the result of Schulman and Langberg [30] who suggested a coreset of size $O(d^2k^3/\varepsilon^2)$ for k-means in the Euclidean case based on non-uniform sampling. The distribution is similar to the distribution in [17] over the input points, however in [17] the goal was to have weaker version coresets that can be used to solve the optimal solution, but are of size independent of d.

Some kind of coreset for k-means of size near-linear in k was suggested in [19]. However, unlike the definition of this part, the multiplicative weights of some of the points in this coreset were (i) negative, and (ii) depends on the query, i.e., instead of a weight $w(p) > 0$ for an input point p, as in this part, the weight is $w(p, C) \in \mathbb{R}$ where C is the set of queries. While exhaustive search was suggested to compute a PTAS for the coreset, it is not clear how to compute existing algorithms or heuristics (what is actually done in practice) on such a coreset. On the contrary, generalizing an existing approximation algorithm for the k-means problem to handle positively weights is easy, and public implementations are not hard to find (e.g. in Matlab and Python).

k-mean for handling outliers. Coresets for k-means and its variants that handle outliers via m-estimators were suggested in [21], and also inspired this part of the thesis. The size of these coresets is exponential in k and also depend on logn. For comparison, we suggest similar coreset of size near-linear in k, and independent of n. PTAS for handling exactly m-outliers was suggested in [11] but with no coreset or streaming version.

Streaming. The metric results of [12, 19] and Euclidean results of [12, 27, 26, 19] that rely on merge-and-reduce have already been mentioned. A summary of these results appears in the tables below. For the specific case of Euclidean space, a more diverse set of stronger results is known. In particular, coreset constructions are known that do not begin with a bicriterion solution, and whose streaming variant does not rely on merge-and-reduce [1]. With the additional assumption in Euclidean space that the points lie on a discrete grid $\{1, \ldots, \Delta\}^d$, alternative techniques are known for k-means and other problems, even when the stream allows the deletion of points [22].

2.2 Streaming Algorithm

We present a streaming algorithm for constructing a coreset for metric k-means clustering that requires the storage of $O(\epsilon^{-2} k \log n)$ points. The previous state-of-the-art [19] required the storage of $O(\epsilon^{-4} k \log k \log^6 n)$ points. In this section we assume the correctness of our offline algorithm, which is proven in Section 2.5.

More generally, our technique works for M-estimators, a general class of clustering objectives that includes the well-known k-median and k-means functions as special cases. Other special cases include the Cauchy functions, the Tukey functions, and the L_p norms. Our method combines a streaming bicriterion algorithm [9] and a batch coreset construction [19] to create a streaming coreset algorithm. The space requirements are combined addivitely, therefore ensuring no overhead.

The streaming algorithm of [9] provides a bicriterion solution using $O(k \log n)$ space. Our new offline construction of Section 2.5 requires $O(\epsilon^{-2} k \log k \log n)$ space. Therefore our main theorem yields a streaming algorithm that combines these spaces additively, therefore requiring $O(\epsilon^{-2} k \log k \log n)$ space while maintaining a coreset for k-means clustering. The previous state-of-the-art framework that works for the metric variant (other methods are known to improve upon this for the special case of Euclidean space) was the merge-and-reduce technique [7] that yields a streaming algorithm requiring $O(\epsilon^{-4} k \log k \log^6 n)$ space, incurring an overhead of $\Theta(\log^5 n)$ over the offline coreset size. In comparison, our framework incurs no overhead. The additional improvement in our space is due the improved offline construction given in Sections 5-7.

We now state our main theorem. The result is stated in full generality: the k-median clustering in a ρ-metric space (see Definition 2.5.1). Note that metric k-means clustering corresponds to setting $\rho = 2$. Also, the probability of success $1 - \delta$ typically has one of two meanings: that the construction succeeds at the end of the stream (a weaker result), or that the construction succeeds at every intermediate point of the stream (a stronger result). Our theorem gives the stronger result, maintaining a valid coreset at every point of the stream.

Theorem 2.2.1 (Main Theorem). *There exists an insertion-only streaming algorithm that maintains a (k, ϵ)-coreset for k-median clustering in a ρ-metric space, requires the storage of $O(\rho^2\epsilon^{-2}k\log(\rho k)\log(n)\log(1/\delta))$ points, has $poly(k, \log n, \rho, \epsilon, \log(1/\delta))$ worst-case update time, and succeeds at every point of the stream with probability $1 - \delta$.*

Our method can be applied to the coreset constructions of [27, 26, 12, 19] with a multiplicative overhead of $O(\log n)$. Our second theorem is a more generally applicable technique; it applies to all constructions that first compute a bicriterion solution and then sample points according to the bicriterion solution. The constructions of [27, 26, 12, 19] follow this outline, and we are unaware of any constructions which do not. The theorem yields immediate corollaries as well as reducing certain streaming coreset problems to that of constructing an offline coreset.

Theorem 2.2.2. *Given an offline algorithm that constructs a (k, ϵ)-coreset consisting of $S = S(n, k, \epsilon, \delta)$ points with probability $1 - \delta$ by sampling points based on a bicriterion solution, there exists a streaming algorithm requiring the storage of $O(S \log n)$ points that maintains a (k, ϵ)-coreset on an insertion-only stream.*

Proof Sketch. The known offline coreset constructions start with a bicriterion solution of $O(k)$ points. We modify the algorithm of [9] to output $O(k \log n)$ centers; this is trivial since the final step of the algorithm of [9] is to take the $O(k \log n)$ centers stored in memory and reduce them to exactly k centers to provide a solution. Our first modification to the original algorithm is thus to simply remove this final step, but we must also keep a datastructure storing $\log(1/\epsilon)$ intermediate states of these $O(k \log n)$ centers. See Section 2.7 for a precise description of our modification and the sampling method, applied to the construction of [19] as an example (but equally applicable to [27, 26, 12]). As the high-level idea, since the bicriterion given to the offline construction consists of $O(k \log n)$ centers instead of exactly k, the number of additional points taken in the coreset increases by a factor of $O(\log n)$. $\square$

Two important corollaries include:

1. Using the result of [19], we obtain a streaming algorithm that maintains a (k, ϵ)-coreset with negative weights for metric k-median requiring the storage of $O(\epsilon^{-2} k \log n)$ points.

2. Given a $O(k \cdot \text{poly}(\epsilon, \log n, \log(1/\delta)))$ point (k, ϵ)-coreset, we would obtain a streaming algorithm that maintains a (k, ϵ)-coreset (with only positive weights) for metric k-median requiring the storage of $O(k \cdot \text{poly}(\epsilon, \log n, \log(1/\delta)))$ points. This differs from Theorem 2.2.1 in that the dependence on k is linear instead of $O(k \log k)$.

2.2.1 Definitions

We begin by defining a ρ-metric space, which is defined in full as Definition 2.5.1. Briefly, let X be a set. If $D : X \times X \to [0, \infty)$ is a symmetric function such that for every $x, z \in X$ we have that $D(x, z) \leq \rho(D(x, y) + D(y, z))$ for every $y \in X$, when we call (X, D) a ρ-metric space. Note that this is a weakening of the triangle inequality, and at $\rho = 1$ we recover the definition of a metric space. All M-estimators can be re-cast for a certain constant value of ρ, and k-means is obtained with $\rho = 2$. This generality is therefore useful and working in this language allows us to naturally generalize our results to any M-estimator.

The k-median problem is, given an input set P and an integer $k \geq 1$, to find a set C

of k points that minimizes:

$$\sum_{p\in P} \min_{c\in C} D(p,c)$$

We use $\mathsf{OPT}_k(P)$ to denote this minimal value. As this is NP-Hard to compute, we settle for an approximation. The notion of a bicriterion approximation is well-known; we state a definition that suits our needs while also fitting into the definition of previous works.

Definition 2.2.3 ((α,β)-approximation)**.** *An (α,β)-approximation for the k-median clustering of a multiset P is a map $\pi : P \to B$ for some set B such that $\sum_{p\in P} w(p)D(p,\pi(p)) \leq \alpha \mathsf{OPT}_k(P)$ and $|B| \leq \beta k$.*

We now define a coreset:

Definition 2.2.4 ((k,ϵ)-coreset)**.** *A (k,ϵ)-coreset of a multiset P is a weighted set (S,v) with non-negative weight function v such that for every $Z \in \mathcal{X}^k$ we have $(1-\epsilon)\sum_{p\in P} D(p,Z) \leq \sum_{s\in S} v(s)D(s,Z) \leq (1+\epsilon)\sum_{p\in P} D(p,Z)$.*

Coresets with arbitrary weight functions (i.e. with negative weights allowed) have been considered [FL11, etc]. However, computing approximate solutions on these coresets in polynomial-time remains a challenge, so we restrict our definition to non-negative weight functions. This ensures that an approximate solution can be quickly produced. This implies a PTAS for Euclidean space and a polynomial-time $2\gamma(1+\epsilon)$-approximation for general metric spaces (where γ is the best polynomial-time approximation factor for the problem in the batch setting). This factor of $2\gamma(1+\epsilon)$ is well-known in the literature, see [10, 9, 24] for details.

2.2.2 Constant-Approximation Algorithm

Let P_i denote the prefix of the stream $\{p_1,\ldots,p_i\}$. The entire stream is then P_n. Consider the moment when the first i points have arrived, meaning that the prefix P_i is the current set of arrived points. The algorithm $\mathcal{A}$ of [9] provides an $(O(1),O(\log n))$-approximation of P_i in the following sense. Define $f_0 : \emptyset \to \emptyset$ as the null map, and define $B_i = \text{image}(f_i)$.

Upon receiving point p_i, algorithm $\mathcal{A}$ defines a map $f_i : B_{i-1} \cup \{p_i\} \to B_i$. We define $\pi_i : P_i \to B_i$ by $\pi_i(p_j) = f_i(f_{i-1}(\dots(f_j(p_j))\dots))$ for each $1 \le j \le i$. These mappings have an essential gaurantee stated in the following lemma.

Theorem 2.2.5 ([9]). *For every $1 \le i \le n$, after receiving P_i, Algorithm $\mathcal{A}(k, n, \delta)$ defines a function f_i such that with probability $1 - \delta$, using the above definition of π_i, the bound $\sum_{p \in P_i} D(p, \pi_i(p)) \le \alpha \textsf{OPT}_k(P_i)$ holds. The algorithm deterministically requires the storage of $O(k(\log n + \log(1/\delta)))$ points.*

2.2.3 Offline Coreset Construction

We briefly describe the offline coreset construction. The proof of correctness can be found in Sections 2.4 and 2.5. It is this construction that we will maintain in the streaming setting.

The sensitivity of a point $p \in P$ is defined as:

$$s(p) = \max_{Z \in \mathcal{X}^k} \frac{D(p, Z)}{\sum_{q \in P} D(q, Z)}$$

Notice that $0 \le s(p) \le 1$. We give an upper bound $s'(p) \in [s(p), 1]$. Define the total sensitivity $t = \sum_{p \in P} s(p)$. Likewise, we give an upper bound $t' \ge t$ where $t' = \sum_{p \in P} s'(p)$ and will show that $t = O(k)$. The sampling probability distribution at point p is set to $s'(p)/t'$. We take an i.i.d. sample from P of size m for any $m \ge ct'\epsilon^{-2}(\log n \log t' + \log(1/\delta))$ where c is a constant.

Let R be the union of these m i.i.d. samples, and then define a weight function $v : R \to [0, \infty)$ where $v(r) = (|R|s'(r))^{-1}$. It is proven as one of our main theorems (Theorem 2.5.6) that the weighted set (R, v) is a (k, ϵ)-coreset for P.

2.2.4 Bounding the Sensitivity

Consider the prefix P_i which is the input after the first i points have arrived. Using Algorithm $\mathcal{A}$ we obtain an (α, β)-approximation π_i where $\alpha = O(1)$ and $\beta = O(\log n)$.

Recall that B_i is the image of this approximation, i.e. $B_i = \text{image}(\pi_i(P_i))$.

Running an offline (γ, λ)-approximation algorithm on B_i, we obtain a multiset C_i of at most λk distinct points. Let p' denote the element of C_i nearest to $\pi_i(p)$ (this is the element of C_i that p gets mapped to when we pass from $P_i \to B_i \to C_i$). The following lemma implies that $\sum_{p \in P} w(p)D(p, p') \leq \bar{\alpha}\mathsf{OPT}(P)$ where $\bar{\alpha} = \rho\alpha + 2\rho^2\gamma(\alpha+1)$. This is an observation used widely in the literature [10, 9], but we include a proof for completeness.

Lemma 2.2.6. *Let B be a (α, β)-approximation of P, and let C be a (γ, λ)-approximation of B. Then C is a $(\rho\alpha + 2\rho^2\gamma(\alpha + 1), \lambda)$-approximation of A.*

Proof. Let $\pi : P \to B$ be the (α, β)-approximation of P and let $t : B \to C$ be the (γ, λ)-approximation of B. In the following, all sums will be taken over all $p \in P$. The hypotheses state that $\sum D(p, \pi(p)) \leq \alpha\mathsf{OPT}(P)$ and $\sum D(\pi(p), t(\pi(p))) \leq \gamma\mathsf{OPT}(B)$. Let P^* be an optimal clustering of P, that is $\sum D(p, P^*) = \mathsf{OPT}(P)$. Then $\frac{1}{2}\mathsf{OPT}(B) \leq \sum D(\pi(p), P^*) \leq \rho \sum(D(\pi(p), p) + D(p, P^*)) \leq \rho(\alpha + 1)\mathsf{OPT}(P)$. The factor of $\frac{1}{2}$ comes from the fact that $\mathsf{OPT}(B)$ is defined using centers restricted to B (see [24] for details). We now write $\sum D(p, t(\pi(p))) \leq \rho \sum(D(p, \pi(p)) + D(\pi(p), t(\pi(p)))) \leq (\rho\alpha + 2\rho^2\gamma(\alpha + 1))\mathsf{OPT}(P)$ as desired. $\square$

We now prove the following lemma which gives us our sampling probability $s'(p)$. Recall that for the construction to succeed, the sampling probability $s'(p)$ must be at least the sensitivity $s(p)$ (defined in the previous subsection). Since we focus on a single iteration, we drop subscripts and write $C = C_i$ and $P = P_i$. Let $p \mapsto p'$ be an $(\bar{\alpha}, \lambda)$-approximation of P. Define $P(p) = \{q \in P : q' = p'\}$ to be the cluster containing p.

Lemma 2.2.7. *Let the map $p \mapsto p'$ define an $(\bar{\alpha}, \lambda)$-approximation for the k-median clustering of P. For every point $p \in P$:*

$$s(p) \leq \frac{\rho\bar{\alpha}D(p, p')}{\sum_{q \in P} D(q, q')} + \frac{\rho^2(\bar{\alpha} + 1)}{|P(p)|}$$

Proof. For an arbitrary $Z \in \mathcal{X}^k$ we need to provide a uniform bound for

$$\begin{aligned}\frac{D(p,Z)}{\sum_{q\in P} D(q,Z)} &\le \frac{\rho D(p,p')}{\sum_{q\in P} D(q,Z)} + \frac{\rho D(p',Z)}{\sum_{q\in P} D(q,Z)} \\ &\le \frac{\bar{\alpha}\rho D(p,p')}{\sum_{q\in P} D(q,q')} + \frac{\rho D(p',Z)}{\sum_{q\in P} D(q,Z)}\end{aligned} \tag{2.1}$$

where the second inequality holds because $\sum_{q\in P} D(q,q') \le \bar{\alpha}\mathsf{OPT}(P) \le \sum_{q\in P} D(q,Z)$. To bound the last term, recall that $q' = p'$ for all $q \in P(p)$ so:

$$\begin{aligned}D(p',Z)|P(p)| = \sum_{q\in P(p)} D(p',Z) &= \sum_{q\in P(p)} D(q',Z) \\ &\le \rho \sum_{q\in P(p)} (D(q',q) + D(q,Z)) \\ &\le \rho \sum_{q\in P} D(q',q) + \rho \sum_{q\in P(p)} D(q,Z) \\ &\le \rho\bar{\alpha} \sum_{q\in P} D(q,Z) + \rho \sum_{q\in P(p)} D(q,Z) \\ &\le \rho(\bar{\alpha}+1) \sum_{q\in P} D(q,Z)\end{aligned}$$

Dividing by $|P(p)| \sum_{q\in P} D(q,Z)$ gives

$$\frac{D(p',Z)}{\sum_{q\in P} D(q,Z)} \le \frac{\rho(\bar{\alpha}+1)}{|P(p)|}$$

Substituting this in (2.24) yields the desired result. □

We therefore define our upper bound $s'(p)$ as in the lemma. An immediate but extremely important consequence of Lemma 2.2.7 is that $t' = \sum_{p\in P} s'(p) = \rho\bar{\alpha} + \rho^2(\bar{\alpha} + 1)k \le 3\rho^2\bar{\alpha}k$. This can be seen by directly summing the formula given in the lemma.

2.2.5 Streaming Algorithm

We now state Algorithm 1, which we then prove maintains a coreset. To use Lemma 2.2.7 to determine $s'(p)$, we will compute the cluster sizes $|P(p)|$ and estimate the clustering cost $\sum_{q \in P} D(q, q')$. We must bound the clustering cost from below because we need an upper-bound of $s(p)$. On Line 9, L is an estimate of the cost of clustering P to the centers C. On Line 4, c is the absolute constant used in Theorem 2.5.6.

Algorithm 1: Input: stream of n points in a ρ-metric space, $\epsilon > 0$, $k \in \mathbb{N}$, maximum failure probability $\delta > 0$

```
1  **Initilization:**
2  $R \leftarrow \emptyset$
3  $t' \leftarrow \rho\bar{\alpha} + \rho^2(\bar{\alpha}+1)k$
4  $x \leftarrow 2c\epsilon^{-2}(\log n \log t' + \log(1/\delta))$
5  Initialize $\mathcal{A}(k, n, \delta)$
6  **Update Process: after receiving** $p_i$
7  $(B_i, f_i) \leftarrow \mathcal{A}.update(p_i)$
8  $C \leftarrow$ an $(\gamma, \lambda)$-approximation of $B_i$
9  $L \leftarrow D(p_i, C) + \bar{\alpha}^{-1}(1+\epsilon)^{-1}\sum_{r \in R} v(r)D(r, C)$
10 **for** $r \in R$ **do**
11     $\pi_i(s) \leftarrow f_i(\pi_{i-1}(r))$
12     $z(r) \leftarrow \frac{\rho\bar{\alpha}D(p,p')}{L} + \frac{\rho^2(\bar{\alpha}+1)}{|P(p)|}$
13     $s'(r) \leftarrow \min(s'(r), z(r))$
14 **for** $r \in R$ **do**
15     **if** $u(r) > xs'(r)$ **then**
16         Delete $r$ from $R$
17 $u(p_i) \leftarrow$ uniform random from $[0, 1)$
18 **if** $u(p_i) \leq xs'(p_i)$ **then**
19     Add $p_i$ to $R$
20     $\pi_i(p_i) \leftarrow f_i(p_i)$
21 **Query Process:**
22 **for** *each* $r \in R$ **do**
23     $v(r) \leftarrow (|R|s'(r))^{-1}$
```

Algorithm 1 outputs (R, v) such that point p is sampled with probability $xs'(p)$ where x is defined on Line 4. For each p that has arrived, the value of $s'(p)$ is non-increasing (notice that it is defined as the minimum of itself and a new value on Line 13), so it is possible to maintain this in the streaming setting since once the deletion condition on Line 15 becomes satisfied, it remains satisfied forever.

We now proceed with the proof of Theorem 1. Since the probability of storing point p is $xs'(p)$, the expected space of Algorithm 1 is xt'. By Lemma 2.2.7 that implies $t' \leq 3\rho^2\bar{\alpha}k$, we then bound the expected space as $2c\epsilon^{-2}(\log n \log t + \log(1/\delta))(3\rho^2\bar{\alpha}k)$.

Simplifying notation by defining an absolute constant $\tilde{c}$ (a function of c and $\bar{\alpha}$), we write this expected space as $\tilde{c}\rho^2\epsilon^{-2}k\log(\rho k)\log n\log(1/\delta)$. By a Chernoff bound, the high-probability gaurantee follows by replacing $\tilde{c}$ with $2\tilde{c}$.

Proof of Theorem 1. The proof of Theorem 1 can be divided into the following pieces:

1. For correctness (to satisfy the bound given in Lemma 2.2.7, we must show that $L \leq \sum_{p\in P_i} D(p,p')$. For space, it is important that L is not too small. In particular, the space grows as $1/L$. We show that L is a ϵ-approximation of the true cost.
2. The value of $|P(p)|$ can be computed exactly for every p. This is needed on Line 12.
3. The construction of Algorithm 1 that samples p with probability $xs'(p)$ can be processed to be identical to the offline construction of Subsection 2.2.3 that takes an i.i.d. sample of size xt' from the distribution where p is given sampling probability $s'(p)/t'$.

1: To lower bound the clustering cost, inductively assume that we have a (k,ϵ)-coreset S_{i-1} of P_{i-1}. Note that p_i is a (k,ϵ)-coreset of itself (in fact a $(k,0)$-coreset of itself), so $S_{i-1}\cup\{p_i\}$ is a (k,ϵ)-coreset of P_i. Let L be the cost of clustering $S_{i-1}\cup\{p_i\}$ to C. Therefore the cost of clustering P_i to C is in the interval $[(1-\epsilon)L,(1+\epsilon)L]$. Recall that the upper bound on $s(p)$ from Lemma 2.2.7 is:

$$\frac{\rho\bar{\alpha}D(p,p')}{\sum_{q\in P} D(q,q')} + \frac{\rho^2(\bar{\alpha}+1)}{|P(p)|}$$

By using L in place of the true cost $\sum_{p\in P_i} D(p,p')$ for defining $s'(p)$, the value of $t' = \sum_{p\in P_i} s'(p)$ increases to at most $\rho\bar{\alpha}\left(\frac{1+\epsilon}{1-\epsilon}\right) + \rho^2(\bar{\alpha}+1)\lambda k = O(\rho^4 k)$. Here there is no dependence on ϵ since we assume $\epsilon \leq 1/2$, so $\frac{1+\epsilon}{1-\epsilon}$ is bounded by an absolute constant.

2: Computing $|P(p)|$ is straightforward. Define $w(b) = |\{p\in P : \pi(p) = b\}|$ and then let $h : B \to C$ be the (γ,λ)-approximate clustering. Then $|P(p)| = \sum_{b\in h^{-1}(p')} w(b)$.

3: In Algorithm 1, we sample point p with probability $s'(p)$ to maintain a set M of non-deterministic size. We now argue that this can be converted to the desired coreset, where an i.i.d. sample of size m is taken from the distribution s'/t'. First, by a Chernoff bound, $|R| \geq E[|R|]/2$ with probability $1 - exp(-E[|R|]/8)$. Since $E[|R|] = xt' = (2c\epsilon^{-2}\log(\rho k)\log n \log(1/\delta)) \cdot (\rho\bar{\alpha} + \rho^2(\bar{\alpha}+1)k) = \Omega(\log(n))$, we have that $|R| \geq E[|R|]/2$ with probability $1 - O(1/n)$. Then by the union bound, this inequality holds true at each of the n iterations of receiving a point throughout the entire stream. Recall that for the offline coreset construction outlined in Subsection 2.2.3 to hold, we need an i.i.d. sample of at least $m = ct'\epsilon^{-2}(\log n \log t' + \log(1/\delta))$. By Lemma 2.2.7 $t' = \rho\bar{\alpha} + \rho^2(\bar{\alpha}+1)k$, and so by plugging in values we see that $E[|R|] = xt' \geq 2m$. Having that $|R| \geq m$ with probability $1 - O(1/n)$, it is well-known (see [16] for example) that this can be converted to the required i.i.d. sample.

□

2.3 Preliminaries for Offline Coreset Construction

2.3.1 Query space

In our framework, as in [19], we are given a finite input set P of items that are called *points*, a (usually infinite) set Q of items that are called *queries*, and a *cost* function f that maps each pair of a point in P and a query in Q to a real number $f(p, q)$. The cost of the set P to this query is the sum over all the costs,

$$\bar{f}(P, q) := \sum_{p \in P} f(p, q).$$

More generally, each input point might be given a multiplicative weight $w(p)$, and the overall cost of each point is then reweighed, so that

$$\bar{f}(P, w, q) = \sum_{p \in P} w(p) f(p, q).$$

The tuple (P, w, f, Q) is thus define our input problem and we call it a *query space*. In the case that the points are unweighted we can simply define $w(p) = 1$ for every $p \in P$.

However, for the following sections (but not in later sections), we assume that the weights in w are non-negative. We might also assume that w is a distribution vector, i.e., that its weights sum to one and thus represent a distribution. The cost $\bar{f}(P, w, q)$ is then the expected value of $f(p, q)$ for a point that is sampled at random from P. For the unweighted case we thus have $w(p) = 1/n$ for each point, and

$$\bar{f}(P, w, q) = \frac{1}{n} \sum_{p \in P} f(p, q)$$

is the average cost per point. The cost function f is usually also scaled as will be explained later, to have values $f(p, q)$ between 0 and 1, or -1 to 1.

Example 1: Consider the following problem where the input is a set P of n points in $\mathbb{R}^d$. Given a ball $B = B(c, r)$ of radius r that is centered in $c \in \mathbb{R}^d$, we wish to compute the fraction of input points that are covered by this ball, $\frac{|P \cap B|}{|P|}$. More generally, the query is a set of k balls, and we wish to compute the fraction of points in P that are covered by the union of these balls. In this case, each input point $p \in P$ has a weight $w(p) = 1/n$, the set Q of queries is the union over every k balls in $\mathbb{R}^d$,

$$Q = \left\{ B(c_1, r_1) \cup \cdots \cup B(c_k, r_k) \mid \forall i \in [k] : c_i \in \mathbb{R}^d, r_i \geq 0 \right\}, \tag{2.2}$$

and the cost $f(p, q)$ for a query $q = \{B_1 \cup \cdots \cup B_k\} \in Q$ is either $1/n$ if p is inside one of the balls of q, and 0 otherwise. The overall cost $\bar{f}(P, w, q) = \sum_{p \in P} f(p, q)$ is thus the fraction of points of P that are covered by the union of these k balls.

The motivation of defining query spaces is usually to solve some related optimization problem, such as the query q that minimizes the cost $\bar{f}(P, w, q)$. In this case, the requirement to approximate every query in Q is too strong, and we may want to approximate only the optimal query in some sense. To this end, we may wish to replace the set Q by a function that assigns a different set of queries $Q(S)$ for each subset S of P. For the correctness of our results, we require that this function Q will be monotonic in the following sense: if T is a subset of S then $Q(T)$ must be a subset of $Q(S)$. If we wish to have a single set $Q(P)$ of queries as above, we can simply define $Q(S) := Q(P)$ for every subset S of P, so that the desired monotonic property will hold.

Example 2: For the set Q in (2.2), define $Q(S)$ to be the set of balls in $\mathbb{R}^d$ such that the center of each ball is a point in S. More generally, we can require that the center of each ball will be spanned by (i.e., linear combination of) at most 10 points from S.

We now conclude with the formal definitions for the above discussion.

Definition 2.3.1 (weighted set)**.** *Let S be a subset of some set P and $w : P \to \mathbb{R}$ be a function. The pair (S, w) is called a* weighted set*. If $\sum_{p\in P} w(p) = 1$ and $w(p) \geq 0$ for every $p \in P$ then w is called a* distribution*.*

Definition 2.3.2 (query space)**.** [19]*. Let Q be a function that maps every set $S \subseteq P$ to a corresponding set $Q(S)$ called* queries*, such that $Q(T) \subseteq Q(S)$ for every $T \subseteq S$. Let $f : P \times Q(P) \to \mathbb{R}$ be a* cost function*. The tuple (P, w, Q, f) is called a* query space*. We denote the* cost *of a query $q \in Q(P)$ by*

$$\overline{f}(P, w, q) := \sum_{p\in P} w(p)f(p, q).$$

Instead of using weights (or non-uniform distribution) we can define $g(p, q) = n \cdot w(p)f(p, q)$ and $u(p) = 1/n$ for every $p \in P$, i.e., a uniform distribution, to get the same expectation

$$g(P, u, q) = \overline{f}(P, w, q) = \sum_{p\in P} w(p)f(p, q).$$

However, when we try to approximate this expected value e.g. using uniform or non-uniform sampling, bounds such as the Hoeffding bound implies that the size of the sample depends on the range of the function f or g, but not on their corresponding distribution (weights). Since we can assign many pairs of distributions and functions that yield the same expected value $g(P, u, q)$, $f(P, w, q)$ and so on, the natural question is what pair will minimize the range of the corresponding function?

In the next sections we develop tools that aim to answer this question.

2.3.2 (ε, ν)-Approximation

Consider the query space (P, w, Q, f) in Example 1 for $k = 1$, and suppose that we wish to compute a set $S \subseteq P$, such that, for every given ball B, the fraction of points in P that are covered by B, are approximately the same as the fraction of the points that are covered in S, up to a given small additive percentage error $\varepsilon > 0$. That is, for every ball B,

$$\left| \frac{|P \cap B|}{|P|} - \frac{|S \cap B|}{|S|} \right| \leq \varepsilon.$$

By defining the weight $u(p) = 1/|S|$ for every $p \in S$, this implies that for every query (ball) $q = B$,

$$\left| \bar{f}(P, w, q) - \bar{f}(S, u, q) \right| = \left| \sum_{p \in P} \frac{1}{|P|} \cdot f(p, q) - \sum_{p \in S} \frac{1}{|S|} \cdot f(p, q) \right| = \left| \frac{|P \cap B|}{|P|} - \frac{|S \cap B|}{|S|} \right| \leq \varepsilon.$$

A weighted set (S, u) that satisfies the last inequality for every query $q \in Q(S)$ is called an *ε-approximation* for the query space (P, w, f, Q). Note that the above example assumes that the maximum answer to a query q is $f(p, q) \leq 1$. Otherwise, the error guaranteed by an ε-approximation for a query q is $\varepsilon \max_{p \in P} |f(p, q)|$.

The above inequalities implies that if a ball covers a fraction of at least ε points from P (i.e., at least εn points), then it must cover at least one point of P. If we only ask for this (weaker) property from S then S is called an *ε-net*. To obtain the new results of this paper, we use a tool that generalizes the notion of ε-approximation and ε-net, but less common in the literature, and is known as *(ε, ν)-approximation* [31].

By letting $a = \bar{f}(P, w, q)$ and $b = \bar{f}(S, u, q)$, an ε-approximation implies that $|a - b| \leq \varepsilon$ for every query q, and ε-net implies that $b > 0$ if $a \geq \varepsilon$. Following [32], we define below a distance function that maps a positive real $|a - b|_\nu$ for each pair of positive real numbers

a and b. A specific value ν will imply $|a-b|_\nu \leq \varepsilon$, i.e., that S is an ε-approximation, and a different value of ν will imply that S is an ε-net for P. This is formalized as follows.

Definition 2.3.3 ((ε, ν)-approximation [32]). *Let $\nu > 0$. For every $a, b \geq 0$, we define the distance function*

$$|a-b|_\nu = \frac{|a-b|}{a+b+\nu}.$$

Let (P, w, Q, f) be a query space such that w is a distribution and $f(p,q) \in [0,1]$ for every $p \in P$ and $q \in Q(p)$. For $\varepsilon > 0$, a weighted set (S, u) is an (ε, ν)-approximation *for this query space, if for every $q \in Q(S)$ we have*

$$\left|\overline{f}(P, w, q) - \overline{f}(S, u, q)\right|_\nu \leq \varepsilon. \tag{2.3}$$

Corollary 2.3.4 ([32, 28]). *Let $a, b \geq 0$ and $\tau, \nu > 0$ such that $|a-b|_\nu \leq \tau$. Let $\varepsilon > 0$. Then*

(i) (ε-sample/approximation). *If $\tau = \varepsilon/4$ and $\nu = 1/4$ then*

$$|a-b| \leq \varepsilon.$$

(ii) (ε-net). *If $\tau = 1/4$ and $\nu = \varepsilon$ then*

$$(a \geq \varepsilon \quad \Rightarrow \quad b > 0).$$

(iii) (Relative ε-approximation). *Put $\mu > 0$. If $\nu = \mu/2$ and $\tau = \varepsilon/9$ then*

(1) $a \geq \mu \quad \Rightarrow \quad (1-\varepsilon)a \leq b \leq (1+\varepsilon)a.$

(2) $a < \mu \quad \Rightarrow \quad b \leq (1+\varepsilon)\mu.$

2.3.3 Constructing ε-Approximations

Unlike the notion of coresets in the next section, the idea of ε-approximations is known for decades [39, 34]. In particular, unlike coresets, ε-approximations and (ε, ν)-approximations in general, can be constructed using simple uniform random sampling of the points. The size of the sample depends linearly on the complexity of the queries in the sense soon to be defined in this section.

Intuitively, and in most practical cases, including the examples in this thesis, this complexity is roughly the number of parameters that are needed to define a single query. For example, a ball in $\mathbb{R}^d$ can be defined by $d+1$ parameters: its center $c \in \mathbb{R}^d$, which is defined using d numbers and the radius $r > 0$ which is an additional number. A query of k balls is similarly defined by $k(d+1)$ parameters. However, by the set theory we have $|R| = |R^m|$ for every integer $c \geq 1$, which means that we can encode every m integers to a single integer. We can thus always reduce the number of parameters that are needed to define a query from m to 1 by redefining our cost function f and re-encoding the set of queries.

There are also natural examples of query spaces whose cost function f is defined by one parameter, but the size of the sampling needed for obtaining an ε-approximation is unbounded, e.g., the query space where $f(p,q) = \text{sign}(\sin(pq))$ and $P = Q = \mathbb{R}^d$, where $\text{sign}(x) = 1$ if $x > 0$ and 0 otherwise. Hence, a more involved definition of complexity is needed as follows.

While the number of subsets from a given set of n points is 2^n, its can be easily verified that the number of subsets that can be covered by a ball in $\mathbb{R}^d$ is roughly $n^{O(d)}$. The exponent $O(d)$ is called the *VC-dimension* of the family of balls in $\mathbb{R}^d$. The following definition is a simple generalization by [19] for query spaces where the query set is a function and not a single set. The original definition of pseudo-dimension can be found e.g. in [32] and is very similar to the definition of VC-dimension given by [39] as well as many other similar measures for the complexity of a family of shapes.

Definition 2.3.5 (dimension [19]). *For a query space (P, w, Q, f) and $r \in [0, \infty)$ we define*

$$\operatorname{range}(q, r) = \{p \in P \mid w(p) \cdot f(p, q) \leq r\}.$$

The dimension *of (P, w, Q, f) is the smallest integer d such that for every $S \subseteq P$ we have*

$$\left| \{\operatorname{range}(q, r) \mid q \in Q(S), r \in [0, \infty)\} \right| \leq \left(\frac{|S|}{d}\right)^d.$$

The main motivation of the above definition, is that it tells us how many samples we need to take uniformly at random from the input set P, to get an (ε, ν)-approximation (S, u) as follows. The original and famous version for the theorem below is by [35] and the bound was improves and proven to be optimal by [32]. Both versions are for the case of a constant function Q (i.e., a set of queries or ranges). In [19] it was shown that the same bounds hold for the case where Q is a function, which is used e.g. to construct coresets of size independent of the dimension of the input points as will be shown later.

Theorem 2.3.6 ([32, 19]). *Let (P, w, Q, f) be a query space of dimension d such that w is a distribution and $f : P \to [0, 1]$, and let $\varepsilon, \delta, \nu > 0$. Let S be a random sample from P, where every point $p \in P$ is sampled i.i.d. with probability $w(p)$. Let u be a uniform distribution over S, i.e., $u(p) = \frac{1}{|S|}$ for every $p \in S$. Then there is a constant $c > 0$ such that if*

$$|S| \geq \frac{c}{\varepsilon^2 \nu}\left(d \log\left(\frac{1}{\nu}\right) + \log\left(\frac{1}{\delta}\right)\right)$$

then, with probability at least $1 - \delta$, (S, u) is an (ε, ν)-approximation *for the query space (P, w, Q, f).*

2.4 Improved Coreset Framework

2.4.1 Improved (ε, ν)-approximations

In this section we show the first technical result of this part: that the additive error $\varepsilon \max_{p \in P} |f(p,q)|$ in (2.3) can be replaced by ε, and the assumption $\sum_{i=1} w_i = 1$ in Theorem 2.3.6 can be removed. The condition for this to work is that the total importance value t, as defined below, is small. More precisely, the required sample size will be near-linear with t. Also, the uniform sampling in Theorem 2.3.6 will be replaced by non-uniform sampling with a distribution that is proportional to the importance of each point. This result is essentially a generalization and significant simplification of the framework in [19] that used ε-approximations for constructing coresets. Maybe more importantly: using the idea of (ε, ν)-approximations we are able to show that the sample size is near linear in t while in [19] it is quadratic in t. For some applications, such an improvement means turning a theoretical result into a practical result, especially when t is close to $\sqrt{n}$.

We define the *importance* of a point as the maximum absolute weighted cost $w(p)f(p,q)$ of a point p, over all the possible queries q, i.e,

$$s(p) := w(p) \max_{q \in Q(P)} f(p,q),$$

and hope that this sum is small (say, constant or $\log n$), in other words, that not *all* the points are very important. More precisely, if the sum $t = \sum_{p \in P} s(p)$ of these costs is 1, then we prove below that a new query space (P, w', Q, f') can be constructed, such that an (ε, ν)-approximation (S, u) for the new query space would imply

$$\left|\overline{f}(P, w, q) - \overline{f}(S, u, q)\right|_{\nu} \leq \varepsilon. \tag{2.4}$$

That is, the additive error in (2.3) is reduced as desired.

The new query space (P, w', Q, f') is essentially a re-scaling of the original weights by their importance, $w'(p) = s(p)$. To make sure that the cost of each query will still be the same, we need to define f' such that $w(p)f(p,q) = w'(p)f'(p,q)$. This implies $f'(p,q) := w(p)f(p,q)/s(p)$. While the new cost $\bar{f}'(P,w',q)$ is the same as the old one $\bar{f}(P,w,q)$ for every query q, the maximum value of $|f'(p,q)|$ is 1, by definition of $s(p)$, even if $|f(p,q)|$ is arbitrarily large. Hence, the additive error $\varepsilon|f'(p,q)|$ in (2.3) reduced to ε in (2.4).

More generally, an $(\varepsilon, \nu/t)$-approximation for (P, w', Q, f') would yield (2.4). Using the uniform sample construction of Theorem 2.3.6, this implies that to get (2.4) we need to increase the sample size by a factor that is nearly linear in t.

A crucial difference between (2.12) and 2.3 is that w is not necessarily a distribution and its weights are unbounded.

Theorem 2.4.1. *Let*

- (P, w, Q, f) *be a query space where* f *and* w *are non-negative functions.*
- $s : P \to (0, \infty)$ *such that* $s(p) \geq w(p) \max_{q \in Q(P)} f(p,q)$.
- $t = \sum_{p \in P} s(p)$.
- $w' : P \to (0, 1]$ *be a distribution such that* $w'(p) := s(p)/t$.
- $f' : P \times Q(P) \to [0,1]$ *be defined as* $f'(p,q) := \dfrac{w(p)f(p,q)}{s(p)}$.
- (S, u) *be an* $(\varepsilon, \nu/t)$-approximation *for* (P, w', Q, f').
- $u'(p) = u(p) \cdot \frac{w(p)}{w'(p)}$ *for every* $p \in S$.

Then for every $q \in Q(S)$,

$$|\overline{f}(P, w, q) - \overline{f}(S, u', q)|_\nu \leq \varepsilon. \tag{2.5}$$

Proof. Put $q \in Q(S)$.

$$
\begin{aligned}
|\overline{f}(P,w,q) - \overline{f}(S,u',q)|_\nu &= \left|\sum_{p\in P} w(p)f(p,q) - \sum_{p\in S} \frac{u(p)w(p)}{\frac{s(p)}{t}} \cdot f(p,q)\right|_\nu &(2.6)\\
&= \left|t\sum_{p\in P} \frac{s(p)}{t}\cdot w(p)\cdot \frac{f(p,q)}{s(p)} - t\sum_{p\in S} u(p)\cdot w(p)\cdot \frac{f(p,q)}{s(p)}\right|_\nu \\
&= \left|t\sum_{p\in P} w'(p)\cdot f'(p,q) - t\sum_{p\in S} u(p)\cdot f'(p,q)\right|_\nu &(2.7)\\
&= \left|\sum_{p\in P} w'(p)\cdot f'(p,q) - \sum_{p\in S} u(p)\cdot f'(p,q)\right|_{\nu/t} &(2.8)\\
&\le \varepsilon, &(2.9)
\end{aligned}
$$

where (2.6) is by the definition of u', (2.7) is by the definition of f', (2.8) follows since for every $t > 0$ and $a, b \geq 0$,

$$|ta - tb|_\nu = \frac{|ta - tb|}{ta + tb + \nu} = \frac{|a-b|}{a + b + \nu/t} = |a - b|_{\nu/t}, \tag{2.10}$$

and (2.9) is by (2.3) and the definition of S in the sixth bullet of the statement. □

Corollary 2.4.2. *Let*

- (P, w, Q, f) *be a query space where* w *is a non-negative function and* $f : P \times Q(P) \to [a, b]$ *for* $a, b \in \mathbb{R}$ *where* $a \neq b$.
- $f(p, \perp) := b$ *for every* $p \in P$.
- $Q'(S) = Q(S) \cup \{\perp\}$ *for every* $S \subseteq P$.
- $s : P \to (0, \infty)$ *such that* $s(p) \geq w(p) \max_{q\in Q(P)} \left(\frac{f(p,q)-a}{b-a} + 1\right)$
- $t = \sum_{p\in P} s(p)$.
- $w' : P \to (0, 1]$ *be a distribution such that* $w'(p) := s(p)/t$.
- $f' : P \times Q'(P) \to [0, 1]$ *be defined as* $f'(p, q) := \dfrac{w(p)(g(p,q) - a)}{(b-a)s(p)}$

- (S, u) *be an* $(\varepsilon/(4(b-a)), 1/(4t))$-approximation *for* (P, w', Q', f') *and* $\varepsilon \in (0,1)$.
- $u'(p) = u(p) \cdot \frac{w(p)}{w'(p)}$ *for every* $p \in S$.

Then for every $q \in Q(S)$,

$$|\overline{f}(P, w, q) - \overline{f}(S, u', q)| \leq \varepsilon + \frac{\varepsilon |a|}{b-a}. \tag{2.11}$$

Moreover,

$$|\sum_{p\in P} w(p) - \sum_{p \in S} u'(p)| \leq \frac{\varepsilon}{b-a}.$$

Proof. Let $g : p \times Q'(P) \to [0,1]$ such that $g(p,q) = \frac{f(p,q)-a}{b-a}$. By letting $\nu = 1/4$, $\tau = \varepsilon/4$ and replacing Q with Q' in Theorem 2.4.1 and Corollary2.3.4 we obtain that for every $q \in Q'(S)$ we have

$$|\overline{g}(P, w, q) - \overline{g}(S, u', q)| \leq \varepsilon/(b-a). \tag{2.12}$$

Hence, for every $q \in Q'(S)$,

$$|\sum_{p \in P} w(p) \cdot \frac{f(p,q)-a}{b-a} - \sum_{p\in S} u'(p) \cdot \frac{f(p,q)-a}{b-a}| \leq \varepsilon/(b-a). \tag{2.13}$$

For the special case $q = \perp$, where $f(p,q) = b$ for every $p \in P$, we obtain

$$\begin{aligned} \left|\sum_{p\in P} w(p) - \sum_{p\in S} u'(p)\right| &= \left|\sum_{p\in P} w(p) \cdot \frac{f(p,q)-a}{b-a} - \sum_{p\in S} u'(p) \cdot \frac{f(p,q)-a}{b-a}\right| \\ &= \left|\sum_{p\in P} w(p) g(p,q) - \sum_{p\in S} u'(p) g(p,q)\right| \\ &= |\overline{g}(P, w, q) - \overline{g}(S, u', q)| \leq \varepsilon/(b-a). \end{aligned} \tag{2.14}$$

We thus have, for every $q \in Q(S)$,

$$
\begin{aligned}
&|\overline{f}(P,w,q) - \overline{f}(S,u',q)| = |\sum_{p\in P} w(p)f(p,q) - \sum_{p\in S} u'(p)\cdot f(p,q)| \\
&= (b-a)|\sum_{p\in P} w(p)\frac{f(p,q)-a}{b-a} - \sum_{p\in S} u'(p)\frac{f(p,q)-a}{b-a} + \sum_{p\in P}\frac{a(w(p)-u'(p))}{b-a}| \\
&\leq (b-a)|\sum_{p\in P} w(p)\frac{f(p,q)-a}{b-a} - \sum_{p\in S} u'(p)\frac{f(p,q)-a}{b-a}| + |a|\cdot|\sum_{p\in P}(w(p)-u'(p))| \quad (2.15)\\
&\leq \varepsilon + \frac{|a|\varepsilon}{b-a}, \quad (2.16)
\end{aligned}
$$

where (2.15) is by the triangle inequality, and (2.16) is by (2.13) and (2.14). □

Plugging Theorem 2.3.6 in Theorem 2.4.2 yields our main technical result that would imply smaller coresets in the next sections.

Theorem 2.4.3. *Let (P, w, Q, f) be a query space of dimension d and s, t be defined as in Theorem 2.4.2. Let $\varepsilon > 0$ and $\delta \in (0, 1)$. Let S be a random sample where every point $p \in P$ is sampled i.i.d. from P with probability $s(p)/t$. Assign a weight $u'(p) = \frac{tw(p)}{s(p)\cdot|S|}$ for every $p \in S$. There is a constant $c > 0$ such that if*

$$|S| \geq \frac{ct(b-a)^2}{\varepsilon^2}\left(d\log t + \log\left(\frac{1}{\delta}\right)\right),$$

then, with probability at least $1-\delta$,

$$|\overline{f}(P,w,q) - \overline{f}(S,u',q)| \leq \varepsilon,$$

and

$$|\sum_{p\in P} w(p) - \sum_{p\in S} u'(p)| \leq \frac{\varepsilon}{b-a}.$$

Proof. By Corollary2.4.2, it suffices to prove that (S, u') is an $(\varepsilon/(4(b-a)), 1/4t)$-approximation for (P, w', Q, f'). Indeed, since $\sum_{p\in P} w'(p) = 1$ and S is a random sample where $p \in P$ is sampled with probability $w'(p) = s(p)/t$, the weighted set (S, u) is, with probability at least $1 - \delta$, an $(\varepsilon/(4(b-a)), 1/(4t))$-approximation for (P, w', Q, f') by Theorem 2.3.6, for choosing a sufficiently large constant $c > 0$. $\square$

While Theorem 2.3.6 suggests to construct an ε-approximation simply by taking a uniform random sample from P, Theorem 2.4.3 requires us to take *non-uniform* sample where the distribution is defined by the importance $s(\cdot)$. Bounding these importances such that the sum $t = \sum_{p \in P} s(p)$ of importances will be small raise a new optimization problem that, as we will see in the next sections, might be not easy at all to solve. So what did we gain by proving Theorem 2.3.6 (or the framework of [19] in general)?

Most of the existing papers for constructing coresets essentially had to bound the total importance of the related problem anyway. However, without Theorem 2.3.6, their proofs also had to deal with complicated terms that involved ε and include sophisticated probability arguments. Essentially each paper had to re-prove in some sense the very involved proofs in [39, 32] that were researched for decades, as well as the mathematics behind the usage of Definition 2.3.3. Beside the complicated proofs, the final bounds, the dependency on ε and δ, as well as the coreset construction algorithm, were usually sub-optimal compared to Theorem 2.3.6. On the contrary, bounding the total importance allows us to focus on a deterministic results (no δ involved) and the terms $s(p)$ can be approximated up to constant factors (and not $(1+\varepsilon)$-factors). This is demonstrated in Section 2.5.

2.4.2 Improved Coreset Framework

While the result in the previous section can be used to improve the quality of (ε, ν)-approximation in general, its main motivation is to construct the more recent type of data structures that is sometimes called coresets.

As explained in Theorem 2.3.6, ε-approximation and (ε, ν)-approximation in general, can be easily computed using uniform random sampling. While ε-approximations are useful for hitting sets, where we wish to know how much points are covered by a set of shapes, they are less relevant for shape fitting, where we wish to approximate the sum of *distances* of the input points to a given query shape or model. The main reason is that in shape fitting the maximum contribution of a point to the overall cost (sum of covered points) is bounded by 1, while in general its distance to the shape is unbounded. Using the notation of the previous section: the importance of each point is the same, so Theorem 2.4.3 yields uniform sample S and the same error as Theorem 2.3.6.

Example: In the Euclidean k-means problem the input is a set P of n points in $\mathbb{R}^d$. For simplicity, assume that the set is unweighted, that is $w(p) = 1/n$ for every $p \in P$. A query in this context is a set of points (centers) in $\mathbb{R}^d$ so $Q(P)$ is the family (set) of k balls in $\mathbb{R}^d$. We denote the squared Euclidean distance from a point $p \in P$ to a center $c \in \mathbb{R}^d$ by $D(p, c) = \|p - c\|_2^2$. The distance from p to its closest center in $C = \{c_1, \cdots, c_k\}$ is then

$$D(p, C) := \min_{c \in C} D(p, c) = \min_{c \in C} \|p - c\|_2^2 .$$

By defining $g(p, C) = D(p, C)$ we obtain the query space (P, w, g, Q), where $\bar{g}(P, w, C)$ is the average squared distances to a given (query) set C of k centers. Our goal is to compute a weighted subset (S, u) such that $\bar{g}(P, w, C)$ will approximate the average squared distances $\bar{g}(S, u, C)$ for every set of centers.

Suppose that (S, u) is an ε-approximation of (P, w, Q, g). Then

$$|\bar{g}(P, w, C) - \bar{g}(S, u, C)| \le \varepsilon \max_{p \in P} g(p, C). \tag{2.17}$$

That is,

$$\left| \frac{1}{n} \sum_{p \in P} D(p, C) - \sum_{p \in P} u(p) D(p, C) \right| \le \varepsilon \max_{p \in P} D(p, C). \tag{2.18}$$

In other words, the additive error depends on the maximum distance between a point to a given center, which can be arbitrary large, unless we assume, say, that both the input points and centers are inside the unit cube. Theorem 2.4.3 would not improve this bound by itself, since the importance of each point, $\max_{c \in Q(P)} D(p, C)$ is unbounded.

In this section we wish to compute a weighted subset (S, u) that will approximate the average distance $\bar{g}(P, w, C)$ for every query, up to a *multiplicative* factor of $1 \pm \varepsilon$ without further assumptions. Such a set is sometimes called a coreset as follows.

Definition 2.4.4 (ε-coreset)**.** *For an error parameter $\varepsilon \in (0, 1)$, the weighted set (S, u) is an ε-coreset for the query space (P, w, Q, g) if $S \subseteq P$ and for every $q \in Q(S)$,*

$$(1 - \varepsilon)\bar{g}(P, w, q) \leq \bar{g}(S, u, q) \leq (1 + \varepsilon)\bar{g}(P, w, q).$$

An equivalent definition of an ε-coreset, is that the additive error $\varepsilon \max_{p \in P} g(p, C)$ in (2.17) is replaced by $\varepsilon \bar{g}(P, w, q)$. That is, for every $q \in Q(S)$

$$|\bar{g}(P, w, q) - \bar{g}(S, u, q)| \leq \varepsilon \bar{g}(P, w, q).$$

ε-coreset implies that not only the average, but also the sum of squared distances (or sum of costs, in general) are preserved up to $1 \pm \varepsilon$. Note also that simply multiplying the weight of each point in S by $(1 - \varepsilon)$ would yield a one sided error,

$$\bar{g}(P, w, q) \leq \bar{g}(S, u, q)| \leq \frac{1 + \varepsilon}{1 - \varepsilon} \cdot \bar{g}(P, w, q) = 1 + \frac{2\varepsilon}{1 - \varepsilon} \cdot \bar{g}(P, w).$$

If we assume in addition that $\varepsilon \in (0, 1 - 2/c)$ for some $c > 2$, then $1 - \varepsilon \geq 2/c$ and thus

$$\bar{g}(P, w, q) \leq \bar{g}(S, u, q)| \leq (1 + c\varepsilon)\bar{g}(P, w).$$

Hence, an ε/c-coreset (S, u) implies,

$$\bar{g}(P, w, q) \leq \bar{g}(S, u, q) \leq (1 + \varepsilon)\bar{g}(P, w, q). \tag{2.19}$$

For example, if $\varepsilon \in (0, 1/2)$, then an $(\varepsilon/4)$-coreset would yield (2.19), by substituting $c = 4$.

The main observation for getting the desired multiplicative approximation is that a multiplicative approximation of $\varepsilon\bar{g}(P, w, q)$, can be turned into an additive approximation of ε by replacing $g(p, q)$ with its scaled version

$$f(p, q) = \frac{g(p, q)}{\bar{g}(P, w, q)}.$$

To get a coreset for g as in (2.19), it suffices to have an ε-approximation for f. In addition, for many problems, while the importance $s(p)$ of a point in p is unbounded with respect to g, it is bounded with respect to f. We formalize this observation as follows.

Corollary 2.4.5. *Let (P, w, Q, g) be a query space for some non-negative function g, and define $f : P \times Q(P) \to \mathbb{R}$ such that for every $p \in P$ and $q \in Q(P)$ we have*

$$f(p, q) = \frac{g(p, q)}{\bar{g}(P, w, q)}.$$

Let (S, u') be a $\left(\frac{\varepsilon}{4}, \frac{1}{4t}\right)$-approximation for (P, w', Q, f') as defined in Theorem 2.4.1. Then (S, u') is an ε-coreset for (P, w, Q, g), i.e., for every $q \in Q(S)$

$$|\bar{g}(P, w, q) - \bar{g}(S, u, q)| \leq \varepsilon\bar{g}(P, w, q).$$

Proof. Put $q \in Q(S)$, $\tau = \varepsilon/4$ and $\nu = 1/4$. Applying Theorem 2.4.1 with ν yields

$$|\bar{f}(P, w, q) - \bar{f}(S, u', q)|_\nu \leq \tau.$$

Since $\bar{f}(P, w, q) = 1$, this implies

$$\left|1 - \frac{\bar{g}(S, u', q)}{\bar{g}(P, w, q)}\right|_\nu = |\bar{f}(P, w, q) - \bar{f}(S, u', q)|_\nu \leq \tau.$$

Substituting $a = 1$, and $b = \bar{f}(S, u', q)$ in Corollary 2.3.4(i) yields

$$\left|1 - \frac{\bar{g}(S, u', q)}{\bar{g}(P, w, q)}\right| \leq \varepsilon.$$

Multiplying by $\bar{g}(P, w, q)$ yields an ε-coreset as

$$|\bar{g}(P, w, q) - \bar{g}(S, u', q)| \leq \varepsilon \bar{g}(P, w, q).$$

□

Combining Corollary 2.4.5 and Theorem 2.3.6 yields the following theorem.

Theorem 2.4.6. *Let (P, w, Q, g) be a query space, where g is a non-negative function. Let $s : P \to (0, \infty)$ such that*

$$s(p) \geq \max_{q \in Q(P)} \frac{w(p) g(p, q)}{\sum_{p \in P} w(p) g(p, q)},$$

and $t = \sum_{p \in P} s(p)$. Let d be the dimension of (P, w', Q, f') as defined in Theorem 2.4.5. Let $c \geq 1$ be a sufficiently large constant, and let S be a random sample of

$$|S| \geq \frac{ct}{\varepsilon^2}\left(d \log t + \log\left(\frac{1}{\delta}\right)\right)$$

points from P, such that for every $p \in P$ and $q \in S$ we have $p = q$ with probability $s(p)/t$. Let $u'(p) = \frac{t \cdot w(p)}{s(p)|S|}$ for every $p \in S$. Then, with probability at least $1 - \delta$, (S, u') is an ε-coreset for (P, w, Q, g), i.e.,

$$\forall Q \in Q(S) : (1 - \varepsilon)\bar{g}(P, w, q) \leq \bar{g}(S, u', q) \leq (1 + \varepsilon)\bar{g}(P, w, q).$$

2.5 Application: Smaller and Generalized Coreset for k-Means

Definition 2.5.1 (ρ-metric space)**.** *Let X be a set, and $D : X^2 \to [0, \infty)$ be a function. Let $\rho > 0$. The pair (X, D) is a ρ-*metric *space if for every $(x, y, z) \in X^3$ we have*

$$D(x, z) \leq \rho(D(x, y) + D(y, z)).$$

*For $C \subseteq X$ we denote $D(x, C) := \min_{c \in C} D(x, c)$ assuming that such minimum exists. Moreover, for $\varepsilon > 0$, the pair (X, D) is a (ϕ, ε)-*metric *space if for every $(x, y, z) \in X^3$ we have*

$$|D(x, z) - D(y, z)| \leq \phi D(x, y) + \varepsilon D(y, z)). \tag{2.20}$$

Note that for every $x, y \in X$, and a center $c_y \in C$ that is closest to y, we have

$$D(x, C) = \min_{c \in C} D(x, c) \leq D(x, c_y) \leq \rho(D(x, y) + D(y, c_y)) = \rho(D(x, y) + D(y, C)). \tag{2.21}$$

Similarly,

$$D(x, C) - D(y, C) \leq D(x, c_y) - D(y, c_y) \leq \phi D(x, y) + \varepsilon D(y, c_y) = \phi D(x, y) + \varepsilon D(y, C), \tag{2.22}$$

where the first and last inequality is by the definition of $D(x, C)$ and c_y, and the second inequality is by (2.20). Switching x and y yields $D(y, C) - D(x, C) \leq \phi D(x, y) + \varepsilon D(x, C)$. Hence,

$$\begin{aligned} |D(x, C) - D(y, C)| &= \max\{D(x, C) - D(y, C), D(y, C) - D(x, C)\} \\ &\leq \phi D(x, y) + \varepsilon \max\{D(x, C), D(y, C)\}. \end{aligned}$$

Since $D(x,C) \le (1+\varepsilon)D(y,C) + \phi D(x,y)$ by (2.30),

$$\begin{aligned}2\phi D(x,c) < |D(x,C) - D(y,C)| &\le \phi D(x,y) + \varepsilon \max\{(1+\varepsilon)D(y,C) + \phi D(x,y), D(y,C)\}\\ &\le (\varepsilon+\phi)D(x,y) + \varepsilon(2+\varepsilon)D(y,C).\end{aligned}$$

A simple way to decide whether (D, X) is indeed a ρ-metric space is to use the following bound, known as Log-Log Lipschitz, that is usually easier to compute. The following lemma is very similar to [21, Lemma 2.1], where in the proof of (i) the constant 4 that appeared there is replaced here by $1/\varepsilon$.

Lemma 2.5.2 (Lemma 2.1(ii) in [21])**.** *Let $\tilde{D} : [0,\infty) \to [0,\infty)$ be a monotonic non-decreasing function that satisfies the following (Log-Log Lipschitz) condition: there is $r > 0$ such that for every $x > 0$ and $\Delta > 1$ we have*

$$\tilde{D}(\Delta x) \le \Delta^r \tilde{D}(x).$$

Let (X, dist) be a metric space, and let $D(x,y) = \tilde{D}(\mathrm{dist}(x,y))$ for every $x, y \in X$. Then (X, D) is a ρ-metric space for $\rho = \max\{2^{r-1}, 1\}$, and a (ϕ,ε)-metric space for every $\varepsilon \in (0,1)$ and $\phi = (r/\varepsilon)^r$.

For example, consider a metric space (X, dist) and the function $\tilde{D}(x) = x^2$ that corresponds to the squared distance $D(p,q) = \tilde{D}(\mathrm{dist}(p,q)) = (\mathrm{dist}(p,q))^2$. Note that (X, D) is not a metric space since the triangle inequality does not hold. However, for every $x > 0$ and $\Delta > 1$

$$\tilde{D}(x\Delta) = (x\Delta)^2 = \Delta^2 x^2 = \Delta^2 \tilde{D}(x).$$

Hence, by Lemma 2.5.2 $r = 2$, $\rho = 2^{r-1} = 2$, (D, X) is a 2-metric space and a (ϕ,ε)-metric space for $\phi = (2/\varepsilon)^2$.

To compute a coreset for a problem we need to decide what are the important points, or more formally, to use Theorem 2.4.6 we need to bound the importance $s(p)$ of each point $p \in P$. To do this, we usually need to solve another optimization problem that is usually related to computing the query with the minimal cost. For example, the bound of importance of a point in the k-mean problem, as will be defined later, is based on the optimal solution for the k-means problem. Unfortunately, this optimization problem is usually hard and the main motivation for constructing the coreset in the first place.

There are two leeways from this chicken-and-egg problem:

(i) Use the merge-and-reduce approach that reduces the problem of computing a coreset for a large set of n items to the problem of computing coresets for $\frac{n}{2|S|}$ small weighted (core)sets. Each input coreset $2|S|$ is reduced to a coreset of $|S|$ and merged with another such coreset, where $2|S|$ is the minimum size of input set that can be reduced to half using the given coreset construction. In this case, if this coreset construction takes time $f(|S|)$ than, since there are such $O(n)$ constructions, the overall running time will then be $O(n) \cdot f(|S|)$.

(ii) For problems such as k-means, it is NP-hard to compute the optimal solution, even for a small set of $n = O(k)$ points. Instead of computing an optimal solution, usually a constant factor approximation suffices for computing the importance of each point. Since for many problems such an approximation is also unknown or too slow to compute, (α, β)-approximation, (or bi-criteria, or bicriterion), can be used instead as explained below.

Suppose that the optimal solution for the k-means problem on a set is OPT_k. That is, there is a set C of k centers whose cost (sum of squared distances to the input points of P) is OPT_k, and there is no such set of smaller cost. Then a set $\tilde{C}$ is called an α-approximation if its cost is at most $\alpha \cdot OPT$. However, for many problems, even a rougher approximation would do: instead of using k centers to approximate OPT_k by a factor of α, we use βk centers, where each input point may be assigned for the nearest center. Note that the cost is still compared to OPT_k and not to $OPT_{\beta k}$. We define (α, β)-approximation formally below. For our purposes later, we generalize this common definition of (α, β)-approximation, and allow a point to be assigned to a different center than its nearest one, as long as the overall cost is small.

Definition 2.5.3 ((α, β)-approximation.)**.** *Let (X, D) be a ρ-metric space, and $k \geq 1$ be an integer. Let (P, w, Q, g) be a query space such that $P \subseteq X$, $Q(P) = \{C \subseteq X \mid |C| \leq k\}$, and $g : P \times Q(P)$ be a function such that*

$$g(p, C) = D(p, C) := \min_{c \in C} D(p, c). \tag{2.23}$$

Let $\alpha, \beta \geq 0$, $B \subseteq X$ such that $|B| \leq \beta k$ and $\mathcal{B} : P \to B$ such that

$$\sum_{p \in P} w(p) g(p, \{\mathcal{B}(p)\}) \leq \alpha \min_{q \in Q(P)} \bar{g}(P, w, q).$$

Then $\mathcal{B}$ is called an (α, β)-approximation *for (P, w, Q, g).*

For every $b \in B$, we denote by $P_b = \{p \in P \mid \mathcal{B}(p) = b\}$ the points that are mapped to the center b. We also denote $p' = \mathcal{B}(p)$ for every $p \in P$.

One of the main tools in our novel streaming algorithm is also a technique to update an (α, β)-approximation. However, due to memory limitations, our streaming algorithm cannot attach each point to its nearest center, but still the distances to the approximated centers is bounded. We thus generalize the definition of an (α, β)-approximation, which is usually a set B of size βk, to a *function* $\mathcal{B} : P \to B$ that assigns each input point to a (not necessarily closest) center in B, while the overall cost is still bounded.

Since an (α, β)-approximation yields a weaker result compared to a PTAS or α-approximation, it can usually be computed very quickly. Indeed, a very general framework for constructing (α, β)-approximation to any query space with a small VC-dimension is suggested in [19] where $\alpha = 1 + \varepsilon$ and $\beta = O(\log n)$.

Reducing (α, β)-approximation to an $\alpha = O(1)$ approximation. The size of the coreset usually depends on α and β. However, if they are reasonably small (e.g. polynomial in $\log n$), we can reduce the approximation factor and number of centers in few phases as follows: (i) Compute an (α, β)-approximation for small (but maybe not constant) α and β. (ii) Compute an ε-coreset for $\varepsilon = 1/2$ using this approximation. (iii) Compute an $O(1)$ factor approximation on the coreset. Since the coreset is small, such an approximation algorithm can run inefficiently, say, in polynomial time if the coreset is of size $(\log n)^{O(1)}$. The resulting $O(1)$ approximation for the coreset is also an $O(1)$ approximation for the original set, by the definition of the $\varepsilon = 1/2$ coreset. (iv) Recompute the coreset for the complete (original) data using the $O(1)$ approximation instead of the (α, β)-approximation to obtain a coreset of size independent of both n and d.

Algorithm 2: $\textsc{Coreset}(P, w, \mathcal{B}, m)$

```
Input:   A weighted set $(P, w)$ where $P \subseteq X$ and $(D, X)$ is a $\rho$-metric space,
         $(\alpha, \beta)$-approximation $\mathcal{B} : P \to B$,
         and sample size $m \geq 1$.
Output:  A pair $(S, u)$ that satisfies Theorem 2.5.6.
1 for each $b \in B$ do
2     Set $P_b \leftarrow \{p \in P \mid \mathcal{B}(p) = b\}$
3 for each $b \in B$ and $p \in P_b$ do
      Set $\text{Prob}(p) \leftarrow \dfrac{w(p)D(p, \mathcal{B}(p))}{2\sum_{q \in P} w(q)D(q, \mathcal{B}(q))} + \dfrac{w(p)}{2|B|\sum_{q \in P_b} w(q)}$.
4 Pick a sample $S$ of at least $m$ points from $P$ such that for each $q \in S$ and $p \in P$
  we have $q = p$ with probability $\text{Prob}(p)$.
5 for each $p \in P$ do
6     Set $u(p) \leftarrow \frac{w(p)}{|S| \cdot \text{Prob}(p)}$.
7 Set $u(p) \leftarrow 0$ for each $p \in P \setminus S$. /* Used only in the analysis. */
9 return $(S, u)$
```

Assumption 2.5.4. *In what follows we assume that:*

- *P is a set that is contained in X where (X, D) is a ρ-metric space and (ϕ, ε)-metric space as defined in* (2.5.1)*.*
- *(P, w, Q, g) is a query space as defined in* (2.23)*.*
- *dk denotes the dimension of f' as defined in Corollary 2.4.6.*

- *c is a sufficiently large constant that can be determined from the proofs of the theorems.*
- *$\mathcal{B}$ is an (α, β)-approximation for (P, w, Q, g) as in Definition 2.5.3.*
- *We are given an error parameter $\varepsilon \in (0, 1)$.*
- *We are given a maximum probability of failure $\delta \in (0, 1)$.*

We begin with the following claim, that is a simplified and generalized version of a similar claim in [30].

Lemma 2.5.5. *For every $b \in B$, and $p \in P_b$ have*

$$\frac{w(p)D(p,C)}{\sum_{q\in P} w(q)D(q,C)} \le \frac{\rho\alpha w(p)D(p,p')}{\sum_{q\in P} w(q)D(q,q')} + \frac{\rho^2(\alpha+1)}{\sum_{q\in P_b} w(q)}.$$

Proof. Put $p \in P$ and $b \in B$ such that $p \in P_b$. We need to bound

$$\begin{aligned}\frac{w(p)D(p,C)}{\sum_{q\in P} w(q)D(q,C)} &\le \frac{\rho w(p)D(p,p')}{\sum_{q\in P} w(q)D(q,C)} + \frac{\rho w(p)D(p',C)}{\sum_{q\in P} w(q)D(q,C)} \\ &\le \frac{\alpha\rho w(p)D(p,p')}{\sum_{q\in P} w(q)D(q,q')} + \frac{\rho w(p)D(p',C)}{\sum_{q\in P} w(q)D(q,C)},\end{aligned} \tag{2.24}$$

where the first inequality holds by (2.21), and the second inequality holds since $\mathcal{B}$ is an (α, β)-approximation.

To bound the last term, we sum the inequality $D(p', C) \le \rho(D(p', q) + D(q, C))$ over every $q \in P_b$ to obtain

$$\begin{aligned}D(p',C)\sum_{q\in P_b} w(q) &= \sum_{q\in P_b} w(q)D(p',C) \le \sum_{q\in P_b} w(q)\cdot\rho(D(p',q)+D(q,C)) \\ &= \rho\sum_{q\in P_b} w(q)D(q',q) + \rho\sum_{q\in P_b} w(q)D(q,C) \\ &\le \rho\alpha\sum_{q\in P_b} w(q)D(q,C) + \rho\sum_{q\in P_b} w(q)D(q,C) \\ &\le \rho(\alpha+1)\sum_{q\in P} w(q)D(q,C).\end{aligned}$$

Dividing by $\sum_{q\in P_b} w(q) \cdot \sum_{q\in P} w(q)D(q,C)$ yields

$$\frac{D(p',C)}{\sum_{q\in P} w(q)D(q,C)} \leq \frac{\rho(\alpha+1)}{\sum_{q\in P_b} w(q)}.$$

Substituting this in (2.24) yields the desired result

$$\frac{w(p)D(p,C)}{\sum_{q\in P} w(q)D(q,C)} \leq \frac{\rho\alpha w(p)D(p,p')}{\sum_{q\in P} w(q)D(q,q')} + \frac{\rho^2(\alpha+1)}{\sum_{q\in P_b} w(q)}.$$

□

Our first theorem suggests a coreset for k-means of size near-quadratic in k and quadratic in ε, based on our improved framework and last lemma. Existing work [30] for obtaining such coresets with only positive weights requires size cubic in k.

Theorem 2.5.6. *Under Assumption 2.5.4, let $t = k \cdot \rho^2(\alpha+1)\beta$. Let (S,u) be the output of a call to algorithm* Coreset$(P, w, \mathcal{B}, m)$*, where*

$$m \geq \frac{ct}{\varepsilon^2}\left(dk \log t + \log\left(\frac{1}{\delta}\right)\right).$$

Then, with probability at least $1-\delta$, (S,u) is an ε-coreset of size m for (P,w,Q,g).

Proof. Let $f : P \to [0,\infty)$ such that for every $C \in Q(P)$,

$$f(p,C) = \frac{D(p,C)}{\sum_{q\in P} w(q)D(q,C)},$$

and define

$$s(p) = \frac{\rho\alpha w(p)D(p,p')}{\sum_{q\in P} w(q)D(q,q')} + \frac{\rho^2(\alpha+1)}{\sum_{q\in P_b} w(q)}.$$

By Lemma 2.5.5,

$$\begin{aligned}\sum_{p\in P} \max_{C\in Q(P)} w(p)|f(p,C)| &= \sum_{b\in B}\sum_{p\in P_b} \max_{C\in Q(P)} w(p)f(p,C) \leq \sum_{b\in B}\sum_{p\in P_b} s(p)\\ &= \rho\alpha + |B|\cdot\rho^2(\alpha+1) = \rho(\alpha + |B|\alpha + |B|) \in O(\rho^2(\alpha+1)\beta k).\end{aligned}$$

Applying Theorem 2.4.6 with the query space (P, w, Q, g) then yields the desired bound. $\square$

Our second theorem in this section suggests a coreset for k-means of size near-linear in k by combining new observations with our improved framework.

Theorem 2.5.7. *Under Assumption 2.5.4, let (S, u) be the output of a call to algorithm* $\textsc{Coreset}(P, w, \mathcal{B}, m)$*, where*

$$m \geq \frac{c(\phi^2\alpha^2)}{\varepsilon^2}\left(dk + \log\left(\frac{1}{\delta}\right)\right) + \frac{c(1+\psi^2+\alpha^2\phi^2)}{\varepsilon^2}\left(d + \log\left(\frac{|B|}{\delta}\right)\right)$$

Then, with probability at least $1-\delta$, (S, u) is an ε-coreset of size m for (P, w, Q, g).

Proof. We use the notaion and varialbles of algorithm $\textsc{Coreset}(P, w, \mathcal{B}, m)$. For $b \in B$ and $p \in P_b$ let $p' = b$ and $v(p) = w(p) - u(p)$. To prove that S is an ε-coreset we need to prove that

$$|\sum_{p\in P} w(p)D(p, C) - \sum_{p\in S} u(p)D(p, C)| \leq \varepsilon \sum_{p\in P} w(p)D(p, C).$$

Using $v(p)$ and dividing by $\sum_{p\in P} u(p)D(p, C)$, it suffices thus to prove that

$$\frac{|\sum_{p\in P} v(p)D(p, C)|}{\sum_{p\in P} w(p)D(p, C)} \leq \varepsilon. \tag{2.25}$$

Without loss of generality, we assume that $\sum_{p\in P} w(p) = 1$, otherwise we divide each weight by this sum.

Let $H \subseteq P$ and $C \in Q$. We first observe that

$$\begin{aligned}
&\left|\sum_{p\in P} v(p)D(p,C)\right| \\
&= \left|\sum_{p\in H} v(p)(D(p,C)-D(p',C)) + \sum_{p\in P} v(p)D(p',C) + \sum_{p\in P\setminus H} v(p)(D(p,C)-D(p',C))\right| \\
&\le \left|\sum_{p\in H} v(p)(D(p,C)-D(p',C))\right| && (2.26)\\
&\quad + \left|\sum_{p\in P} v(p)D(p',C)\right| && (2.27)\\
&\quad + \left|\sum_{p\in P\setminus H} v(p)(D(p,C)-D(p',C))\right|. && (2.28)
\end{aligned}$$

To bound (2.25), we will prove that, with probability at least $1 - c\delta$, each of the expressions (2.26), (2.27) and (2.28) is bounded as desired, after defining

$$H = \left\{ p \in P \mid \frac{w(p)\cdot|D(p,C)-D(p',C)|}{2\sum_{q\in P} w(q)D(q,C)} \le 2\phi\alpha \text{Prob}(p) \right\}.$$

Bound on (2.26): Let

$$h(p,C) = \begin{cases} \frac{(D(p,C)-D(p',C))}{2\sum_{q\in P} w(q)D(q,C)} & p \in H \\ 0 & p \notin H, \end{cases}$$

$b = 2\phi\alpha$, and $a = -b$. We have for every $p \in P$,,

$$w(p)\cdot\frac{h(p,q)-a}{b-a} + w(p) = \frac{w(p)h(p,q)}{2b} + \frac{3w(p)}{2} \le \text{Prob}(p) + \frac{3w(p)}{2}$$

Substituting $f = h$, $s(p) = \text{Prob}(p) + 3w(p)/2$, $t = 1 + 3/2$, and $u = u'$ in Theorem 2.4.3 yields that with probability at least $1 - \delta$ we have

$$\forall C \in Q : |\bar{h}(P,w,C) - \bar{h}(S,u,C)| \le \varepsilon.$$

Substituting the definition of h in this inequality yields the desired bound on (2.26),

$$\frac{\left|\sum_{p\in H} v(p)\cdot(D(p,C)-D(p',C))\right|}{\sum_{q\in P} w(q)D(q,C)} = \left|\sum_{p\in H} v(p)h(p,C)\right| = |\bar{h}(P,w,q)-\bar{h}(S,u,q)| \le \varepsilon.$$

Bound on (2.27) **:**

$$\begin{aligned}\frac{\left|\sum_{p\in P} v(p)D(p',C)\right|}{\sum_{p\in P} w(p)D(p,C)} &= \frac{\left|\sum_{b\in B}\sum_{p\in P_b} v(p)D(p',C)\right|}{\sum_{p\in P} w(p)D(p,C)} \le \sum_{b\in B}\frac{\left|\sum_{p\in P_b} v(p)D(p',C)\right|}{\sum_{p\in P} w(p)D(p,C)}\\ &= \sum_{b\in B}\frac{\sum_{p\in P_b} w(p)D(p',C)}{\sum_{p\in P} w(p)D(p,C)}\cdot\frac{\left|\sum_{p\in P_b} v(p)D(p',C)\right|}{\sum_{p\in P_b} w(p)D(p',C)}\\ &= \sum_{b\in B}\frac{\sum_{p\in P_b} w(p)D(p',C)}{\sum_{p\in P} w(p)D(p,C)}\cdot\frac{\left|\sum_{p\in P_b} v(p)\right|}{\sum_{p\in P_b} w(p)}.\end{aligned}$$

We first bound the rightmost term in the last inequality. Let $I(p,b) = 1/\sum_{q\in P_b} w(q)$ if $p\in P_b$ and $I(p,b)=0$ otherwise. We have for every $p\in P_b$

$$w(p)I(p,b) \le \frac{w(p)}{\sum_{q\in P_b} w(q)} \le 2|B|\text{Prob}(p).$$

Hence, $\sum_{p\in P} w(p)I(p,b) \le 2|B|$. Each point in S is sampled with probability proportional to $2|B|\text{Prob}(p)$, and

$$\begin{aligned}|S| = m &\ge \frac{ck(t+\beta)}{\varepsilon^2}\left(d\log t + \log(\beta k) + \log\left(\frac{1}{\delta}\right)\right)\\ &\in \Omega(1)\cdot\frac{2|B|(1+\psi+\alpha\phi)^2}{\varepsilon^2}\left(\log(2|B|)+\log\left(\frac{|B|}{\delta}\right)\right).\end{aligned}$$

Using this, we substitute the query space $(P,w,\{b\},I)$ whose dimension is $d=1$, $s(p)=2|B|\text{Prob}(p)$, $t=2|B|$, $b=1$, $a=0$ and replace δ by $\delta/|B|$, and ε with $\varepsilon' = \varepsilon/(\rho(1+\alpha))$ in Theorem 2.4.3 to obtain that, for a sufficiently large constant c, with probability at least $1-\delta/|B|$,

$$|\bar{I}(P,w,\{b\}) - \bar{I}(S,u,\{b\})| \le \varepsilon'.$$

Hence,

$$\left|\frac{\sum_{p\in P_b} v(p)}{\sum_{q\in P_b} w(q)}\right| = |\bar{I}(P,w,b) - \bar{I}(S,u,b)| \le \varepsilon' = \frac{\varepsilon}{\rho(1+\alpha)}. \tag{2.29}$$

By the union bound, with probability at least $1-\delta$, the last inequality holds for every $b \in B$ simultaneously. In what follows we assume that this event indeed occurs.

Substituting $x = p'$ and $y = p$ in (2.30) yields

$$D(p',C) \le \rho(D(p,C) + D(p,p')). \tag{2.30}$$

Hence,

$$\begin{aligned}
\sum_{b\in B}\sum_{q\in P_b} w(q)D(b,C) &= \sum_{p\in P} w(p)D(p',C) \\
&\le \rho\sum_{p\in P} w(p)D(p,C) + \rho\sum_{p\in P} w(p)D(p,p') && (2.31)\\
&\le \rho\sum_{p\in P} w(p)D(p,C) + \alpha\rho\sum_{p\in P} w(p)D(p,C) && (2.32)\\
&= \rho(1+\alpha)\sum_{p\in P} w(p)D(p,C),
\end{aligned}$$

where (2.31) is by (2.30), and (2.32) is since B corresponds to an (α,β)-approximation. This bounds (2.27) as

$$\sum_{b\in B}\frac{\sum_{q\in P_b} w(q)D(b,C)}{\sum_{r\in P} w(r)D(r,C)} \le \frac{\rho(1+\alpha)\sum_{p\in P} w(p)D(p,C)}{\sum_{r\in P} w(r)D(r,C)} = \rho(1+\alpha).$$

Bound on (2.28)**:** For $p \in P_b \setminus H$ we have

$$|D(p,C) - D(p',C)| \le \phi D(p,b) + \psi D(b,C) \le (1-\psi/\varepsilon)|D(p,C) - D(p',C)| + \psi D(b,C),$$

where the first inequality is by (2.30), and the last one is since $p \notin H$. Subtracting $(1-\psi/\varepsilon)|D(p,C)-D(p',C)|$ from the last inequality yields

$$|D(p,C)-D(p',C)| \leq \varepsilon D(p',C) = \varepsilon D(b,C).$$

Hence

$$\begin{aligned}\frac{\left|\sum_{p\in P\setminus H} v(p)\cdot (D(p,C)-D(b,C))\right|}{\sum_{q\in P} w(q)D(b,C)} &\leq \frac{\sum_{p\in P\setminus H}(w(p)+u(p))\cdot |D(p,C)-D(b,C)|}{\sum_{q\in P_b} w(q)D(b,C)}\\ &\leq c\psi \leq \varepsilon'\end{aligned} \tag{2.33}$$

We also have,

$$\frac{\sum_{p\in P} u(p)}{\sum_{q\in P_b} w(q)} = 1+\left(\frac{\sum_{p\in P} u(p)}{\sum_{q\in P_b} w(q)}-1\right) \leq 1+\left|\frac{\sum_{p\in P} u(p)}{\sum_{q\in P} w(q)}-1\right| = 1+\left|\frac{\sum_{p\in P} v(p)}{\sum_{q\in P} w(q)}\right| \leq 1+\varepsilon',$$

where the first inequality is the triangle inequality and the last one is by (2.29).

Combining the last inequality and (2.33) bounds (2.28) by

$$\frac{\left|\sum_{p\in P_b} v(p)\cdot (D(p,C)-D(b,C))\right|}{\sum_{q\in P_b} w(q)D(b,C)} \leq 2\psi(2+\varepsilon') \leq 6\varepsilon' = \frac{6\varepsilon}{2+\alpha\phi}. \tag{2.34}$$

Bounding (2.25)**:** By plugging the bounds (2.26), (2.27) and (2.28) above in (19) we bound (2.25) with probability at least $1-O(\delta)$, as

$$\begin{aligned}&\frac{\left|\sum_{p\in P} v(p)D(p,C)\right|}{\sum_{q\in P} w(q)D(q,C)}\\ &\leq \varepsilon + (1+\varepsilon+\alpha\phi)\big(\varepsilon+2\varepsilon(2+\varepsilon)\big) \in O(\varepsilon(1+\alpha\phi)).\end{aligned}$$

Replacing ε with ε/c, and δ with δ/c in the above proof then proves the theorem. □

The values of ρ and ϕ. Algorithm 2 can be applied to compute a coreset for any given variant of the k-means/median problem given a set P and a ρ or (ρ, ε)-metric (X, D). The only difference is the size of the required coreset. The parameters ρ and ϕ can usually be computed easily using Lemma 2.5.2. For example, in the case of distances to the power of $r \geq 1$, the value of ρ is roughly 2^r and the value of ϕ is roughly ε^r. For most common m-estimators the values are similar, when r is some constant.

The values of α and β can be $\alpha = \beta = O(1)$ for k-means and all its variants, by using the generic algorithm for computing (α, β)-approximation in [19] with $\alpha = \beta = O(\log n)$, and then use the technique for reducing α and β. For bounding the approximation of the bi-criteria approximation in [19] only the pseudo-dimension of the problem is required, which is usually easy to compute as explained below.

Dimension d. Unlike the total sensitivity t, the dimension d for numerous problems was already computed in many papers in computational geometry and machine learning (in the context of PAC learning). This includes reductions and connections to similar notions such as the shattering dimension or the VC-dimension of a set. General techniques for computing the dimension of a set based on number of parameters to define a query, or number of operations that are needed to answer a query can be found in the book [4].

Dimension of the k-means problem and its variants. Note that, unlike sensitivity, the dimension is less dependent on the exact type of distance function. For example, using Euclidean distance or Euclidean distance to the power of 3 as a variant for the k-means clustering problem does not change the dimension. This is because set of ranges for both of these problems is the same: subsets of the input points that can be covered by k balls. It is easy to compute the dimension for the k-means problem (the query space (P, w, Q, g), as well as the modified query space for the function f. These bounds can be found in [19]. In addition, [19] provide a simple reduction that shows that the dimension for k centers is the same as the dimension of 1 center multiplied by k.

In short, for the Euclidean space, the dimension of the k-means/median problem is $O(dk)$, and for metric spaces (graph) the dimension is $O(k \log n)$.

Smaller coreset for k-means queries. Consider the k-means queries in $\mathbb{R}^d$, i.e., the cost is the sum of squared distances $\sum_{p \in P} D(p, C)$ over every point in P to its nearest center in a given set C of k points in $\mathbb{R}^d$. It was proven that projecting P onto an $O(k/\varepsilon)$-dimensional subspace that minimizes its sum of squared distances, known as the low-rank approximation of P, would preserve this sum, to any set of k centers, up to a factor of $1 \pm \varepsilon$. This is in some sense an ε-coreset for P of size n and that is not subset of the input, but of low-dimensionality. In particular, this result implies that there is a set of centers (known as centroid set [26]) that is contained in a $O(k/\varepsilon)$-dimensional space, such that every set of k centers in $\mathbb{R}^d$ can be replaced by a set of k centers in the centroid set, that would yield the same cost up to a factor of $1 \pm \varepsilon$. In particular, this implies that the dimension of the k-means problem can be reduced to $O(k/\varepsilon)$ instead of dk, i.e., independent of d. Combining this result with ours yields the first coreset for k-means of size independent of d that is subset of the input (in particular, preserve the sparsity of the input points) that also supports streaming.

Weak coresets of size independent of d. For the non-Euclidean case or non-squared distances it seems that it is impossible to obtain coreset of size independent of d. However, coreset as defined above (sometimes called strong coreset) approximates *every* query in the set of queries, while the main application and motivation for constructing coreset is to compute the *optimal* query or its approximation. A weak coreset is a small set that can be used to give such an approximation. The exact definition of weak coreset also changes from paper to paper. In particular, a weak coreset for k-means was suggested in [17]. However, to extract the approximated solution from the coreset we must run an exhaustive search and cannot use existing algorithms or heuristics as in the case of strong coreset.

In this thesis, following [19], we use a simple and general definition of weak coreset, that is also more practical. Instead of defining a unique (static, global) set Q of queries, we define Q to be a function that maps every subset (potential coreset) S of P to a set of queries. A weak coreset S needs only to approximate the queries in $Q(S)$. It turns out that for many case the (generalized definition) of dimension for such a query space is much smaller compared to the traditional case where $Q(S) = Q(P)$ is the same for every subset $S \subseteq P$.

To be able to use this property in our existing proofs, we require a monotonicity property: that the set $Q(T)$ of queries that are assigned to a subset $T \subset S$ must be contained in $Q(S)$. If we can prove that for every $S \subseteq P$, the set $Q(S)$ contains a $(1+\varepsilon)$-approximation to both the optimal solution of S and P, then we can extract such an approximation from S. For k-means, k-median and their variants, it was proven in [37] that the optimal k-centers of a set S can be approximated by such a set that is spanned by $O(k/\varepsilon)$ points in S. By defining $Q(S)$ to be the union of the optimal center of P, with all the centers that are spanned by $O(k/\varepsilon)$ points in S, we get that S is a weak coreset. Note that the definition of Q can be explicit (without knowing the optimal center of P) and is needed only to bound its dimension as in Definition 2.3.5.

Extracting a $(1+\varepsilon)$-approximation from the coreset. It was proven in [19] that for problems such as k-median such weak coresets have dimension $O(k/\varepsilon)$, i.e., independent of d. Unlike [19], we suggest here a very simple way to extract the approximated solution from the coreset: compute the weighted coreset (S, u) (of size independent of d) as defined in Algorithm 2, and then use any given algorithm to compute a $(1+\varepsilon)$ approximation set C of k centers on the coreset (or any other trusted heuristic that we hope computes such a set C). Since C is not necessarily spanned by few points in S, it may not be a good approximation for the original set P and we should not return it. However,the proof in [37] is constructive and shows a near-linear time algorithm that using such an approximated solution set C, we can compute another $(1+O(\varepsilon))$-approximation C' that has the additional property that C' is spanned by $O(k/\varepsilon)$ points in S. Hence, C' is both a near-optimal solution to S and in $Q(S)$, so it must be a $(1+\varepsilon)$-approximation for the optimal solution of P.

2.6 Appendix A: Merge and Reduce Tree

We now briefly introduce the previous technique for maintaining coresets in the streaming setting due to Har-Peled and Mazumdar [27] and Bentley and Sax [7]. In this method, a merge-and-reduce tree is built by using an offline coreset construction as a blackbox. Previously merge-and-reduce was the only known technique for building a streaming coreset for metric k-median, and it relies solely on the following two properties:

1. Merge: The union of (k,ϵ)-coresets is a (k,ϵ)-coreset.

2. Reduce: A (k,ϵ)-coreset of a (k,δ)-coreset is a $(k,\epsilon+\delta)$-coreset.

The merge-and-reduce tree works as follows. There are buckets B_i for $i \geq 0$. In each step, the bucket B_0 takes in a segment of $O(1)$ points from the stream. Then the tree works like counting in binary: whenever buckets B_0 to B_{i-1} are full, these i buckets are merged and then reduced by taking a $(k, \frac{\epsilon}{\log n})$-coreset and storing the result in B_i.

Let s be the space of offline construction, which depends on ϵ as ϵ^{-a}. At the end of the stream, $O(\log n)$ buckets have been used and each bucket uses $O(s \log^a n)$ space; this incurs a multiplicative overhead of $\Theta(\log^{a+1} n)$ in the storage requirement. The second factor comes from using the accuracy parameter $\frac{\epsilon}{\log n}$, which is necessary by Property 2 since the construction will be compounded $O(\log n)$ times. Due to this compounding, the runtime is multiplied by a factor of $O(\log n)$.

2.7 Appendix B: General Streaming Reduction

We present a general technique for converting an offline coreset construction to a streaming coreset construction with $O(\log n)$ overhead. Given a ρ-metric space (X, D) (recall

Definition 2.5.1), we build a query space (P, w, g, Q) in the same way as in Definition 2.5.3: $Q(P) = \{C \subseteq X \mid |C| \leq k\}$ and $g(p, C) = D(p, C) := \min_{c \in C} D(p, c)$. Here k is a positive integer that denotes how many centers may be used for the clustering.

Our bicriterion algorithm is an adjustment to the algorithms of [9] and [CCF] with the following important difference: our bicriterion is online, so we do not delete and reassign centers as is [9]. This "online" property is critical for the algorithm to work and is one of the main technical ideas. Although a fully online bicriterion can require linear space, we maintain a division of the stream P into a prefix R and a suffix $P \setminus R$ such that our bicriterion is online on the suffix $P \setminus R$ and the prefix R can be largely ignored. To maintain this property of being online for the suffix, we incur only a modest space increase from $O(k \log n)$ to $O(\log(\frac{1}{\epsilon}) k \log n)$.

After having an online bicriterion (it is further explained below why this property is essential), the offline coreset algorithms perform non-uniform sampling procedure with carefully chosen probabilities that are defined by the bicriterion. Equipped with our new bicriterion algorithm, implementing the sampling procedure is rather straightforward computation which is explained in Section 2.7.2. As a result, we can implement any sampling-based coreset algorithm for k-median without merge-and-reduce and in one pass. As such it is applicable to several coreset constructions (such as k-means and other M-estimators). In addition, we believe that our methods will work with other objective functions as well such as (k, j)-subspace, and we hope that future work will investigate these directions.

Many clustering algorithms [10, 24, 9] maintain a weighted set (B, u) of points (which are selected using a facility-location algorithm). Upon arrival of an update $(p, w(p))$ from the stream, this update is added to the set by $u(p) \leftarrow u(p) + w(p)$.

In these algorithms, only a single operation is performed on (B, u) which we call MOVE. For two points $p, p' \in B$ with weights $u(p)$ and $u(p')$, the function MOVE(p, p') does the following: $u(p') \leftarrow u(p') + u(p)$ and $u(p) \leftarrow 0$. This essentially moves the weight at location p to the location of p'. The motivation for building (B, u) will be compression;

(B, u) will be maintained over the stream P in such a way that $|B| = O(\log |P|)$.

Throughout this section, we will assume for ease of exposition that each point in the stream is from a distinct location. This simplifies the analysis, allowing $\mathcal{B}$ to take only a single value for each input point. The algorithm works for general inputs without modification, requiring only a more careful notation for the analysis. Additionally, we state the algorithm for unweighted input (where $w(p) = 1$ for all $p \in P$) and the parameter n is defined as $|P|$. We still include $w(p)$ throughout the algorithm and analysis, as the analysis generalizes to weighted inputs where the parameter n is replaced by the sum of all weights (after normalizing the minimally weighted point to 1).

2.7.1 An Algorithm for building a coreset

First, let us describe how the algorithm of [9] works. We will modify this algorithm as part of our coreset construction. In this summary, we alter the presentation from that of [9] to more fluidly transition to our modified version but the algorithm remains the same. [9] operates in phases $i \geq 1$. This means that the algorithm maintains a phase number i (used internally by the algorithm), beginning in phase $i = 1$. As the stream arrives, the algorithm may decide to increment the phase number. Let (R_i, w_i) denote the prefix of the input received before the end of phase i, and let $\mathsf{OPT}_k(R_i)$ denote the minimal value of $\bar{g}(R_i, w_i, C)$ over every $C \in Q$. When phase $i + 1$ begins, a value L_{i+1} is declared on Line 27 as a lower-bound for the current value of $\mathsf{OPT}_k(R_{i+1})$. The algorithm has computed (M_i, u_i), which we inductively assume is a bicriterion approximation for R_i (more precisely, a map $\mathcal{B} : R_i \to M_i$ such that $\cup_{p \in R_i}(\mathcal{B}(p), w(p)) = (M_i, u_i)$. However, to maintain polylogarithmic-space the algorithm pushes (M_i, u_i) to the beginning of the stream and restarts the bicriterion construction. This means that the algorithm, at this point, restarts by viewing the stream as $(P, w - w_i + u_i)$ (i.e. replacing (R_i, w_i) with (M_i, u_i)). Continuing in this way, the algorithm maintains a bicriterion (M_{i+1}, u_{i+1}) for $(R_{i+1}, w_{i+1} - w_i + u_i)$ (which is also a bicriterion for (R_{i+1}, w_{i+1}) by Theorem 2.7.2) until the next phase change is triggered.

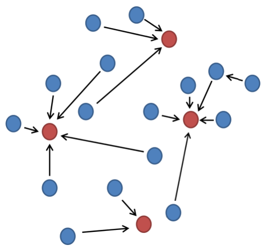


Figure 2.3: A bicriterion approximation. Although connections are not optimal, the sum of all connections costs is $O(\mathsf{OPT})$.

Now we explain our modifications to [9] (see Algorithm 3). The first step is that the bicriterion our algorithm builds must be determined "online" in the following sense: upon receiving a point $(x, w(x))$, the value of $\mathcal{B}(x)$ must be determined (and never be altered) before receiving the next point from the stream.

This generalization is necessary for the following reason. Suppose we connect an element p to a center b_1. Later in the stream, we open a new center b_2 that becomes the closest center to p. However, using polylogarithmic space, we have already deleted b_1 and/or p from memory and the state of our algorithm is identical to the case where b_1 remains the closest center to p. Therefore the connections must be immutable, and this results in non-optimal connections.

Definition 2.7.1. *An online $[\alpha, \beta]$-bicriterion is an algorithm that maintains a bicriterion (B, t) over a stream X, and operates online in the following sense. Upon arrival of each point p, it is immediately decided whether p is added to B, and then $t(p)$ is determined. Both of these decisions are permanent.*

Upon receiving an update $(p, w(p))$ from the stream, the algorithm may call $\mathsf{MOVE}(p, p')$ for the nearest $p' \in B$ to p. In the analysis we use the function $\mathcal{B} : P \to B$ that maps each point p to its immediate location after this initial move (either p itself, or p'). If future moves are performed on p, this does not change the value $\mathcal{B}(p)$. $\mathcal{B}$ is not stored by the algorithm due to space constraints; only the value of $\mathcal{B}(p)$ for the most recent point p is used by the algorithm, and older values are used in the analysis only. We will show that $\mathcal{B}$ is a $(O(1), O(\log n))$-approximation that allows us to maintain a coreset over the stream.

We now state Algorithm 3. ϕ and γ are constants (dependent on ρ) used in the analysis that are defined as in [9]. Each point has a flag that is either raised or lowered (given by the *Flag* function). All points have their flag initially lowered, given on Line 1. A lowered flag shows that the point is being read for the first time (being received from the stream), and a raised flag shows that the point is being re-read by the algorithm (having been stored in memory).

On Line 22, (M_i, u_i) is the weighted set (B, u) as it exists at the end of phase i. We define the cost of $\mathsf{MOVE}(p, p')$ to be $w(p)D(p, p')$. The value K_i is therefore the total cost of all moves performed in phase i.

At a phase change, the set (B, u) is pushed onto the beginning of the stream with all points having a raised flag. This means that during the next $|B|$ iterations of the outer-loop where a point $(x, w(x))$ is received we actually receive a point from memory. We continue to process the stream after these $|B|$ points are read.

The following theorem summarizes the guarantees of this algorithm. We note that, as stated, the algorithm's runtime of $O(nk \log n)$ is not correct - in fact the number of phases may be arbitrarily large. However, using the same technique as detailed in Section 3.3 of [9] the number of phases can be bounded to $O(\frac{n}{k \log n})$ while requiring $O(k^2 \log^2 n)$ time per phase.

Theorem 2.7.2 ([9])**.** *Let Algorithm 3 process a stream (P, w) of at most n points. Let R_i denote the prefix of the stream received before the end of phase i, and let K_i and L_i be their final values from the algorithm (which are never modified after phase i). With probability at least $1 - \frac{1}{n}$, the following statements all hold after processing each point:*

1. $\sum_i K_i \leq \frac{\rho\phi\gamma}{\phi - \rho} \mathsf{OPT}_k(R_i)$

2. *The total runtime is $O(nk \log n)$*

3. *For every phase i, $L_i \leq \mathsf{OPT}_k(R_i) \leq \phi L_i \leq L_{i+1}$*

Algorithm 3: Input: integer k, ρ-metric space (X, D), stream (P, w) of n weighted points from (X, D). Output: $\mathcal{B}(x)$ after receiving each point x, and a weighted set (M_i, u_i) after each phase $i \geq 1$

```
1  $Flag(x) = 0$ for all $x \in P$
2  $L_1 \leftarrow$ minimum $D(x, y)$ for any $x, y$ in the first $k$ distinct points
3  $i \leftarrow 1$
4  $K_1 \leftarrow 0$
5  $B \leftarrow \emptyset$
6  $u(x) \leftarrow 0$ for all $x$
7  for each point $(x, w(x))$ received do
8      $u(x) \leftarrow u(x) + w(x)$
9      $y \leftarrow argmin_{y \in Q} D(x, y)$ (break ties arbitrarily)
10     $I \leftarrow 1$ with probability $\min\{\frac{w(x)D(x,y)}{L_i/k(1+\log_2 n)}, 1\}$, otherwise $I \leftarrow 0$
11     if $I$ then
12         if $Flag(x) = 0$ then
13             $\mathcal{B}(x) \leftarrow x$
14             $B \leftarrow B \cup \{x\}$
15     else
16         $K_i \leftarrow K_i + w(x)D(x, y)$
17         $u(y) \leftarrow u(y) + u(x)$ /* Step 1 of MOVE$(x, y)$ */
18         $u(x) \leftarrow 0$ /* Step 2 of MOVE$(x, y)$ */
19         if $Flag(x) = 0$ then
20             $\mathcal{B}(x) \leftarrow y$
21     if $K_i > \gamma L_i$ or $|B| > (\gamma - 1)(1 + \log_2 n)k$ then
22         $(M_i, u_i) \leftarrow (B, u)$
23         $Flag(b) \leftarrow 1$ for all $b \in B$
24         Push $(B, u)$ onto the stream $(P, w)$ before the next point to read
25         $B \leftarrow \emptyset$
26         $u(x) \leftarrow 0$ for all $x$
27         $L_{i+1} \leftarrow \phi L_i$
28         $q \leftarrow 0$
29         $K_{i+1} \leftarrow 0$
30         $i \leftarrow i + 1$
```

4. At the execution of Line 22, M_i consists of $O(k \log n)$ points

Part 1 of the preceding theorem, which bounds the total cost of all moves performed by the algorithm, also serves as an upper bound on the Earth-Mover distance between the stream (P, w) and the maintained set (B, u).

Definition 2.7.3 (Earth-Mover Distance)**.** *Let (A, w_A) and (B, w_B) be weighted sets in (X, D). Morever, let them be of equal weight in the sense that $\Sigma_{a \in A} w(a) = \Sigma_{b \in B} w(b)$. Define a "movement" from (A, w_A) to (B, w_B) to be a weighted set (Y, v) in $(X \times X, D)$ such that $\cup_{(a,b) \in Y}(a, v(a, b)) = (A, w_A)$ and $\cup_{(a,b) \in Y}(b, v(a, b)) = (B, w_B)$. Then $d_{EM}(A, w_A, B, w_B)$ is the minimum of $\Sigma_{(a,b) \in Y} v((a, b))D(a, b)$ over all movements Y from (A, w_A) to (B, w_B).*

Another way to view the preceding definition is in terms of probability distributions. Over all joint distributions over $A \times B$ with marginal distributions (A, w_A) and (B, w_B), we seek the minimum possible cost (as defined) for any such joint distribution.

From now on we will write a weighted set A instead of (A, w_A) when the meaning is clear. If we start with a set A_0 and apply n operations of MOVE until it becomes the set A_n, we can provide an upper-bound for $d_{EM}(A_0, A_n)$ by summing $d_{EM}(A_i, A_{i+1})$ for $0 \leq i < n$. This is a direct application of the triangle inequality. And if A_{i+1} is obtained from A_i by applying MOVE(p, p'), then $d_{EM}(A_i, A_{i+1}) = D(p, p')w(p)$, the cost of this move.

The Earth-Mover distance is important for clustering problems for the following reason. For any weighted sets (P, w) and (B, u) and query C, $|\bar{g}(P, w, C) - \bar{g}(B, u, C)| \leq d_{EM}(P, w, B, u)$. This is immediate from a repeated application of the triangle-inequality (proofs are found in Theorem 2.3 of [24] as well as in [10, 9, 23]).

Theorem 2.7.4. *There exists an algorithm stores $O(\log\left(\frac{1}{\epsilon}\right) k \log n)$ points and maintains for every prefix (R, w') of the stream (P, w): (1) a weighted set (M, u) such that $d_{EM}(M, u, R, w') \leq \epsilon \mathsf{OPT}_k(R)$, and (2) a $(O(1), O(\log(\frac{1}{\epsilon}) \log n))$-bicriterion $\mathcal{B}$ for (R, w').*

Morever, this bicriterion $\mathcal{B}$ is computed online in the sense that $\mathcal{B}(x)$ is determined upon receiving x from the stream.

Proof. The algorithm will consist of running Algorithm 3 and storing certain information for the λ most recent phases (where λ depends on ϵ).

Let i be the current phase number, and let (R, w') be the points received so far. We remind the reader that when the meaning is clear we suppress notation and write R for the weighted set (R, w'), and likewise for (M, u). Define $\lambda = 2 + \lceil \log_\phi(\frac{\rho\gamma}{(\phi-\rho)\epsilon}) \rceil$. The prefix R and the set M in the statement of the theorem will be $R_{i-\lambda}$ and $M_{i-\lambda}$. To upper bound $d_{EM}(R, M)$, which by definition is the minimum cost of any movement from R to M, we note that one movement is the set of moves carried out by the algorithm through phase $i - \lambda$, whose cost is $\sum_{j=1}^{i-\lambda} K_j$. Part 1 of Theorem 2.7.2 then shows that $d_{EM}(R, M) \leq \Sigma_{j=1}^{i-\lambda} K_j \leq \frac{\rho\phi\gamma}{\phi-\rho}\mathsf{OPT}_k(R_{i-\lambda})$. Combining Statements 3 and 4 of the theorem shows that $\mathsf{OPT}_k(R_{i-\lambda}) \leq \phi^{1-\lambda}\mathsf{OPT}_k(R)$. Therefore $d_{EM}(R, M) \leq \frac{\rho\phi\gamma}{\phi-\rho}\phi^{1-\lambda}\mathsf{OPT}_k(R) \leq \epsilon\mathsf{OPT}_k(R)$ as desired.

As for the second statement of the theorem, the algorithm defines the map $\mathcal{B}$ and we are currently interested in the restriction of $\mathcal{B}$ to $R \setminus R_{i-\lambda}$. $\mathcal{B}$ maps to at most $O(k \log n)$ points per phase - this is guaranteed by Statement 5 (a direct result the termination condition on Line 17). Over the last λ phases, this then maps to $O(\lambda k \log n) = O((\frac{1}{\epsilon})k \log n)$ points. Therefore we have that $\beta = O((\frac{1}{\epsilon}) \log n)$. And the value of α is immediate from Statement 1 of Theorem 2.7.2, since $\mathcal{B}$ incurs only a subset of the costs as K_j (the subset that comes from the new portion of the stream $R_j \setminus R_{j-1}$).

As a final note, we do not store $\mathcal{B}(P)$ in memory (it may be linear in space). For algorithms in the following sections it is only required to know it's value for the most recent point received; previous values are used only in the analysis. □

2.7.2 Maintaining a coreset over the stream

In the previous section, we presented an algorithm that maintains an (α, β)-approximation of the stream (this is given by the function $\mathcal{B} : P \to B$). In this section we show how we can use this approximation to carry out the coreset construction of Algorithm 2 on an insertion-only stream. In the offline construction, a sample S of m points is taken from X according to a distribution where point p sampled with probability depending on $D(p, \mathcal{B}(p))$, $|B|$, and $n_{\mathcal{B}(p)}$ (the total weight of points connected to the center $\mathcal{B}(p)$, which is written as $\Sigma_{q \in P_b} w(q)$ where $b = \mathcal{B}(p)$) - the specific formula for the probability written below (and comes from Line 3 of Algorithm 2). All three of these quantities can easily be maintained over the stream using Algorithm 4 since $\mathcal{B}$ is online (i.e. $\mathcal{B}(p)$ never changes).

$$Prob(p) = \frac{w(p)D(p, \mathcal{B}(p))}{2\sum_{q \in P} w(q)D(q, \mathcal{B}(q))} + \frac{w(p)}{2|B|\sum_{q \in P_b} w(q)}$$

Upon receiving a point p, we assign $r(p)$ a uniform random number in the interval $(0, 1)$. This is the threshold for keeping p in our sample S - we keep the point p if and only if $r(p) < Prob(p)$. For each point p, $Prob(p)$ is non-increasing as the stream progresses (this is immediate from the formula and from the fact that clusters never decrease in size since $\mathcal{B}$ is online). Therefore after receiving point p, we update $Prob(s)$ for each $s \in S$ and delete any such s that drop below their threshold: $r(s) \geq Prob(s)$. Once a point crosses the threshold, it may be deleted since $Prob(s)$ is non-increasing and so it will remain below the threshold at all future times. In this way, the construction exactly matches the output as if the offline Algorithm 2 had been used.

Algorithm 4: Input: integer k, ρ-metric space (X, D), stream P of points n from (X, D), online bicriterion $\mathcal{B}$, sample size $m \geq 1$. Output: a coreset S

```
1  $S \leftarrow \emptyset$
2  for each point $p$ that arrives do
3      Assign $r(p)$ a uniform random in $(0, 1)$
4      $S \leftarrow S \cup \{p\}$
5      Compute Prob($p$) according to Line 3 of Algorithm 2
6      for each point $s \in S$ do
7          Update Prob($s$)
8          if $Prob(s) \leq r(s)$ then
9              Delete $s$ from $S$
10         Update $u(s)$ according to Line 6 of Algorithm 2
```

Algorithm 3 provides the function $\mathcal{B}$ and a weighted set $M_{i-\lambda}$. Beginning in phase $i-\lambda$, we will begin running Algorithm 4. This outputs the sample S for phases $i-\lambda+1$ until the current phase i. The following theorem shows that $M_{i-\lambda} \cup S_i$ is a coreset for the stream. Of course, we need to do this construction for each of the $\lambda = O(\log(\frac{1}{\epsilon}))$ most recent phases, so the space gets multiplied by this factor.

We return to definition 2.5.2 of an r Log-Log Lipschitz function $\tilde{D}$. Given a metric space $(X, dist)$, the space (X, D) where $D = \tilde{D}(dist)$ is a ρ-metric space for $\rho = \max\{2^{r-1}, 1\}$. It is well-known that most M-estimators can be recast as a Log-Log Lipschitz function for a low constant value of r. For example, k-means has $r = 2$.

Theorem 2.7.5. *There exists a single-pass streaming algorithm requiring $O(\epsilon^{-O(1)} k \log n(\log n + \log k + \log \frac{1}{\delta}))$ space that maintains a (k, ϵ)-coreset for the k-median clustering of an r-Log-Log-Lipschitz function $\tilde{D}$ on a stream of at most n elements with probability at least $1-\delta$.*

Proof. Running Algorithm 2 (producing the M_i) and Algorithm 3 (producing the S_i) in parallel, we will show that $M_{i-\lambda} \cup S_i$ is the desired coreset of the stream.

We already have by Theorem 2.7.4 that $d_{EM}(M_{i-\lambda}, R_{i-\lambda}) \leq \epsilon \mathsf{OPT}_k(P) \leq \epsilon \bar{g}(P, w, C)$ for any set C of size k. Also, by Theorem 2.5.6 we have the S_i is an (k, ϵ)-coreset for $P \setminus R_{i-\lambda}$. This is because (in the statement of Theorem 2.5.6) we are required to carry out the construction of Algorithm 2 using $m \geq \frac{ck(t+\beta)}{\varepsilon^2}\left(d \log t + \log(\beta k) + \log\left(\frac{1}{\delta}\right)\right)$ where $t = \alpha/\phi$. It was shown in Lemma 2.5.2 that $\phi = (\epsilon/r)^r$. We are using the $(O(1), O(\log(\frac{1}{\epsilon}) \log n))$-approximation $\mathcal{B}$, and in [19] it is shown that $d = O(\log n)$ for k-median in a ρ-metric space. So therefore the minimal possible value of m that satisfies the hypotheses of the theorem is $O(\frac{k(\epsilon^{-r} + \log n)}{\varepsilon^2}\left(\log n \log \epsilon^{-r} + \log(k \log n) + \log\left(\frac{1}{\delta}\right)\right))$. Simplifying notation, this is $O(\epsilon^{-O(1)} k \log n(\log n + \log k + \log \frac{1}{\delta}))$.

We write $\mathsf{COST}(A, C)$ to denote $\bar{g}(A, w_A, C)$ to simplify notation; by $A \cup B$ we mean $(A \cup B, w_A + w_B)$. Taking the union, we get that $|\mathsf{COST}(P, C) - \mathsf{COST}(M_{i-\lambda} \cup S_i, C)| \leq |\mathsf{COST}(R_{i-\lambda}, C) - \mathsf{COST}(M_{i-\lambda}, C)| + |\mathsf{COST}(P \setminus R_{i-\lambda}, C) - \mathsf{COST}(S_i, C)|$. The first term is upper-bounded by the Earth-Mover distance, and the second term is upper-bounded by

ϵ since S_i is a (k, ϵ)-coreset for $P \setminus R_{i-\lambda}$. So therefore $M_{i-\lambda} \cup S_i$ is a 2ϵ-coreset for the stream P, and the proof is complete after rescaling ϵ. $\square$

The previous theorem has, as special cases, streaming coresets for k-median, k-means, L_p, Cauchy estimators, Tukey estimators, etc. This the first algorithm that does not use merge-and-reduce for building coresets over streams for any of these problems. Moreover, the constant in the exponent for ϵ is small, for the example of k-median the ϵ dependence is $\epsilon^{-3} \log(1/\epsilon)$.

Part III

High-Dimensional Euclidean Coresets for Insertion-Deletion Streams

3.1 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. [50]). 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 [52]).

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 part of the thesis, 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.

3.1.1 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 3.1.1 (Main Theorem). *Fix $\epsilon \in (0, 1/2)$, positive integers k and Δ, 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 ϵ-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 3.3.6 and the proof is presented in Section 3.3.3. 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 3.1.2 (Alternative Results). *Fix $\epsilon \in (0, 1/2)$, positive integers k and Δ, 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 ϵ-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* $\mathrm{poly}(d, 1/\epsilon, L, \log k)$ *time to process 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 3.4.3 in Section 3.4 and the proof is presented therein. Both approaches can be easily extended to maintain a coreset for a general metric space.

3.1.2 Our Techniques

From a high level, both algorithms can be viewed as a combination of the ideas introduced by Frahling and Sohler [31] with the coreset construction by Chen [17].

To explain our high-level idea, we first summarize the idea of Chen [17]. In their construction, they first obtain a (α, β)-bi-criterion solution. Namely find a set of at most αk centers such that the k-median cost to these αk centers is at most $\beta\mathsf{OPT}$, where OPT is the optimal cost for a k-median solution. Around each of the αk 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 $P_1, P_2, \ldots, P_{k'}$, $\sum_i |P_i|\mathrm{diam}(P_i) \lesssim \beta\mathsf{OPT}$. They then sample a set of points from each part to estimate the cost of an arbitrary k-set from $[\Delta]^d$ up to $O(\epsilon|P_i|\mathrm{diam}(P_i)/\beta)$ additive error. Combining the samples of the k' parts, this gives an additive error of at most $\epsilon\mathsf{OPT}$ and therefore an ϵ-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 (α, β)-bi-criterion solution to the k-median problem, as it does in [17], our first idea (similar to the idea used in [44]) 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 $\sum_i |\mathcal{C}_i|\mathrm{diam}(\mathcal{C}_i) \le \beta\mathsf{OPT}$ 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 [17].

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 $\sum_p d(c_p^l, Z) - d(c_p^{l-1}, Z)$, where c_l^p 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 $|d(c_p^l, Z) - d(c_p^{l-1}, Z)|$ 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 [54, 42] or k-wise independent hash functions) to do the sampling and use a `K-Set` data structure [32] to store the sampled points in the dynamic stream.

3.1.3 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 [22, 43], approximately maitain the convex hull [20, 40], the min-volume bounding box [8, 9], maintain ϵ-nets and ϵ-approximations of a data stream [3]. 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 [39, 38, 17, 26, 27, 25]. Additionally [10, 34, 51] 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 [13]. However, all of the above methods do not work for dynamic streams.

The most relevant works to ours are those by Indyk [44], Indyk & Price [45] and Frahling & Sohler [31]. Indyk [44] 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 (α, β)-bi-criterion approximation algorithm. Indyk & Price [45] 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 [31], 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.

3.2 Preliminaries

For integer $a \leq b$, 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 Δ is of the form 2^L for some integer L, since otherwise we can always pad Δ 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 $\mathrm{cost}(P, Z) = \sum_{p \in P} d(p, Z)$ 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 [39].

Definition 3.2.1. *Let $P \subseteq [\Delta]^d$ be a point set. A small weighted set S is called an ϵ-coreset for the k-median problem, if for every set of k centers $Z \subset [\Delta]^d$ we have* [1]

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

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

[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 $\mathrm{polylog}(1/\epsilon)$ factor in the space. We discuss this point further in Section 3.6.

Through out Part III, 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 Δ 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 $i \geq 1$, 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 OPT as the optimal cost for Z^*. The proof of the following lemma is delayed to Section 3.7.

Lemma 3.2.2. *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 $d(\mathcal{C}, Z) \leq \Delta/(2^{i+1}d)$ is at most $e|Z|(L + 1)/\rho$.*

3.2.1 Outline

In Section 3.3, we introduce the coreset with negative weights. In Section 3.4, we introduce a modified construction with all positive weights. Section 3.6 comes with the final remarks.

3.3 Generally Weighted Coreset

In this section, we present our generally weighted coreset construction. In Section 3.3.1, we introduce the telescope sum representation of a point p and the coreset framework.

In Section 3.3.2, we illustrate our coreset framework with an offline construction. In Section 3.3.3 we present an one pass streaming algorithm that implements our coreset framework.

3.3.1 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,

$$\sum_{i=0}^{L} \sum_{p \in P} d(c_p^i, Z) - d(c_p^{i-1}, Z) + \sum_{p \in P} d(c_p^{-1}, Z). \tag{3.1}$$

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,

$$\begin{aligned}
\operatorname{cost}(\mathcal{G}_i, Z) &:= \sum_{p\in P} d(c_p^i, Z) - d(c_p^{i-1}, Z) \\
&= \sum_{\mathcal{C}\in\mathcal{G}_i} \sum_{p\in\mathcal{C}} d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \\
&= \sum_{\mathcal{C}\in\mathcal{G}_i} |\mathcal{C}| \left[d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right] \\
&= \sum_{\mathcal{C}\in\mathcal{G}_i} |\mathcal{C}| d(c(\mathcal{C}), Z) \\
&\quad - \sum_{\mathcal{C}'\in\mathcal{G}_{i-1}} \sum_{\mathcal{C}\in\mathcal{G}_i : \mathcal{C}\subset\mathcal{C}'} |\mathcal{C}| d(c(\mathcal{C}'), Z). \qquad (3.2)
\end{aligned}$$

For $i = -1$, we denote $\operatorname{cost}(\mathcal{G}_{-1}, Z) = |P| d(c_p^{-1}, Z)$. 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

$$\operatorname{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\mathcal{C}' : \mathcal{C}'\in\mathcal{G}_{i+1}, \mathcal{C}'\subset\mathcal{C}} \widehat{|\mathcal{C}'|}. \qquad (3.3)$$

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 3.3.1. *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 $Z \subset [\Delta]^d$,*

$$\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) := \sum_{\mathcal{C}\in\mathcal{G}_i} \widehat{|\mathcal{C}|} d(c(\mathcal{C}), Z) - \sum_{\mathcal{C}'\in\mathcal{G}_{i-1}} \sum_{\mathcal{C}\in\mathcal{G}_i:\mathcal{C}\subset\mathcal{C}'} \widehat{|\mathcal{C}|} \cdot d(c(\mathcal{C}'), Z), \tag{3.4}$$

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 3.3.2. *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 (3.3). 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 $i \in [0, L]$*

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

then S is an ϵ-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 $|\mathrm{cost}(S, Z) - \mathrm{cost}(P, Z)| \le \epsilon \mathsf{OPT}$. □

3.3.2 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 [41]. 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

$$\pi_i = \min\left(\frac{200(L+1)^2\Delta d^2}{2^i\epsilon^2 o}\ln\frac{2(L+1)\Delta^{kd}}{\rho}, 1\right) \tag{3.5}$$

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 (3.3). Denote $\mathcal{S}$ as the coreset, which contains the set of cell centers of non-zero weight.

Theorem 3.3.3. *Fix $\epsilon, \rho \in (0, 1/2)$, then with probability at least $1 - 8\rho$, the offline construction $\mathcal{S}$ is an ϵ-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 3.3.3. 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 3.3.2 that, $\mathcal{S}$ is an ϵ-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 (3.2), we have that, $\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) = \sum_{\mathcal{C} \in C_{Z'}} \widehat{|\mathcal{C}|} \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right) + \sum_{p \in \mathcal{S}_i}(d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i$. 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 $|C_{Z'}| \leq e|Z'|(L+1)^2/\rho = O(kL^2/\rho)$. By Lemma 3.2.2, $\mathcal{A}$ 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 $d(p, Z') \geq \mathrm{diam}(\mathcal{C})/(2d^{3/2})$. Therefore, $\sum_{p \in \mathcal{C} \notin C_Z} \mathrm{diam}(\mathcal{C}) \leq (2d^{3/2}) \sum_{p \in \mathcal{C} \notin C_Z} d(p, Z') \leq 20d^{3/2}\mathsf{OPT}$. By Lemma 3.3.4, with probability at least $1 - \frac{\rho}{(L+1)\Delta^{kd}}$, $|A_2 - E(A_2)| \leq \frac{\epsilon \mathsf{OPT}}{L+1}$. Since there are at most Δ^{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 $Z \subset [\Delta]^d$, $\left|\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) - \mathrm{cost}(\mathcal{G}_i, Z)\right| \leq \epsilon\mathsf{OPT}/(L+1)$. 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 $|\mathcal{S}| = O(d^4kL^4 \log\frac{1}{\rho}/\epsilon^2 + \frac{kL^2}{\rho})$. $\square$

Lemma 3.3.4. *Fix $\epsilon, \rho \in (0, 1/2)$, if a set of cells C from grid $\mathcal{G}_i$ satisfies $\sum_{\mathcal{C}\in C} |\mathcal{C}|\mathrm{diam}(\mathcal{C}) \leq \beta\mathsf{OPT}$ for some $\beta \geq 2\epsilon/(3(L+1))$, let S be a set of independent samples from the point set $\cup\{\mathcal{C} \in C\}$ with probability*

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

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

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

The proof is a straightforward application of Bernstein inequality. It is presented in Section 3.7.

3.3.3 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 (α, β)-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 3.3.4. We will show that we can split the set of cells into two parts. The first part satisfies the property that $\sum_{\mathcal{C}} |\mathcal{C}|\mathrm{diam}(\mathcal{C}) \leq \beta\mathsf{OPT}$ for some parameter β. The second part satisfies that $|\mathcal{C}|\mathrm{diam}(\mathcal{C}) \geq a\mathsf{OPT}/k$ 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 [32]. 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 αk 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 3.3.5 ([47] 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, ϵ, δ) 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 $(i, \hat{f}_i) \in H$, $f_i^2 \geq \sum_{j=1}^n f_j^2/k - \epsilon^2 \sum_{j=k+1}^n f_j^2$;
(2) if for any $i \in [n]$ and $f_i^2 \geq \sum_{j=1}^n f_j^2/k + \epsilon^2 \sum_{j=k+1}^n f_j^2$, then $(i, \hat{f}_i) \in H$;
(3) for each $(i, \hat{f}_i) \in H$, $|\hat{f}_i - f_i| \leq \epsilon \sqrt{\sum_{j=k+1}^n f_j^2}$.
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 $O(k + 1/\epsilon^2)\text{polylog}(n)$ query time.

Thus, using `HEAVY-HITTER`, we are guaranteed that the error of the number of points in heavy cells is upper bounded by ϵ times the number of points in the non-heavy cells. The first heavy hitter algorithm that achieves an l_2 guarantee is by [11], 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 [42]. Our main theorem for this section is as follows. The formal proof of this theorem is postponed to Section 3.8.

Theorem 3.3.6 (Main Theorem). *Fix $\epsilon, \rho \in (0, 1/2)$, positive integers k and Δ, 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 ϵ-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.

3.4 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 3.9. 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 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 1 `CoreSet`(S, k, ρ, ϵ): construct a ϵ-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 $\mathtt{KS}_{o,i}$ 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, $\mathtt{HH}_0, \mathtt{HH}_1, \ldots, \mathtt{HH}_L$, one for a level;
Update $(S)$:
for each update $(op, q) \in S$:
    /*$op \in$ {Insert, 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 $i$;
        $\mathtt{HH}_i$.update$(op, c_q^i)$;
        for each $o \in [O]$:
            if $h_{o,i}(q) == 1$:
                $\mathtt{KS}_{o,i}$.update$(op, c_q^i)$;
Query:
Let $o^*$ be the smallest $o$ such that no instance of $\mathtt{KS}_{o,0}, \mathtt{KS}_{o,1}, \ldots, \mathtt{KS}_{o,L}$ 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:
            $f \leftarrow$ GetFreq$(c, \mathtt{HH}_i, \mathtt{KS}_{o^*,i}, \pi_i(o^*))$;
        if $i < L$:
            $g \leftarrow \sum_{\mathcal{C}' \subset \mathcal{C}:\mathcal{C}' \in \mathcal{G}^{i+1}}$
              GetFreq $(c(\mathcal{C}'), \mathtt{HH}_{i+1}, \mathtt{KS}_{o^*i+1}, \pi_i(o^*))$;
            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 $R$.
```

Algorithm 2 GetFreq(e, HH, KS, π_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 $f_K(e)/\pi_i$.
```

3.4.1 Reformulation of the Telescope Sum

Definition 3.4.1. *A* heavy cell identification scheme $\mathcal{H}$ *is a map* $\mathcal{H} : \mathcal{G} \to \{$*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 $i \in [0, L]$, $|\{\mathcal{C} \in \mathcal{G}_i : \mathcal{H}(\mathcal{C}) = \text{heavy}\}| \leq \frac{\lambda_1 k L}{\rho}$, *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 3.4.2. *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, $d(p, Z) =$

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

3.4.2 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 3.9.9 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 π_l. Here π_l for $l \in [-1, L]$ is redefined as $\pi_l =$

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

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,

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}': \mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i}. \tag{3.6}$$

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.

3.4.3 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 [45]. The procedure is shown in Algorithm 3.4.3. 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 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 $|\text{wt}(\mathcal{C})|$ 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 3.4.3. *Fix $\epsilon, \rho \in (0, 1/2)$, positive integers k and Δ, 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 ϵ-coreset for k-median of size*

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

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

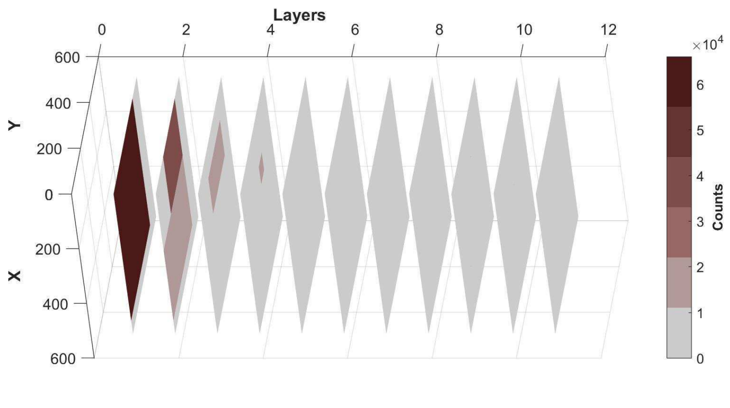
$$O\left[\frac{d^7L^7k}{\epsilon^2}\left(\rho dL + \frac{L}{\rho}\log^2\frac{dkL}{\rho\epsilon}\right)\log^2\frac{dkL}{\rho\epsilon}\right]$$

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

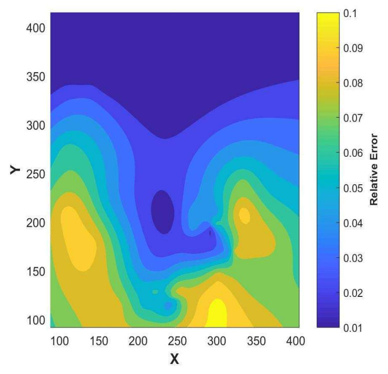
3.5 Experiments

We illustrate our construction using an offline construction on Gaussian mixture data in $\mathbb{R}^2$. As shown in Figure 3.5 in Section 3.10, 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 3.4(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 3.4(b).



(a)



(b)

Figure 3.4: (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%.

3.6 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 π_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 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.

3.7 Proofs of Section 3.3

Proof of Lemma 3.2.2. Fix an i and consider a grid $\mathcal{G}_i$. For each center z_j, denote $X_{j,\alpha}$ the indicator random variable for the event that the distance to the boundary in dimension α of grid $\mathcal{G}_i$ is at most $\Delta/(2^{i+1}d)$. Since in each dimension, if the center is close to a boundary, it contributes a factor at most 2 to the total number of close cells. It follows that the number of cells that have distance at most $\Delta/(2^{i+1}d)$ to z_j is at most,

$$N = 2^{\sum_{\alpha=1}^{d} X_{j,\alpha}}.$$

Defining $Y_{j,\alpha} = 2^{X_{j,\alpha}}$, we obtain,

$$E[N] = E\left[\prod_{\alpha=1}^{d} Y_{j,\alpha}\right] = \prod_{\alpha=1}^{d} E[Y_{j,\alpha}].$$

We have that $Pr[X_{j,\alpha} = 1] \leq 1/d$ and so we get,

$$E[Y_{j,\alpha}] \leq E\left[1 + X_{j,\alpha}\right] = 1 + E[X_{j,\alpha}] \leq 1 + 1/d.$$

Thus $E(N) = \prod_{\alpha=1}^{d} E[Y_{j,\alpha}] \leq (1+1/d)^d \leq e$. Thus the expected number of center cells is at most $(1+1/d)^d|Z| \leq e|Z|$. By Markov's inequality, the probability that we have more than $e|Z|(L+1)/\rho$ center cells in each grid is at most $\rho/(L+1)$. By a union bound, the probability that in any grid we have more than $e|Z|(L+1)/\rho$ center cells is at most ρ. □

Proof of Lemma 3.3.4. Let $L' = L + 1$. Note that for each point $p \in P$, $|d(c_p^i, Z) - d(c_p^{i-1}, Z)| \leq \Delta\sqrt{d}/2^i$. Denote $\hat{A} = \sum_{p \in S}(d(c_p^i, Z) - d(c_p^{i+1}, Z))/\pi_i$ and $A = \sum_{p \in \cup\{\mathcal{C} \in C\}}(d(c_p^i, Z) -$

$d(c_p^{i+1}, Z))$. We have that $E(\hat{A}) = A$. Let

$$X_p := \mathbb{I}_{p \in S}(d(c_p^i, Z) - d(c_p^{i+1}, Z))/\pi_i,$$

where $\mathbb{I}_{p\in S}$ is the indicator function that $p \in S$. Then we have that $Var(X_p) \le \Delta^2 d/(4^i \pi_i)$ and $b := \max_p |X_p| \le \Delta\sqrt{d}/(2^i\pi_i)$. By Bernstein's inequality,

$$\begin{aligned} Pr\left[|\hat{A} - A| > t\right] &\le 2e^{-\frac{t^2}{2|P|\Delta^2 d/(4^i\pi_i) + 2bt/3}} \\ &\le 2e^{-\frac{3\times 2^{i-1} t^2 \pi_i}{(\beta \mathsf{OPT} + \frac{t}{3})\Delta\sqrt{d}}}. \end{aligned} \tag{3.7}$$

By setting $t = \epsilon\mathsf{OPT}/L'$, we have that

$$Pr\left[|\hat{A} - A| > \frac{\epsilon\mathsf{OPT}}{L'}\right] \le 2e^{-\ln\frac{2L'\Delta^{dk}}{\rho}} \le \frac{\rho}{L'\Delta^{dk}}. \tag{3.8}$$

Thus with probability $1 - \rho/(L'\Delta^{dk})$, $\hat{A}$ is an $\epsilon\mathsf{OPT}/L'$ additive approximation to the sum $\sum_{p\in P} d(c_p^i, Z) - d(c_p^{i+1}, Z)$. □

3.8 Proof of Theorem 3.3.6

Before we prove this theorem, we first present Lemma 3.8.1 and Lemma 3.8.2. In Algorithm 1, for each level $i \in [0, L]$, let H_i be the set of cells in $\mathcal{G}_i$ whose frequencies are returned by `HEAVY-HITTER` in the `RetrieveFrequency` procedure. For each $\mathcal{C} \in H_i$, let $\widehat{|\mathcal{C}|}$ be the returned frequency of $\mathcal{C}$. Let H_i' be the set of cells in $\mathcal{G}_i$ whose frequencies are returned by a `K-set` in the `RetrieveFrequency` procedure. Then H_i and H_i' are complements in $\mathcal{G}_i$.

Lemma 3.8.1. *Let $L' = L + 1$. Fix $\epsilon, \rho \in (0, 1/2)$. Let $Z^* \subset [\Delta]^d$ be a set of optimal k-centers for the k-median problem of the input point set. For each $i \in [0, L]$, if at most ekL'/ρ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \le \Delta/(2^{i+1}d)$, then with probability $1 - \rho/L'$, the following two statements hold:*

1. $\left|\sum_{\mathcal{C}\in H_i}(\widehat{|\mathcal{C}|} - |\mathcal{C}|)\left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right)\right| \le \frac{\epsilon OPT}{2L'}$ *for every $Z \subset [\Delta]^d$.*

2. $\sum_{\mathcal{C}\in H_i'} |\mathcal{C}| diam(\mathcal{C}) \leq \beta \mathsf{OPT}$ *for* $\beta = 3d^{3/2}$

Proof of Lemma 3.8.1. Let $L' = L+1$. Fix a value $i \in [0, L]$ and then $W = \Delta/2^i$ is the width of a cell in $\mathcal{G}_i$. Since at most ekL'/ρ cells in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq W/(2d)$, then of the remaining cells, at most $2kL'/\rho$ cells can contain more than $\rho d\mathsf{OPT}/(WkL')$ points. This is because each such cells contribute at least $\frac{\rho d \mathsf{OPT}}{WkL'}\frac{W}{2d} = \frac{\rho \mathsf{OPT}}{2kL'}$ to the cost which sums to OPT. Therefore, at most $(e+2)L'k/\rho$ cells contain more than $\rho d\mathsf{OPT}/(WkL')$ points.

The number of cells in grid $\mathcal{G}_i$ is at most $N = (1+2^i)^d$ (and perhaps as few as 2^{id}, depending on the random vector v), so `HEAVY-HITTER` receives cells of at most N types. Enumerating all cells $C \in \mathcal{G}_i$ such that $|C_j| \geq |C_{j+1}|$, define $f_j = |C_j|$. Algorithm 1 sets $k' = (e+2)L'k/\rho$, and the additive error of the estimator of f_i of `HEAVY-HITTER` is given by $\epsilon'\sqrt{\sum_{j=k'+1}^{N} f_j^2}$. We know that for all $j > k'$ the value $f_j \leq \rho d\mathsf{OPT}/(WkL')$. Moreover, the sum $\sum_{j=k'+1}^{N} f_j \leq 2d\mathsf{OPT}/W$ because each point is at distance at least $W/(2d)$ to a point of Z^*. Under these two restraints, the grouping of maximal error is with $f_j = \rho d\mathsf{OPT}/(WkL')$ for $k' < j \leq k' + 2kL'/\rho$ and $f_j = 0$ for $j > k' + 2kL'/\rho$. Then the additive error becomes $\epsilon'\sqrt{2\rho/(kL')}d\mathsf{OPT}/W$.

The error from a single cell $\mathcal{C}_j$ is at most $|f_j - \hat{f}_j|\sqrt{d}W$, and `HEAVY-HITTER` gaurantees with probability $1-\delta$ that $|f_j - \hat{f}_j| \leq \epsilon'\sqrt{2\rho/(kL')}d\mathsf{OPT}/W$ for every j. Therefore to ensure total error over all k' cells is bounded by $\epsilon\mathsf{OPT}/(2L')$, we set $\epsilon' \leq \epsilon\sqrt{\frac{\rho}{8(2+e)^2kd^3L'^3}}$. Setting $\delta = \rho/L'$, the above bound holds with probability at least $1-\rho/L'$.

For the second claim, we must bound $\sum_{\mathcal{C}\in H_i'} |\mathcal{C}|$. H_i consists of the top k' cells when ordered by value of $\hat{f}_j$. This may differ from the top k' cells when ordered by value of f_j, but if j and j' change orders between these two orderings then $|f_j - f_{j'}| \leq 2\epsilon'\sqrt{2\rho/(kL')}d\mathsf{OPT}/W$. Since the sum may swap up to k' indices, the difference is bounded by $2k'\epsilon'\sqrt{2\rho/(kL')}d\mathsf{OPT}/W$. By setting $\epsilon' \leq \sqrt{\frac{\rho}{8(2+e)^2dkL'}}$, we can ensure that the difference is at most $d\mathsf{OPT}/W$. We know that $\sum_{j=k'+1}^{N} f_j \leq 2d\mathsf{OPT}/W$, and so $\sum_{\mathcal{C}\in H_i'} |\mathcal{C}| \leq 3d\mathsf{OPT}/W$. For all cells $\mathcal{C} \in \mathcal{G}_i$, $\mathrm{diam}(\mathcal{C}) = \sqrt{d}W$. Therefore $\sum_{\mathcal{C}\in H_i'} |\mathcal{C}|\mathrm{diam}(\mathcal{C}) \leq 3d^{3/2}\mathsf{OPT}$.

□

Lemma 3.8.2. *Let $L' = L+1$. In Algorithm 1, fixing $\epsilon, \rho \in (0, 1/2)$, $o \in O$ and $i \in [0, L]$, if $\mathsf{OPT}/2 \leq o \leq \mathsf{OPT}$, then with probability $1 - \rho/(L'\Delta^{kd})$, at most $\frac{(2+e)L'k}{\rho} + \frac{24d^4L'^3k}{\epsilon^2}\ln\frac{1}{\rho}$ cells of $\mathcal{G}_i$ contain a point of $S_{i,o}$.*

Proof. Similar to the proof of Lemma 3.8.1, there are at most $k' = (2+e)L'k/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ that satisfy $|\mathcal{C}| \geq \rho d\mathsf{OPT}/(Wk)$ and/or $d(\mathcal{C}, Z^*) \leq W/(2d)$. Considering the other cells, together they contain at most $2d\mathsf{OPT}/W$ points. So by a Chernoff bound, with probability $1 - \rho/(L'\Delta^{kd})$ at most $O(2d\pi_{i,o}\mathsf{OPT}/(W\rho)) \leq 24d^4L'^3k\ln\frac{1}{\rho}/\epsilon^2$ points are sampled. The claim follows since each non-empty cell must contain at least one point. □

Proof of Theorem 3.3.6. Let $L' = L+1$. W.l.o.g. we assume $\rho \geq \Delta^{-d}$, since otherwise we store the entire set of points and the theorem is proved. By Lemma 3.2.2, with probability at least $1-\rho$, for every level $i \in [0, L]$, at most ekL'/ρ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(C, Z^*) \leq \Delta/(2^{i+1}d)$. Conditioning on this event, we will show 1) in the query phase, if $o^* \leq \mathsf{OPT}$, then with probability at least $1-4\rho$, S is the desired coreset; 2) there exists $o \leq \mathsf{OPT}$ in the guesses $O = \{1, 2, 4, \ldots, \Delta^{d+1}\}$ such that with probability $1-4\rho$, none of the K-set structures return `Nil`. 1) and 2) guarantee the correctness of the algorithm. Note that one can always rescale ρ to $\rho/9$ to achieve the correct probability bound. Finally, we will bound the space, update time and query time of the algorithm.

To show 1), we first note that the coreset size is at most $O(KL)$ as desired. Then by Lemma 3.3.2, we only need to show that with probability at least $1-4\rho$, for any k-set $Z \subset [\Delta]^d$ and any level $i \in [-1, L]$,

$$|\mathrm{cost}(\mathcal{G}_i, Z) - \widehat{\mathrm{cost}}(\mathcal{G}_i, Z)| \leq \frac{\epsilon\mathsf{OPT}}{L'},$$

where the value of each $\widehat{|C|}$ is returned by `RetrieveFrequency`. For each level i, we denote C_i as the set of cells that gets frequency from a `HEAVY-HITTER` instances in the `RetrieveFrequency` procedure, and $S_i = \{p \in \mathcal{C} : \mathcal{C} \notin C_i, h_{o^*,i}(p) = 1\}$ be the set of points sampled in the rest of cells. Since $KS_{o^*,i}$ does not return `Fail`, then for each

$\mathcal{C} \in \mathcal{G}_i \backslash C_i$, $\widehat{|\mathcal{C}|} = |S_i \cap \mathcal{C}|/\pi_i(o^*)$. Fix a k-set $Z \subset [\Delta]^d$, we rewrite the cost as,

$$\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) = \sum_{\mathcal{C} \in C_i} \widehat{|\mathcal{C}|} \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right) + \sum_{p \in S_i} (d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i(o^*),$$

where $\mathcal{C}^P$ is the parent cell of $\mathcal{C}$ in grid $\mathcal{G}_{i-1}$. By Lemma 3.8.1 we have that, with probability at least $1 - \rho/L'$, for every $Z \subset [\Delta]^d$,

$$\left| \sum_{\mathcal{C} \in C_i} \widehat{|\mathcal{C}|} \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right) - \sum_{\mathcal{C} \in C_i} |\mathcal{C}| \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right) \right| \leq \frac{\epsilon \mathsf{OPT}}{2L'},$$

and that, $\sum_{\mathcal{C} \in \mathcal{G}_i \backslash C_i} |\mathcal{C}| \mathrm{diam}(\mathcal{C}) \leq 3d^{3/2}\mathsf{OPT}$. Conditioning on this event, by Lemma 3.3.4, with probability at least $1 - \rho/(L'\Delta^{kd})$,

$$\left| \sum_{p \in S_i} (d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i - \sum_{\mathcal{C} \in \mathcal{G}_i \backslash C_i} |\mathcal{C}| \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right) \right| \leq \frac{\epsilon \mathsf{OPT}}{2L'}.$$

By a union bound, we show with probability at least $1 - 4\rho$, for any k-set $Z \subset [\Delta]^d$ and any level $i \in [-1, L]$,

$$|\mathrm{cost}(\mathcal{G}_i, Z) - \widehat{\mathrm{cost}}(\mathcal{G}_i, Z)| \leq \frac{\epsilon \mathsf{OPT}}{L'},$$

as desired.

To show 2), we will consider some $\mathsf{OPT}/2 \leq o \leq \mathsf{OPT}$. By Lemma 3.8.2 with probablity at least $1 - \rho/\Delta^{kd}$, the total number of cells occupied by sample points in each level is upper bounded by $K = \frac{(2+e)L'k}{\rho} + \frac{24d^4L'^3k}{\epsilon^2}\ln\frac{1}{\rho}$. Thus by the guarantee of the `K-Set` structure, with probability at least $1 - \rho$, none of the $\mathtt{KS}_{o,0}, \mathtt{KS}_{o,1} \ldots, \mathtt{KS}_{o,L}$ will return `Fail`.

The memory requirement of the algorithm is determined by the L instances of `HEAVY-HITTER` and the dL^2 instances of K-set. By Theorem 3.3.5, each instance of `HEAVY-HITTER` requires $O\left((k' + \frac{1}{\epsilon'^2}) \log \frac{N}{\delta} \log m\right)$ bits of space. Here $N \leq (1 + \Delta/W)^d \leq \Delta^d$ and m is the maximum number of elements active in the stream. Since we require that at most one point exists at each location at the same time, we have that $m \leq N$. The parameters are set to $k' = (2+e)Lk/\rho$, $\epsilon' = \left(\epsilon\sqrt{\frac{\rho}{8(2+e)^2kd^3L^3}}\right)$, and $\delta = \rho/L$. This translates to a space bound of $O\left(dL+\log\frac{1}{\rho}\right)\frac{d^4L^5k}{\rho\epsilon^2}$ bits. For each `K-Set` data structure, it requires

$$O(KdL\log(KL/\rho)) = O\left(\frac{d^5L^4k}{\epsilon^2}+\frac{dkL^2}{\rho}\right)\log\frac{dkL}{\epsilon\rho}$$

bits of space. In total, there are $O(dL^2)$ `K-Set` instances and thus all `K-Set` instances cost $O\left(\frac{d^6L^6k}{\epsilon^2}+\frac{d^2kL^4}{\rho}\right)\log\frac{dkL}{\epsilon\rho}$ bits of space. By the same argument as in the offline case, the last paragraph of the proof of Theorem 3.3.3, the size of the coreset is at most $O((k'+K)L) = O(d^4kL^4\epsilon^{-2}+kL^2/\rho)$ points. Finally, to derandomize the fully random functions, we use Nissan's pseudorandom generator [54] in a similar way used in [42]. But our pseudo-random bits only need to fool the sampling part of the algorithm rather than whole algorithm. We consider an augmented streaming algorithm $\mathcal{A}$ that does exactly the same as in `CoreSet` but with all the `HEAVY-HITTER` operations removed. Thus all K-set instances will have identical distribution with the ones in algorithm `CoreSet`. $\mathcal{A}$ uses $O(KdL\log(KL/\rho))$ bits of space. To fool this algorithm, using Nissan's pseudo-random generator, the length of random seed to generate the hash functions we need is of size $O(KdL\log(KL/\rho)\log(|O|\Delta^d)) = O\left(\left(\frac{d^7kL^7}{\epsilon^2}+\frac{d^3kL^5}{\rho}\right)\log\frac{dkL}{\rho\epsilon}\right)$. This random seed is thus sufficient to be used in Algorithm `CoreSet`. Thus the total space used in the algorithm is $O\left(\left(\frac{d^7kL^7}{\epsilon^2}+\frac{d^3kL^5}{\rho}\right)\log\frac{dkL}{\rho\epsilon}+\frac{d^5kL^6}{\epsilon^2\rho}\right)$ bits.

Regarding the update time, for the `HEAVY-HITTER` operations, it requires $O(L\log N) = O(dL^2)$ time. For the K-set operations, it requires $|O|LO(\log(KL/\rho)) = dL^2\log(dkL/(\rho\epsilon))$ time. The de-randomized hash operation takes $O(dL)$ more time per update. The final query time is dominated by the `HEAVY-HITTER` data structure, which requires $\mathrm{poly}(d,k,L,1/\epsilon)$ time. □

3.9 Full Construction of 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 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 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.

This section is organized as follows. We reformulate the telescope sum in Subsection 4.1, provide a different construction (still with negative weights) in Subsection 4.2, modify our different construction to output non-negative weights in Subsection 4.3, and move this construction into to the streaming setting in Subsection 4.4. For simplicity of presentation, we will use $\lambda_1, \lambda_2, \ldots$ to denote some fixed positive universal constants.

3.9.1 Reformulation of the Telescope Sum

Definition 3.9.1. *A* heavy cell identification scheme $\mathcal{H}$ *is a map* $\mathcal{H} : \mathcal{G} \to \{$*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* $|\mathcal{C}| \geq \frac{2^i \rho d OPT}{k(L+1)\Delta}$ *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* $i \in [0, L]$*,* $|\{\mathcal{C} \in \mathcal{G}_i : \mathcal{H}(\mathcal{C}) = \text{heavy}\}| \leq \frac{\lambda_1 k L}{\rho}$*, 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 3.9.2. *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,

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

Let P_l be all the points with ending level $l(p) = l$. We now present the following lemmas.

Lemma 3.9.3. *Let P_i be the set of points with ending level i. Let $Z^* \subset [\Delta]^d$ be a set of optimal k-centers for the k-median problem of the input point set. Assume that for each $i \in [-1, L]$, at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \le \Delta/(2^{i+1}d)$. Then*

$$|P_i| \cdot \frac{\Delta\sqrt{d}}{2^i} \le \lambda_2 d^{3/2} \mathsf{OPT},$$

where $\lambda_2 > 0$ is a universal constant.

Before we prove this lemma, we first introduce the following lemmas to bound the cells with a large number of points.

Lemma 3.9.4. *Assume that for each $i \in [0, L]$, at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \le \Delta/(2^{i+1}d)$. Then for any $r > 0$ there are at most $\frac{(e+2r)k(L+1)}{\rho}$ cells that satisfy $|\mathcal{C}| \ge \frac{2^i \rho d \mathsf{OPT}}{rk(L+1)\Delta}$.*

Proof of Lemma 3.9.4. Let $L' = L + 1$. Fix a value $i \in [0, L]$ and then $W = \Delta/2^i$ is the width of a cell in $\mathcal{G}_i$. Since at most ekL/ρ cells in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \le W/(2d)$, then of the remaining cells, each contribute at least $\frac{\rho d \mathsf{OPT}}{rWkL'} \frac{W}{2d} = \frac{\rho \mathsf{OPT}}{2rkL'}$ to the cost, and the cost of these cells is at most OPT. Therefore there can be at most $2rL'k/\rho$ cells such that $d(\mathcal{C}, Z^*) > W/(2d)$. Along with the at most ekL'/ρ cells (by the assumption) such that $d(\mathcal{C}, Z^*) \le W/(2d)$, there are at most $(e+2r)L'k/\rho$ cells that contain at least $\rho d\mathsf{OPT}/(rWkL')$ points. $\square$

Lemma 3.9.5. *Assume that for each $i \in [0, L]$, at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \le \Delta/(2^{i+1}d)$. Then for $i \in [-1, L]$, the points of P_i can be partitioned to at most $k' = \frac{2(e+6)k(L+1)}{\rho}$ groups, $G_1, G_2, \ldots, G_{k'}$, such that for each $j \in [k']$, there exists a $\mathcal{C} \in \mathcal{G}_i$, such that $G_j \in \mathcal{C}$, $|G_j| < 5\frac{2^{i-1}\rho d OPT}{k(L+1)\Delta}$.*

Proof of Lemma 3.9.5. Let $L' = L+1$. For each heavy cell in $\mathcal{G}_i$, if the number of points falling into its non-heavy subcells (in $\mathcal{G}_{i+1}$) is less than $\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$, we group all these subcells into a single group. Let the groups formed this way be called type I, and by Property 4 of Definition 3.4.1 there are at most $(e+4)kL'/\rho$ type I groups.

For each of the remaining heavy cells in $\mathcal{G}_i$, we group its subcells into groups such that each group contains a number of points in the interval $\left[\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}, 5\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}\right)$. This can be done since each non-heavy subcell contains less than $\frac{2^{i+1}\rho d\mathsf{OPT}}{kL'\Delta} = 4\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$ points, and the total number of points contained in them is at least $\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$ (otherwise we would have formed a type I group). Let the groups formed this way be called type II. By the assumption of of Lemma 3.9.3, at most $\frac{ekL'}{\rho}$ of these non-heavy subcells are within distance $\frac{\Delta}{2^{i+2}d}$ from an optimal center of Z^*. Since each type II group contains at least $\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$ points, by the same argument as in Lemma 3.9.4, the number of type II groups further than distance $\frac{\Delta}{2^{i+2}d}$ from an optimal center is at most $\frac{8kL'}{\rho}$. We conclude that,

$$k' \le \frac{(e+4)kL'}{\rho} + \frac{(e+8)kL'}{\rho}. \qquad \square$$

Proof of Lemma 3.9.3. Let $L' = L+1$. Fix a value $i \in [-1, L]$ and then $W = \Delta/2^i$ is the upper bound of the width of a cell in $\mathcal{G}_i$. Let $G_1, G_2, \ldots G_{k'}$ be group of points satisfying Lemma 3.9.5. Thus, $\sum_{p \in P_l} \frac{\Delta\sqrt{d}}{2^i} \le \sum_{j \in [k']} \frac{2^{i+1}\rho d\mathsf{OPT}}{kL'\Delta} \cdot \frac{\Delta\sqrt{d}}{2^i} \le \frac{\lambda' kL'}{\rho} \cdot \frac{2^{i+1}\rho d\mathsf{OPT}}{kL'\Delta}\frac{\Delta\sqrt{d}}{2^i} \le \lambda_2 d^{3/2}\mathsf{OPT}$ for some universal constants λ' and λ_2. $\square$

Proof of Proposition 3.9.10. First notice that the weighted set satisfies the about condition is an ϵ-coreset. If we replace each $|\widehat{\mathcal{C}_i}|$ by the exact number of points in $|\mathcal{C}_i|$, then the new weighted set is an $(\epsilon/2)$-coreset. For each $\mathcal{C} \in \mathcal{G}$, let $b_{\mathcal{C}}$ be the new value returned by the algorithm, and b_q is the new value of a point $q \in S$. The error of the cost introduced is at most,

$$A = \sum_{i=0}^{L} \left(\sum_{\mathcal{C} \in \mathcal{G}_i\text{: heavy}} ||\widehat{\mathcal{C}}| - b_{\mathcal{C}}| + \sum_{p \in S_{i-1}} \left| \left(\frac{1}{\pi_{i-1}} - b_p \right) \right| \right) \frac{\Delta\sqrt{d}}{2^i}.$$

By the procedure, the new value of a cell is always smaller than its original value, thus

$$A = \sum_{i=0}^{L} \left(\sum_{\mathcal{C} \in \mathcal{G}_i\text{: heavy}} |\widehat{\mathcal{C}}| - b_{\mathcal{C}} + \sum_{p \in S_{i-1}} \left(\frac{1}{\pi_{i-1}} - b_p \right) \right) \frac{\Delta\sqrt{d}}{2^i} = \sum_{i=0}^{L} g_i,$$

where

$$g_i = \left(\sum_{\mathcal{C} \in \mathcal{G}_i\text{:heavy}} |\widehat{\mathcal{C}}| - b_{\mathcal{C}} + \sum_{p \in S_{i-1}} \frac{1}{\pi_{i-1}} - b_p \right) \frac{\Delta\sqrt{d}}{2^i}.$$

Let

$$f_i = \left(\sum_{\mathcal{C} \in \mathcal{G}_i\text{:heavy}} \left| |\mathcal{C}| - |\widehat{\mathcal{C}}| \right| + \sum_{\mathcal{C}' \in \mathcal{G}_{i-1}\text{:heavy}} \left| \frac{|S_{i-1} \cap \mathcal{C}'|}{\pi_{i-1}} - |P_{i-1} \cap \mathcal{C}'| \right| \right) \frac{\Delta\sqrt{d}}{2^i}.$$

Thus $f_i \leq \epsilon\mathsf{OPT}/L$ by choosing appropriate λ_6. Now consider heavy cell $\mathcal{C} \in \mathcal{G}_i$, let $s_{\mathcal{C}} = \left| b_{\mathcal{C}} - \sum_{\mathcal{C} \in \mathcal{G}_{i+1}\text{:heavy}} |\widehat{\mathcal{C}}| - \frac{|S_i \cap \mathcal{C}|}{\pi_i} \right|$. Then,

$$\begin{aligned} s_{\mathcal{C}} &= \left| b_{\mathcal{C}} - |\widehat{\mathcal{C}}| + |\widehat{\mathcal{C}}| - |\mathcal{C}| - \sum_{\mathcal{C}' \in \mathcal{G}_{i+1}\text{:heavy}} (|\widehat{\mathcal{C}'}| - |\mathcal{C}'|) - \left(\frac{|S_i \cap \mathcal{C}|}{\pi_i} - |P_i \cap \mathcal{C}| \right) \right| \\ &\leq \left| b_{\mathcal{C}} - |\widehat{\mathcal{C}}| \right| + \left| |\widehat{\mathcal{C}}| - |\mathcal{C}| \right| + \sum_{\mathcal{C}' \in \mathcal{G}_{i+1}\text{:heavy}} \left| |\widehat{\mathcal{C}'}| - |\mathcal{C}'| \right| + \left| \frac{|S_i \cap \mathcal{C}|}{\pi_i} - |P_i \cap \mathcal{C}| \right|. \end{aligned} \tag{3.9}$$

Then

$$g_i = \sum_{\mathcal{C} \in \mathcal{G}_{i-1}} s_{\mathcal{C}} \frac{\Delta\sqrt{d}}{2^i} + \left(\sum_{p \in S_{i-1}} \frac{1}{\pi_{i-1}} - b_p \right) \frac{\Delta\sqrt{d}}{2^i} \leq \frac{1}{2} g_{i-1} + \frac{1}{2} f_{i-1} + f_i. \tag{3.10}$$

Since $g_{-1} = f_{-1} = 0$, thus

$$g_i \le f_i + 3\sum_{j=0}^{i-1} 2^{j-i} f_j, \quad \text{and} \quad \sum_{i=0}^{L} g_i \le \sum_{i=0}^{L} f_i \Big(1 + \sum_{j=1}^{i} \frac{3}{2^j}\Big) \le 4\sum_{i=1}^{L} f_i \le 4\epsilon\mathsf{OPT}. \qquad \square$$

Remark 3.9.6. *The multiset of centers of heavy cells with each assigned a weight of the number of points in the cell is a $O(d^{3/2})$-coreset. This can be easily seen by removing the term of $d(c_p^L, Z) - d(c_p^{l(p)}, Z)$ from Equation* (3.9.1) *together with Lemma 3.9.3, which bounds the error introduced by this operation.*

3.9.2 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 present the following lemma that governs the correctness of sampling from the last level.

Lemma 3.9.7. *If a set of points $P_i \subset P$ satisfies $|P_i|\Delta\sqrt{d}/(2^i) \le \beta\mathsf{OPT}$ for some $\beta \ge 2\epsilon/(3(L+1))$, let S_i be an independent sample from P_i such that $p \in S_i$ 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 $1 - \rho/((L+1)Delta^{kd})$, $|\sum_{p\in S_i}(d(c_p^i, Z) - d(p, Z))/\pi_i - \sum_{p\in P_i}(d(c_p^i, Z) - d(p, Z))| \le \frac{\epsilon\mathsf{OPT}}{L+1}$.

Proof. The proof is identical to that of Lemma 3.3.4. $\square$

Lemma 3.9.8. *Consider a set of sets $\{P_i\}_{i=0}^{L}$ which satisfies $|P_i|\Delta\sqrt{d}/(2^i) \le \frac{\beta\rho}{k(L+1)}\mathsf{OPT}$ for some $\beta \ge \epsilon/(3(L+1))$. For each $i \in [0, L]$, let S_i be an independent sample from P_i with sampling probability*

$$\pi_i \ge \min\left(\frac{4a\beta k(L+1)^3\Delta\sqrt{d}}{2^i\epsilon^2\rho o}\log\frac{2}{\delta}, 1\right)$$

where $0 < o \leq a\mathsf{OPT}$ *for some* $a > 0$*, then with probability at least* $1 - \delta$*,*

$$\left|\frac{|S_i \cap P_i|}{\pi_i} - |P_i|\right| \frac{\Delta\sqrt{d}}{2^i} \leq \frac{\epsilon\rho \mathit{OPT}}{k(L+1)^2}.$$

Proof of Lemma 3.9.8. The proof is simply by Bernstein inequality. Let $t = \frac{2^i \epsilon\rho\mathsf{OPT}}{\sqrt{d}k(L+1)^2\Delta}$, $X_p := \mathbb{I}_{p\in S_i}/\pi_i$, then we have that $Var(X_p) \leq 1/\pi_i$ and $b := \max_p |X_p| \leq 1/\pi_i$. By Bernstein's inequality, for any $j \in [k']$,

$$Pr\left[\left|\frac{|P_j \cap S_i|}{\pi_i} - |P_j|\right| > t\right] \leq 2e^{-\frac{t^2}{2|\mathcal{C}|/\pi_i + 2bt/3}} \leq \delta. \qquad \square$$

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 3.9.9 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 π_l. Here π_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,

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}':\mathcal{C}'\in\mathcal{G}_{i+1}, \mathcal{C}'\subset\mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i}. \tag{3.11}$$

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.

We now state the following proposition for a coreset construction, which immediately serves as an offline coreset construction.

Proposition 3.9.9. *Let $\mathcal{H}$ be an arbitrary heavy cell identification scheme. Fix $\Omega(\Delta^{-d}) \leq \rho < 1$ and for each heavy $\mathcal{C} \in \mathcal{G}_i$ in level i, $\widehat{|\mathcal{C}|}$ is an estimation of number of points in $\mathcal{C}$ with additive error at most $\frac{\epsilon}{\lambda_5 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta}$, where $\lambda_5 > 0$ is a universal constant. Let S_l be the set of i.i.d. samples of P_l with probability $\pi_l(o)$. If $0 < o \leq \mathsf{OPT}$, then with probability at least $1 - 4\rho$, for every k-set $Z \subset [\Delta]^d$,*

$$\left| \sum_{q \in \mathcal{S}} \mathrm{wt}(q) d(q, Z) - \sum_{p \in P} d(p, Z) \right| \leq \epsilon \mathsf{OPT}.$$

And the coreset size $|\mathcal{S}|$ is

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

Proof of Proposition 3.9.9. Fix a k-set $Z \subset [\Delta]^d$. First notice that,

$$\begin{aligned}
\widehat{\mathrm{cost}}(Z) &= \sum_{q \in \mathcal{S}} \mathrm{wt}(q) d(q, Z) \\
&= \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} \left(\widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}' : \mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i} \right) d(c(\mathcal{C}), Z) + \sum_{p \in S_i} \frac{d(p, Z)}{\pi_i} \right] \\
&= \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} \widehat{|\mathcal{C}|} (d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)) + \sum_{p \in S_i} \frac{d(p, Z) - d(c_p^i, Z)}{\pi_i} \right],
\end{aligned} \tag{3.12}$$

where we denote $d(c(\mathcal{C}_{-1}^P), Z) = 0$ for convenience. Let $\mathrm{cost}(Z) = \sum_{p \in P} d(p, Z)$. Note that we can also write the true cost of Z as

$$\mathrm{cost}(Z) = \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} |\mathcal{C}| (d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P, Z))) + \sum_{p \in P_i} d(p, Z) - d(c_p^i, Z) \right].$$

We have that,

$$\widehat{\mathrm{cost}}(Z) - \mathrm{cost}(Z) = A_1 + A_2,$$

where

$$A_1 = \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C}\in\mathcal{G}_i:\mathcal{C}\text{ heavy}} (|\widehat{\mathcal{C}}| - |\mathcal{C}|)(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P, Z))) \right]$$

and

$$A_2 = \sum_{i=-1}^{L-1} \left(\sum_{p\in S_i} \frac{d(p,Z) - d(c_p^i, Z)}{\pi_i} - \sum_{p\in P_i} d(p,Z) - d(c_p^i, Z) \right).$$

Let $Z^* \subset [\Delta]^d$ be a set of optimal k-centers for the k-median problem of the input point set. By Lemma 3.2.2, with probability at most $1-\rho$, for each $i \in [0, L]$, if at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \le \Delta/(2^{i+1}d)$. Conditioning on this event, we have that, by Lemma 3.9.4 there are at most $k' = O\left(\frac{kL}{\rho}\right)$ heavy cells per level. Since for each $\mathcal{C} \in \mathcal{G}_i$, $\left||\widehat{\mathcal{C}}| - |\mathcal{C}|\right| \le \frac{\epsilon}{\lambda_5 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta}$, by choosing appropriate constant $\lambda_5 > 0$ we have

$$\begin{aligned} |A_1| &\le \sum_{i=-1}^{L-1} \left| \sum_{\mathcal{C}\in\mathcal{G}_i:\mathcal{C}\text{ heavy}} (|\widehat{\mathcal{C}}| - |\mathcal{C}|)(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P, Z))) \right| \\ &\le L \cdot k' \cdot \frac{\epsilon}{\lambda_5 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta} \cdot \frac{\Delta\sqrt{d}}{2^i} \le \frac{\epsilon \mathsf{OPT}}{2}. \end{aligned} \tag{3.13}$$

For A_2, let

$$A_{2i} = \left(\sum_{p\in S_i} \frac{d(p,Z) - d(c_p^i, Z)}{\pi_i} - \sum_{p\in P_i} d(p,Z) - d(c_p^i, Z) \right).$$

By Lemma 3.9.3, for each $i \in [-1, L-1]$, $|P_i|\Delta\sqrt{d}/(2^i) = \lambda_2(d^{3/2}\mathsf{OPT})$. Thus by Lemma 3.9.7, and choosing appropriate constants, with probability at least $1-\rho/(L+1)\Delta^{dk}$, $|A_{2i}| \le \frac{\epsilon\mathsf{OPT}}{2(L+1)}$. By the union bound, with probability at least $1-\rho$, for every level i, and every k-set $Z \subset [\Delta]^d$, $|A_{2i}| \le \frac{\epsilon\mathsf{OPT}}{2(L+1)}$. Thus $|A_2| \le \epsilon\mathsf{OPT}/2$. In total, with probability at least $1-3\rho$, $|A_1 + A_2| \le \epsilon\mathsf{OPT}$ for any k-set $Z \subset [\Delta]^d$.

The coreset size is the number of heavy cells plus the number of sampled points. The number of heavy cells is $O(kL^2/\rho)$. The expected number of sampled points per level is at most,

$$|S_i| = O\left(\frac{d^4 L^3 k}{\epsilon^2} + \frac{d^3 L^2 k}{\epsilon^2 \rho} \log\left(\frac{kL}{\rho}\right)\right) \frac{\mathsf{OPT}}{o}.$$

By a Chernoff bound, with probability at least $1-\rho/\Delta^{dk}$, for every level $i \in [0, L]$, the number of sampled points is,

$$|S_i| \leq O\left(\frac{d^4L^3k}{\epsilon^2} + \frac{d^3L^3k}{\rho\epsilon^2}\log\left(\frac{kL}{\rho}\right)\right)\frac{\mathsf{OPT}}{o}.$$

Thus the size of the coreset $\mathcal{S}$ is,

$$|\mathcal{S}| \leq O\left[\frac{d^3L^4k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\frac{\mathsf{OPT}}{o}\right].$$

$\square$

3.9.3 Ensuring Non-Negative Weights

In this section, we will 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 [45]. The procedure is shown in Algorithm 3.4.3.

Proposition 3.9.10. *Let $\mathcal{S}$ be a weighted set constructed using the Non-Negatively Weighted Construction, i.e. each heavy cell $\mathcal{C}$ has value $\widehat{|\mathcal{C}|}$ and the set of sampled points $S = S_{-1} \cup S_0 \ldots \cup S_L$ with each point in S_l has weight $1/\pi_l$. If for each heavy cell $\mathcal{C} \in \mathcal{G}_i$, $|\widehat{|\mathcal{C}|} - |\mathcal{C}|| \leq \frac{\epsilon}{\lambda_6 L d^{3/2}} \cdot \frac{2^i \rho d \mathit{OPT}}{kL\Delta}$ for some universal constant $\lambda_6 > 0$ and for each $i \in [-1, L]$ and any k-set $Z \subset [\Delta]^d$,*

$$\left|\sum_{p\in S}\mathrm{wt}(p)(d(c_p^i, Z) - d(p,Z)) - \sum_{p\in P_i}(d(c_p^i, Z) - d(p,Z))\right| \\ \leq \frac{\epsilon \mathit{OPT}}{2L},$$

and

$$\sum_{\mathcal{C}\in\mathcal{G}_i:\ heavy}\left|\frac{|S_i \cap \mathcal{C}|}{\pi_i} - |P_i \cap \mathcal{C}|\right|\frac{\Delta\sqrt{d}}{2^i} \leq \frac{\epsilon \mathit{OPT}}{L}.$$

Then on input $\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|}, \ldots, \widehat{|\mathcal{C}_{k'}|}$ and S, where k' is the number of heavy cells, the coreset output by Algorithm 3.4.3 is a (4ϵ)-coreset.

3.9.4 The Streaming Algorithm

Sampling From Sparse Cells

For the streaming algorithm, we can still use `HEAVY-HITTER` algorithms to find the heavy cells. The major challenge is to do the sampling for each point from its ending level. We do this using a combination of hash functions and `K-Set`. In Algorithm 3.9.4, we provide a procedure that recovers the set of points from cells with a small number of points and ignore all the heavy cells. The guarantee is,

Theorem 3.9.11. *Given as input a set of dynamically updating streaming points $P \subset [N]$, a set of mutually disjoint cells $C \subset [M]$, whose union covers the region of P. Algorithm 3.9.4 outputs all the points in cells with less than β points or output* ***Fail****. If with the promise that at most α cells from C contain a point of P, then the algorithm outputs* ***Fail*** *with probability at most δ. The algorithm uses $O(\alpha\beta(\log(M\beta) + \log N)\log N \log(\log N\alpha\beta/\delta)\log(\alpha\beta/\delta))$ bits in the worst case.*

The high level idea of this algorithm is to hash the original set of points to a universe of smaller size. For cells with less points, the collision rate is much smaller than cells with more points. To recover one bit of a point, we update that bit and the cell ID and also its hash tag to the `K-Set`-data structure. If there are no other points with hash values colliding with this point, the count of that point is simply 1. If this is the case, we immediately recover the bit. By repeating the above procedure once for each bit, we can successfully recover the set of points with no colliding hash tags. For those points with colliding hash tags, we simply ignore them. Each point has a constant probability to collide with another point, thus not be in the output. By running the whole procedure $O(\log(\alpha\beta/\epsilon))$ times in parallel, we reduce the probability to roughly ϵ for each point in cells with less than β points. To formally prove Theorem 3.9.11, we first prove the following lemma, which is the guarantee of Algorithm 3.9.4.

Lemma 3.9.12. *Given input a set of dynamically updating streaming points $P \subset [N]$, a set of mutually disjoint cells $C \subset [M]$, whose union covers the region of P. Algorithm 3.9.4 outputs a set of points in cells with less than β points or output* ***Fail****. If with* *the promise that at most α cells from C contain a point of P, then the algorithm outputs* `Fail` *with probability at most δ. Conditioning on the event that the algorithm does not output* `Fail`*, each point p from cell with less than β points is in the output with marginal probability at least* 0.9*. The algorithm uses $O(\alpha\beta(\log(M\beta)+\log N)\log N\log(\log N\alpha\beta/\delta))$ bits in the worst case.*

Proof. We prove this lemma by showing that (a) if a point $p \in P$ contained in cell $\mathcal{C}$, with $|\mathcal{C} \cap P| \leq \beta$, then with probability at least 0.99, there are no other points $p' \in \mathcal{C} \cap P$ with $H(p) = H(p')$, (b) conditioning on the event that the algorithm does not output `Fail`, then for any cell $\mathcal{C} \in C$, if a point $p \in \mathcal{C}$ such that no other points in $\mathcal{C} \cap P$ has the same hash value $H(p)$, then p is in the output and (c) the algorithm outputs `Fail` with probability at most δ. The correctness of the algorithm follows by (a), (b) and (c).

To show (a), consider any cell $\mathcal{C} \in C$ with $|\mathcal{C}| \leq \beta$, let $p \in \mathcal{C}$ with hash value $H(p)$. Since H is 2-wise independent, the expected number of other points hashed to the same hash value $H(p)$ is at most $\beta/U = 1/100$. By Markov's inequality, with probability at least 0.99, no other point in $\mathcal{C}$ is hashed to $H(p)$.

To show (b), notice that if the algorithm does not output `Fail`, then for a given cell $\mathcal{C}$, let c be its ID, and p_j be the j-th bit of point p. Then (c, h, p_j) has 1 count and $(c, h, 1 - p_j)$ has 0 count for each $j \in [t]$, where $t = \lceil \log N \rceil$. Thus we can uniquely recover each bit of point p, hence the point p.

For (c), since there are at most α cells, there are at most $2\alpha U = O(\alpha\beta)$ many different updates for each `KS` structure. Therefore, with probability at most $\frac{\delta}{t}$, a single `KS` instance outputs `Fail`. By the union bound, with probability at least $1-\delta$, no `KS` instance outputs `Fail`.

Finally, the space usage is dominated by the `KS` data structures. Since the input data to `KS` is from universe $[M] \times [U] \times \{0, 1\}$, each `KS` structure uses space $O(\alpha\beta(\log(M\beta) + \log N)\log(t\alpha\beta/\delta))$ bits of memory, the total space is $O(\alpha\beta(\log(M\beta)+\log N)\log N\log(t\alpha\beta/\delta))$.

□

Proof of Theorem 3.9.11. Each instance of `SparseCellsSingle` fails with probability at most $\delta/(4A)$, where A is the number independent `SparseCellsSingle` instances. By the union bound, with probability at least $1 - \delta/4$, none of them output `Fail`. Conditioning on this event, the random bits of the hash functions of each `SparseCellsSingle` instance are independent, thus by Lemma 3.9.12 a fixed point $p \in \mathcal{C}$ with $|\mathcal{C}| \leq \beta$ is in the output with probability at least $10^{-\log \frac{4\alpha\beta}{\delta}} \leq \delta/(4\alpha\beta)$. Since there are at most $\alpha\beta$ points in cells with less than β points, by the union bound we conclude that with probability at least $1 - \delta/4$, every point in cells with less than β points is in the output set S. In sum, with probability at least $1 - \delta/2$, S contains all the desired points.

The other `KS` instance outputs `Fail` with probability at most $\delta/2$. Thus if T is not `Fail`, then T contains the exact number of points of each cell. If any desired point is not in S, then $|\mathcal{C}| > |\mathcal{C} \cap S|$, we output `Fail`. This happens with probability at most δ under the gaurantee of the `KS`structure.

Since each `SparseCellsSingle` instance uses $O(\alpha\beta(\log(M\beta)+\log N)\log N \log(tA\alpha\beta/\delta))$ bits of space, the final space of the algorithm is $O(\alpha\beta(\log(M\beta)+\log N)\log N \log(t\alpha\beta/\delta)\log(\alpha\beta/\delta))$.

□

Algorithm 4 `SparseCells`$(N, M, \alpha, \beta, \delta)$: input the point sets $P \subset [N]$ and set of cells $C \subset [M]$ such that at most α cells containing a point, output the set of points in cells with less than β points.

```
Let $A \leftarrow \log \frac{4\alpha\beta}{\delta}$;
Let $R_1, R_2, \ldots, R_A$ be the results of independent instances of SparseCellsSingle$(N, M, \alpha, \beta, \delta/(4A))$ running in parallel;
Let $T$ be the results of another parallel KS structure with space parameter $\alpha$ and error $\delta/2$ and with input as the cell IDs of points in $P$;    /*$T$ returns the exact counts of each cell*/
if any of the data structures returns Fail:
  | return Fail;
Let $S \leftarrow R_1 \cup R_2 \cup \ldots R_A$;
if $\exists$ set $\mathcal{C} \in T$ with $|\mathcal{C}| \leq \beta$ and $|\mathcal{C}| \neq |\mathcal{C} \cap S|$:
  | return Fail;
return $S$;
```

The Algorithm

With the construction of algorithm `SparseCells`, we now have all the tools for the streaming coreset construction. The streaming algorithm is composed by $O(L)$ levels of `HEAVY-HITTER` instances, which serve as a heavy cell identifier and by $O(L)$ levels

Algorithm 5 `SparseCellsSingle`$(N, M, \alpha, \beta, \delta)$: input the point sets $P \subset [N]$ and set of cells $C \subset [M]$ such that at most α cells containing a point, output the set of points in cells with less than β points.

```
    Initization:
    $U \leftarrow 100\beta$ .
    $t \leftarrow \lceil \log N \rceil$;
    $H : [N] \to [U]$, 2-wise independent;
    K-Set structures $\mathtt{KS}_1, \mathtt{KS}_2, \dots \mathtt{KS}_t$ with space parameter $2\alpha U$ and probability $\frac{\delta}{t}$;
    Update($p, op$):    /*$op \in \{\mathtt{Insert}, \mathtt{Delete}\}$*/
    $c \leftarrow$ cell ID of $p$;
    for $i \in [t]$:
            /*A point $p$ is represented as $(p_1, p_2, \dots, p_t)$*/;
        $p_i \leftarrow$ the $i$-th bit of point $p$;
        $\mathtt{KS}_i$.update($(c, H(p), p_i), op$);
    Query:
    if for any $i \in [t]$, $KS_i$ returns Fail:
        return Fail ;
    $S \leftarrow \emptyset$;
    for each $(c, h, p_1)$ in the output of $KS_1$:
        if $(c, h, p_j) \notin KS_j$ for some $j \in [t]$:
                /*A checking step, may not happen at all*/;
            return Fail ;
        Let $s(c, h, p_j)$ be the counts of $(c, h, p_j)$ in $\mathtt{KS}_j$;
        if $s(c, h, p_j) = 1$ and $s(c, h, 1 - p_j) = 0$ for each $j \in [t]$:
            $p \leftarrow (p_1, p_2, \dots, p_t)$;
            $S \leftarrow S \cup \{p\}$;
return $S$
```

of `SparseCells` instances, which sample the points from their ending levels. The full algorithm is stated in Algorithm 6. The guarantee of the algorithm is stated in the following theorem.

Theorem 3.9.13. *Fix $\epsilon, \rho \in (0, 1/2)$, positive integers k and Δ, 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 ϵ-coreset for k-median of size $O\left[\frac{d^3L^4k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$, where $L = \log \Delta$. The algorithm uses $O\left[\frac{d^7L^7k}{\epsilon^2}\left(\rho dL + \frac{1}{\rho}\log^2\frac{dkL}{\rho\epsilon}(\log\log\frac{dkL}{\rho\epsilon} + L)\right)\log^2\frac{dkL}{\rho\epsilon}\right]$ 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.*

Proof. W.l.o.g. assume $\rho \geq \Delta^{-d}$, since otherwise we can store the entire set of points. In the sequel, we will prove the theorem with parameter $O(\rho)$ and $O(\epsilon)$. It translates to ρ and ϵ directly by scaling and with losing at most a constant factor in space and time bounds. By Lemma 3.2.2, with probability at least $1-\rho$, for every level $i \in [0, L]$, at most ekL/ρ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(C, Z^*) \leq \Delta/(2^{i+1}d)$. We condition on this event for the following proof.

We first show that the `HEAVY-HITTER` instances faithfully implement a *heavy cell identification scheme*. First note that with probability at least $1-\rho$, all `HEAVY-HITTER` instances succeed. Conditioning on this event for the following proof. As shown in the proof of Lemma 3.8.1, by setting $\epsilon' = \epsilon\sqrt{\frac{\rho}{\lambda_7 k d^3 L^3}}$ and $k' = \lambda_8 kL/\rho$, for appropriate positive universal constants λ_7, λ_8, then the additive error to each cell is at most $\frac{\epsilon}{\lambda_9 d^{3/2} L} \cdot \frac{2^i d\mathsf{OPT}}{kL\Delta}$ for some universal constant λ_9, which matches the requirement of Proposition 3.9.10. For each cell $\mathcal{C}$ with at least $2^i\rho d\mathsf{OPT}/(k(L+1)\Delta)$ points, by Lemma 3.9.4 it must be in the top $(e+2)k(L+1)/\rho$ cells. For each cell $\mathcal{C}'$ with at least $2^{i-1}\rho d\mathsf{OPT}/(k(L+1)\Delta)$ points, it must be in the top $(e+4)k(L+1)/\rho$ cells. Since the additive error is $\frac{\epsilon}{\lambda_7 d^{3/2}(L+1)} \cdot \frac{2^i d\mathsf{OPT}}{k(L+1)\Delta} \ll \frac{1}{2}\frac{2^i d\mathsf{OPT}}{k(L+1)\Delta}$. Thus $\mathcal{C}$ is in the output of the `HEAVY-HITTER` instances, since otherwise $\widehat{|\mathcal{C}|} \leq \frac{1}{2}\frac{2^i d\mathsf{OPT}}{k(L+1)\Delta} + \frac{\epsilon}{\lambda_7 d^{3/2}(L+1)} \cdot \frac{2^i d\mathsf{OPT}}{k(L+1)\Delta}$ contradicts the error bound (by choosing sufficiently large λ_7). Thus the algorithm faithfully implements a heavy cell identification scheme.

Now we show that if there exists an $o \leq \mathsf{OPT}$ such that no instance of `SparseCells` outputs `Fail`, then the result is a desired $O(\epsilon)$-coreset. This follows by Proposition 3.9.9 and Proposition 3.9.10. Then we note that the coreset size is upper bounded by $O\left[\frac{d^3L^4k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$ as desired.

Next we show that there exists an $\mathsf{OPT}/2 \leq o^* \leq \mathsf{OPT}$ that with probability at least $1-\rho$, no `SparseCells` instance $\mathtt{SC}_{o^*,i}$ outputs `Fail`. By Chernoff bound, with probability at least $1 - O(\rho)$, as also shown in the proof of Proposition 3.9.9, per level at most $O\left[\frac{d^3L^4k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\frac{\mathsf{OPT}}{o}\right]$ cells is occupied by a point. And at most $O\left[\frac{d^3L^2}{\epsilon^2}\left(\rho d + \log\frac{kL}{\rho} + \frac{\rho}{kL}\log\frac{L}{\rho}\right)\right]$ points is sampled per light cell. Conditioned on this fact and that each instances fails with probability at most $O(\rho/(dL))$, with probability at least $1 - O(\rho)$, no nstance $\mathtt{SC}_{o^*,i}$ fails.

Lastly, we bound the space usage and update/query time. For the HEAVY-HITTER instances, the total space used is $O\left(dL + \log\frac{1}{\rho}\right)\frac{d^4L^5k}{\rho\epsilon^2}$ bits, analogous to the proof of Theorem 3.3.6. Each SparseCells instance uses space $O\left[\frac{d^5L^4r^2k}{\epsilon^2}\left(\rho dL + \frac{r^2(\log r + L)}{\rho}\right)\right]$, where $r = \log\frac{dkL}{\rho\epsilon}$. The total space bound is $O\left[\frac{d^6L^6r^2k}{\epsilon^2}\left(\rho dL + \frac{r^2(\log r + L)}{\rho}\right)\right]$ bits. As a same argument in the proof of Theorem 3.3.6, the cost of de-randomization introduce an additional dL factor. Thus, the final space bound is $O\left[\frac{d^7L^7r^2k}{\epsilon^2}\left(\rho dL + \frac{r^2(\log r + L)}{\rho}\right)\right]$ bits. The query time and update time is similar to that of Theorem 3.3.6 thus poly $\left(d, L, \frac{1}{\epsilon}, \frac{1}{\rho}, k\right)$ and poly $\left(d, L, \frac{1}{\epsilon}, \log k\right)$.

$\square$

3.10 Synthetic Dataset

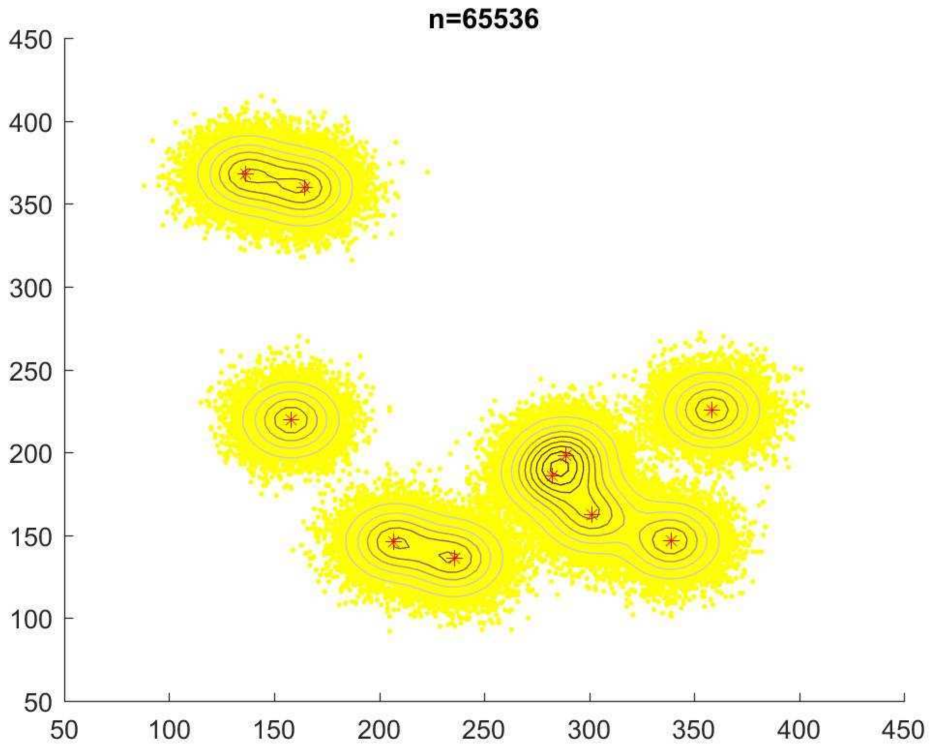


Figure 3.5: 65536 points are drawn from a Gaussian Mixture distribution. The contours illustrate the PDF function.

Algorithm 6 `PositiveCoreSet`(S, k, ρ, ϵ): construct a ϵ-coreset for dynamic stream S.

```
Initization:
Initialize a grid structure;
$O \leftarrow \{1, 2, 4, \dots, \sqrt{d}\Delta^{d+1}\}$; $L \leftarrow \lceil \log \Delta \rceil$; $\pi_i(o) \leftarrow \min\left(\frac{\lambda_3 d^2 \Delta L^2}{2^l \epsilon^2 o} \log\left(\frac{2L\Delta^{dk}}{\rho}\right) + \frac{\lambda_4 d^2 k L^3 \Delta}{2^i \epsilon^2 \rho o} \log \frac{30kL^2}{\rho^2}, 1\right)$; $\alpha \leftarrow O\left[\frac{d^3 L^4 k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$, $\beta \leftarrow O\left[\frac{d^3 L^2}{\epsilon^2}\left(\rho d + \log\frac{kL}{\rho} + \frac{\rho}{kL}\log\frac{L}{\rho}\right)\right]$, $\epsilon' \leftarrow \epsilon\sqrt{\frac{\rho}{\lambda_7 k d^3 L^3}}$; $m \leftarrow 0$;
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 SparseCells$(\Delta^d, (1+2^i)^d, \alpha, \beta, O(\rho/(dL)))$ instances SC$_{o,i}$;
Initialize HEAVY-HITTER$(\Delta^d, 10Lk/\rho, \epsilon', \rho/L)$ instances, HH$_0$, HH$_1, \dots,$ HH$_{L-1}$, one for a level;

    Update $(S)$:
    for each update $(op, q) \in S$:
    |                   /*$op \in \{$Insert, 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 $i$;
    |   |   HH$_i$.update$(op, c_q^i)$;
    |   |   for each $o \in [O]$:
    |   |   |   if $h_{o,i}(q) == 1$:
    |   |   |   |   SC$_{o,i}$.update$(op, c_q^i)$;
    Query:
    Let $o^*$ be the smallest $o$ such that no instance of SC$_{o,0}$, SC$_{o,1}, \dots,$ SC$_{o,L}$ returns Fail;
    $S \leftarrow \{\}$;
    $\mathcal{C}_{-1} \leftarrow$ the cell of the entire space $[\Delta]^d$; $\widehat{|\mathcal{C}_{-1}|} \leftarrow m$;
    for $i \in [0, L-1]$:
    |   $C_i \leftarrow$ HH$_i$.query().top$((e+4)(L+1)k/\rho)$;
    |   Remove cells $\mathcal{C}$ from $C_i$ if $\mathcal{C}^P(\mathcal{C}) \notin C_{i-1}$, where $\mathcal{C}^P(\mathcal{C})$ is the parent cell of $\mathcal{C}$ in level $i-1$;
    |   $B_i \leftarrow$ SC$_{o^*,i}$.query();
    |   $S_i \leftarrow \{p \in B_i : \mathcal{C}(p, i-1) \in C_{i-1}$ AND $\mathcal{C}(p, i) \notin C_i\}$;
    |   Each point in $\mathcal{S}_i$ receives weight $1/\pi_i(o^*)$;
    |   $S \leftarrow S \cup S_i$;
$k' \leftarrow \sum_{i \in [0,L]} |C_i|$;
Let $\{\mathcal{C}_1, \mathcal{C}_2, \dots \mathcal{C}_{k'}\} = \cup_{i \in [0,L]} C_i \cup \{\mathcal{C}_{-1}\}$ be the set of heavy cells;
Let $\left\{\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|}, \dots \widehat{|\mathcal{C}_{k'}|}\right\}$ be the estimated frequency of each cell;
$R \leftarrow$ RectifyWeights$(\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|}, \dots \widehat{|\mathcal{C}_{k'}|}, S)$;
return $R$.
```

Bibliography

[1] Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++: A clustering algorithm for data streams. *J. Exp. Algorithmics*, 17:2.4:2.1–2.4:2.30, May 2012.

[2] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 852–863. Morgan Kaufmann, 2004.

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

[4] Artem Barger and Dan Feldman. k-means for streaming and distributed big sparse data. *SDM'16 and arXiv preprint arXiv:1511.08990*, 2015.

[5] Jon Louis Bentley and James B Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[6] Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *CoRR*, abs/1612.00889, 2016.

[7] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In *Proceed-*

ings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, pages 26–40. SIAM, 2011.

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

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

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

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

[12] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 30–39, New York, NY, USA, 2003. ACM.

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

[14] Ke Chen. A constant factor approximation algorithm for k -median clustering with outliers. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 826–835. SIAM, 2008.

[15] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

[16] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, August 2009.

[17] Ke Chen. 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.

[18] Kenneth L. Clarkson and David P. Woodruff. Sketching for m-estimators: A unified approach to robust regression. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 921–939, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[19] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 163–172. ACM, 2015.

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

[21] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for lp regression. *CoRR*, abs/0707.1714, 2007.

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

[23] D. Feldman, M. Monemizadeh, and C. Sohler. A PTAS for k-means clustering based on weak coresets. In *SoCG*, 2007.

[24] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *STOC*, pages 569–578, 2011.

[25] Dan Feldman and Michael Langberg. 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*, pages 569–578, 2011.

[26] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. 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*, pages 11–18, 2007.

[27] Dan Feldman, Melanie Schmidt, and Christian Sohler. 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*, pages 1434–1453, 2013.

[28] Dan Feldman, Melanie Schmidt, and Christian Sohler. 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*, pages 1434–1453. SIAM, 2013.

[29] Dan Feldman and Leonard J Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1343–1354. SIAM, 2012.

[30] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. In *Proc. 37th Annu. ACM Symp. on Theory of Computing (STOC)*, pages 209–217, 2005.

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

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

[33] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003.

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

[35] S. Har-Peled. Geometric approximation algorithms. Manuscript, 2009.

[36] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete Comput. Geom.*, 37(1):3–19, 2007.

[37] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, 2004.

[38] Sariel Har-Peled and Akash Kushal. 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*, pages 126–134, 2005.

[39] Sariel Har-Peled and Soham Mazumdar. 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*, pages 291–300, 2004.

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

[41] Piotr Indyk. *High-dimensional computational geometry.* PhD thesis, Citeseer, 2000.

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

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

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

[45] Piotr Indyk and Eric Price. 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*, pages 627–636. ACM, 2011.

[46] M. Langberg and L. J. Schulman. Universal ε approximators for integrals. *To appear in proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

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

[48] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences (JCSS)*, 62, 2001.

[49] Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. In *SODA*, pages 309–318, 2000.

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

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

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

[53] V N. Vapnik and A Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theoretical Probabibility and its Applicactions*, 17:264–280, 01 1971.

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

[55] Dimitris Papailiopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 997–1006. ACM, 2014.

[56] Nariankadu D Shyamalkumar and Kasturi Varadarajan. Efficient subspace approximation algorithms. In *SODA*, volume 7, pages 532–540. Citeseer, 2007.

[57] David E Tyler. A distribution-free m-estimator of multivariate scatter. *The Annals of Statistics*, pages 234–251, 1987.

[58] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Prob. Appl.*, 16:264–280, 1971.

[59] Z. Zhang. M-estimators. *http://research. microsoft.com/en-us/um/people/zhang/INRIA/ Publis/Tutorial-Estim/node20.html*, [accessed July 2011].