Fast, Space-Optimal Streaming Algorithms for Clustering and Subspace Embeddings

Vincent Cohen-Addad^{*} Liudeng Wang[†] David P. Woodruff[‡] Samson Zhou[§]

Abstract

We show that both clustering and subspace embeddings can be performed in the streaming model with the same asymptotic efficiency as in the central/offline setting.

For (k, z)-clustering in the streaming model, we achieve a number of words of memory which is independent of the number n of input points and the aspect ratio Δ , yielding an optimal bound of $\tilde{O}\left(\frac{dk}{\min(\varepsilon^4,\varepsilon^{z+2})}\right)$ words for accuracy parameter ε on d-dimensional points. Additionally, we obtain amortized update time of $d \log(k) \cdot \operatorname{polylog}(\log(n\Delta))$, which is an exponential improvement over the previous $d \operatorname{poly}(k, \log(n\Delta))$. Our method also gives the fastest runtime for (k, z)-clustering even in the offline setting.

For subspace embeddings in the streaming model, we achieve $\mathcal{O}(d)$ update time and spaceoptimal constructions, using $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right)$ words for $p \leq 2$ and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right)$ words for p > 2, showing that streaming algorithms can match offline algorithms in both space and time complexity.

^{*}Google Research. E-mail: cohenaddad@google.com.

[†]Texas A&M University. E-mail: eureka@tamu.edu

[‡]Carnegie Mellon University. E-mail: <u>dwoodruf@andrew.cmu.edu</u>.

[§]Texas A&M University. E-mail: samsonzhou@gmail.com.

1 Introduction

The streaming model of computation is a powerful framework for processing large-scale datasets that are too vast to be stored in memory. In the streaming setting, algorithms must quickly make decisions based on a sequence of updates that are irrevocably discarded after processing, without the possibility of revisiting past data. The primary challenge is to design algorithms that can efficiently approximate, compute, or detect properties of the underlying dataset, while ensuring both fast update time and using memory that is sublinear in both the dataset size and the length of the data stream. This model is well-suited for applications in data summarization, where the goal is to extract meaningful insights from large, evolving datasets. Such applications include clustering, which is widely applied in domains such as customer segmentation, anomaly detection in network traffic, and pattern recognition in sensor networks, as well as subspace embeddings, which reduce dimensionality in areas including image processing, recommendation systems, and natural language processing. Despite significant progress, it remained open whether such fundamental tasks inherently require more space or time in the streaming model compared to their offline counterparts.

Clustering and data streams. Clustering is a fundamental problem that seeks to partition a dataset so that similarly structured points are grouped together and differently structured points are separated. Variations of clustering are used across a wide range of applications, including bioinformatics, combinatorial optimization, computational geometry, computer graphics, data science, and machine learning. In the Euclidean (k, z)-clustering problem, the input is a set X of n points $x_1, \ldots, x_n \in \mathbb{R}^d$, a cluster parameter k > 0, and a positive integer exponent z > 0, and the goal is to minimize the clustering cost across all sets C of at most k centers:

$$\min_{\mathcal{C} \subset \mathbb{R}^d, |\mathcal{C}| \le k} \operatorname{cost}(X, \mathcal{C}) := \min_{\mathcal{C} \subset \mathbb{R}^d, |\mathcal{C}| \le k} \sum_{i=1}^n \min_{c \in \mathcal{C}} \|x_i - c\|_2^z.$$

For z = 1 and z = 2, respectively, the (k, z)-clustering problem corresponds to the well-known k-median and k-means clustering problems.

For clustering in the standard insertion-only streaming model, the points x_1, \ldots, x_n of X arrive sequentially along with an input accuracy parameter $\varepsilon > 0$. The goal is for each time $t \in [n]$ to find a clustering C_t on the prefix X_t of the first t points of X with cost that is a $(1 + \varepsilon)$ -multiplicative approximation of the optimal clustering of X_t . To that end, note that a set C of k centers implicitly defines the overall clustering, since each point is assigned to the closest cluster center. In fact, it is not feasible to directly output a label for every point in X using space sublinear in n = |X|, and so in some sense, outputting C is the most reasonable objective we can hope for. We also remark that due to bit representation, input points cannot be represented in arbitrary precision and thus we generally rescale the points in X to lie within the grid $\{1, \ldots, \Delta\}^d$.

Subspace embeddings and data streams. Subspace embeddings are a fundamental tool in dimensionality reduction, enabling the preservation of algebraic and geometric structure, while significantly reducing data dimensionality. The problem has been extensively studied, with applications spanning theoretical computer science, machine learning, randomized linear algebra, scientific computing, and data science. In the subspace embedding problem, the input is once again a set A of n points $x_1, \ldots, x_n \in \mathbb{R}^d$, which induces a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, where typically $n \gg d$. Given an accuracy parameter $\varepsilon > 0$, the goal is to construct a matrix \mathbf{M} such that for all vectors $\mathbf{x} \in \mathbb{R}^d$,

$$(1-\varepsilon)\|\mathbf{M}\mathbf{x}\|_p \le \|\mathbf{A}\mathbf{x}\|_p \le (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_p,$$

where $\|\cdot\|_p$ denotes the L_p norm of a vector. In the standard insertion-only streaming model, the rows of **A** arrive sequentially and at each time $t \in [n]$, the goal is to find a subspace embedding for the matrix \mathbf{A}_t consisting of the first t rows of **A**.

Streaming algorithms and coresets. The most common approach for both fast and spaceefficient streaming algorithms for data summarization problems such as (k, z)-clustering and subspace embeddings is to collect a small weighted subset of the points from the input dataset. For clustering, this forms a (strong) coreset Z_t at time t, which is a data structure that preserves the clustering cost for any choice C of k centers, i.e., $cost(X_t, C) \approx cost(Z_t, C)$. We can then perform an efficient (k, z)clustering algorithm on each coreset Z_t to identify an optimal or near-optimal set of cluster centers. For subspace embeddings, this forms a coreset \mathbf{M}_t at time t, from which we can approximately extract $\|\mathbf{A}_t \mathbf{x}\|_2$ for any $\mathbf{x} \in \mathbb{R}^d$, i.e., $\|\mathbf{M}_t \mathbf{x}\|_2 \approx \|\mathbf{A}_t \mathbf{x}\|_2$. Naturally, smaller coreset constructions correspond to both faster and more space-efficient algorithms.

In the offline setting where the input dataset X is given upfront and there are no space restrictions, there exist coreset constructions for both clustering and subspace embeddings that sample a number of weighted points that is independent of the size of the input. For (k, z)-clustering, there exist coreset constructions [CSS21, CLSS22, HLW23, CLS⁺22] that sample $\tilde{\mathcal{O}}\left(\min\left(\frac{1}{\varepsilon^2} \cdot k^{2-\frac{z}{z+2}}, \frac{1}{\min(\varepsilon^4, \varepsilon^{2+z})} \cdot k\right)\right)$ weighted points of X^1 , where we use $\tilde{\mathcal{O}}(f)$ for a function f to denote $f \cdot \operatorname{polylog}(f)$. Similarly for subspace embeddings, there exist coreset constructions that sample $\tilde{\mathcal{O}}\left(\frac{d}{\varepsilon^2}\right)$ weighted rows of **A** [DMM06a, DMM06b, Mag10, Woo14]. These coreset constructions have size that is *independent* of the size n of the input dataset, which is especially important as modern datasets often consist of hundreds of millions of points.

On the other hand, despite a long line of work on clustering in the streaming model [HM04, HK07, Che09, FL11, BFLR19, CWZ23, WZZ23], all known results achieved space overheads with dependencies in n, compared to the offline setting. A natural question is thus:

Is there an inherent *space complexity* cost for data summarization problems, such as (k, z)-clustering and subspace embeddings, in the streaming model?

Input size and runtime. We remark that even in the offline setting, where the entire input is given, $\Omega(nd)$ runtime is necessary simply to process a dataset of size n in d-dimensional space. Thus, $\Omega(nd)$ runtime is necessary for both (k, z)-clustering and subspace embeddings without additional assumptions on the input. The lower bound is matched by an offline algorithm that uses $\mathcal{O}(nd)$ runtime for subspace embeddings by [CW13] and nearly matched, up to a logarithmic factor, by an offline algorithm that uses $\mathcal{O}(nd \log(n\Delta))$ runtime for (k, z)-clustering by [DSS24]. We ask:

Is there an inherent *time complexity* cost for data summarization problems, such as (k, z)-clustering and subspace embeddings, in the streaming model?

1.1 Our Contributions

We present a unified technique that resolves the above questions for both (k, z)-clustering and subspace embeddings, showing there is no overhead for working in the streaming model for either space complexity or time complexity. At a high level, the technique is summarized in Figure 1. However, efficiently implementing each of the steps is challenging and requires structural properties to each specific problem. We provide a technical overview of these steps in Section 1.2.

¹Note that the bound is actually $\tilde{\mathcal{O}}\left(\min\left(\frac{k^{3/2}}{\varepsilon^2}, \frac{k}{\min(\varepsilon^4, \varepsilon^{2+z})}\right)\right)$ for k-means and $\tilde{\mathcal{O}}\left(\min\left(\frac{k^{4/3}}{\varepsilon^2}, \frac{k}{\min(\varepsilon^4, \varepsilon^{2+z})}\right)\right)$ for k-median.

- (1) Given a stream S of length n, use a crude sampling scheme to generate a stream S' of length $n^{1-\Omega(1)}$.
- (2) Use S' and a refined sampling scheme that takes longer to compute but generates a stream S'' of length polylog(n), omitting dependencies in other parameters.
- (3) Finally, run merge-and-reduce on \mathcal{S}'' to produce a coreset \mathcal{C} .
- (4) Produce an efficient encoding of C using a constant-factor global encoding.

Fig. 1: High-level summary of our approach

Clustering. We describe the applications of our technique in Figure 1 in the context of (k, z)clustering. Our first main result is the following:

Theorem 1.1 (Fast and space-optimal clustering). Given a set X of n points on $[\Delta]^d$ and an accuracy parameter $\varepsilon \in (0, 1)$, there is a one-pass insertion-only streaming algorithm that uses $\tilde{O}\left(\frac{dk}{\min(\varepsilon^4,\varepsilon^{2+z})}\right)$ words of space and $d\log(k) \cdot \operatorname{polylog}(\log(n\Delta))$ amortized update time and outputs a $(1 + \varepsilon)$ -strong coreset of X.

Theorem 1.1 has a number of implications. First, the amortized runtime of Theorem 1.1 is an *exponential* improvement for dependencies in k and $\log n$ over existing work [HK20, BCLP23, BCG⁺24]. Indeed, the fastest existing algorithm due to [BCLP23, BCG⁺24] uses amortized update time $k \cdot \text{polylog}(n)$ for constant d, whereas we use $\log(k) \cdot \text{polylog}(\log(n\Delta))$ amortized update time. Thus, Theorem 1.1 is the first result to show that (k, z)-clustering can be performed in the insertion-only streaming model with amortized update time sublinear in the number k of clusters.

In general, it is known that $\Omega(nk)$ runtime is necessary for even a constant-factor approximation to the metric (k, z)-clustering problem [BCLP23], which would imply an $\Omega(k)$ lower bound for update time amortized over the stream. However, inputs to data streams require bounded bit precision and thus are intrinsically more appropriate for Euclidean (k, z)-clustering, where it was previously unclear whether the $\Omega(nk)$ total runtime lower bounds held. Our result in Theorem 1.1 shows that they do not.

| Streaming algorithm | Amortized Update Time | |
|-------------------------|---|--|
| [HK20] | $k^2 \cdot \operatorname{polylog}(n\Delta)$ | |
| [BCLP23, BCF24] | $k \cdot \operatorname{polylog}(n\Delta)$ | |
| Theorem 1.1 (this work) | $\log(k) \cdot \operatorname{polylog}(\log(n\Delta))$ | |

Fig. 2: Table of (k, z)-clustering algorithms on data streams, omitting linear dependencies in the dimension d. We remark that [HK20, BCLP23, BCF24] can handle the fully-dynamic setting, whereas ours cannot. However, our algorithm uses sublinear space while theirs does not.

Additionally, Theorem 1.1 is the first result to achieve (k, z)-clustering on insertion-only streams using $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space, i.e., space usage independent of n. Despite a long line of work on clustering in the streaming model [HM04, HK07, Che09, FL11, BFLR19, BBC⁺19, WZZ23], the majority of previous results achieved space overheads that were polylogarithmic in n, compared to the offline setting. In a recent breakthrough work, [CWZ23] gave a streaming algorithm for (k, z)clustering that uses $o_{k,d,\varepsilon}(\log(n\Delta))$ words of space. However, their algorithm requires exponential time to process each stream item. Hence, the question remains whether we can achieve space complexity independent of n altogether or even if we can achieve efficient update time with space overhead that is even polylogarithmic in n. Theorem 1.1 shows that we can simultaneously achieve both, i.e., space complexity (in words) entirely independent of n, while also achieving $\log(k) \cdot \operatorname{polylog}(\log(n\Delta))$ amortized update time. In particular, our results match the best known offline coreset constructions [CSS21, CLSS22] in terms of dependencies on d, k, and $\frac{1}{\varepsilon}$ in the space, and thus match or improve upon the space usage of all previous streaming algorithms for insertion-only streams across all factors. See Theorem 3.22 for a more formal statement and Figure 3 and Figure 2 for a comprehensive summary for the time and space complexity of existing results in the streaming model.

| Streaming algorithm | Words of Memory |
|---|---|
| [HK07], $z \in \{1, 2\}$ | $\tilde{\mathcal{O}}\left(\frac{dk^{1+z}}{\varepsilon^{\mathcal{O}(d)}}\log^{d+z}n\right)$ |
| [HM04], $z \in \{1, 2\}$ | $\tilde{\mathcal{O}}\left(\frac{dk}{\varepsilon^d}\log^{2d+2}n\right)$ |
| [Che09], $z \in \{1, 2\}$ | $\tilde{\mathcal{O}}\left(\frac{d^2k^2}{\varepsilon^2}\log^8 n\right)$ |
| [FL11], $z \in \{1, 2\}$ | $\tilde{\mathcal{O}}\left(\frac{d^2k}{\varepsilon^{2z}}\log^{1+2z}n\right)$ |
| Sensitivity and rejection sampling [BFLR19] | $	ilde{\mathcal{O}}\left(rac{d^2k^2}{arepsilon^2}\log n ight)$ |
| Online sensitivity sampling | $\tilde{\mathcal{O}}\left(\frac{d^2k^2}{\varepsilon^2}\log^2 n\right)$ |
| Merge-and-reduce with coreset of [CLSS22] | $\tilde{\mathcal{O}}\left(rac{dk}{\min(arepsilon^4,arepsilon^{2+z})}\log^4n ight)$ |
| [CWZ23] | $\tilde{\mathcal{O}}\left(\frac{dk}{\min(\varepsilon^4,\varepsilon^{2+z})}\right) \cdot \operatorname{polylog}(\log n)$ |
| Theorem 1.1 (this work) | $	ilde{\mathcal{O}}\left(rac{dk}{\min(arepsilon^4,arepsilon^{2+z})} ight)$ |

Fig. 3: Table of (k, z)-clustering algorithms on insertion-only streams. We summarize existing results with $z = \mathcal{O}(1)$, $\Delta = \text{poly}(n)$, and the assumption that $k > \frac{1}{\varepsilon^z}$ for the purpose of presentation.

Moreover, we note that Theorem 1.1 also implies that we can efficiently compute the approximately optimal centers for (k, z)-clustering in the incremental setting as well. In particular, given an insertion-only stream of n points that defines a dataset X on $[\Delta]^d$, our one-pass streaming algorithm uses $d \log(k) \cdot \operatorname{polylog}(\log(n\Delta))$ amortized update time and $\tilde{\mathcal{O}}(dk \log(n\Delta))$ bits of space, and outputs an $\mathcal{O}(z)$ -approximation to (k, z)-clustering at all times in the stream. We show this in Theorem 3.23. Additionally, Theorem 1.1 surprisingly even gives the best known runtime for (k, z)-clustering in the offline setting.

Theorem 1.2. Given an dataset X of n in $[\Delta]^d$, there is an algorithm that uses $nd \log(k) \cdot polylog(log(n\Delta))$ runtime and outputs an $\mathcal{O}(z)$ -approximation to the (k, z)-clustering problem.

Previously, the best existing (k, z)-clustering algorithms use $(nd+nk) \cdot \text{polylog}(n\Delta)$ runtime, by using bicriteria approximations [AV07] in conjunction with dimensionality reduction [CEM⁺15, BBC⁺19, MMR19, ISZ21] and importance sampling [HV20, CSS21, CLSS22] to construct a coreset, which is

then given as input to any polynomial-time approximation algorithm such as local search [GT08]. By comparison, Theorem 1.2 has exponentially better dependencies on both k and $\log(n\Delta)$.

In concurrent and independent works, techniques by [DSS24, ITHS24] can be used to achieve offline algorithms for (k, z)-clustering that use $\mathcal{O}(nd \log(n\Delta k))$ runtime, while [ITS24] achieved an algorithm that uses $\mathcal{O}(n^{1+\Omega(1)}d\log(n\Delta k))$ runtime, for the setting of $\Delta = \text{poly}(n)$. In the same spirit as Theorem 1.2, these works observed that the $\Omega(nk)$ lower bound of [BCLP23] is not necessarily for Euclidean clustering. However, our result is faster across all settings.

Finally, we remark on the implications of our techniques to communication complexity. Our approach in Theorem 1.1 uses a black-box reduction that utilizes an efficient encoding for any coreset construction. For instance, if $k < \frac{1}{\varepsilon^2}$, it may be desirable to use existing coreset constructions of size $\tilde{\mathcal{O}}\left(\frac{k^2}{\varepsilon^2}\right)$ rather than $\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^{z+2}}\right)$; our algorithm transitions to this case smoothly. In fact, our algorithm actually uses $\tilde{\mathcal{O}}\left(dk\log(n\Delta)\right) + \frac{dk}{\min(\varepsilon^4,\varepsilon^{2+z})} \cdot \operatorname{polylog}\left(k,\frac{1}{\varepsilon},\log(n\Delta)\right)$ bits of space, i.e., the $\frac{1}{\varepsilon}$ factors do not multiply the $\log(n\Delta)$ factors. Therefore, our results are near-optimal with a recent lower bound on the communication complexity of clustering [ZTHH24].

Subspace embeddings. We next show the applications of our technique in Figure 1 to the problem of L_p subspace embeddings. In addition to structural results about efficient encodings for L_p subspace embeddings, our main result is the following:

Theorem 1.3. Given a matrix \mathbf{A} of n rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$, with M = poly(n)and online condition number κ , there exists a one-pass streaming algorithm in the row arrival model that outputs a $(1 + \varepsilon)$ -strong coreset of \mathbf{A} and amortized runtime is $\mathcal{O}(d)$ per update. For $p \in [1, 2]$, the algorithm uses $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right)$ words of space, while for p > 2, the algorithm uses $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right)$ words of space.

We first emphasize that the amortized runtime of Theorem 1.3 translates to an offline algorithm with $\mathcal{O}(nd)$ runtime. Since any offline algorithm requires $\mathcal{O}(nd)$ time to read the input, without additional assumptions, our result is tight and shows that there is no inherent time complexity separation between the offline setting and the streaming model for L_p subspace embeddings.

We next remark on the differing behaviors across the ranges of p. This is not a coincidence, as [LWW21] showed that any constant-factor L_p subspace embedding for $p \ge 2$ requires $\Omega(d^{\max(p/2,1)}+1)$ bits. Thus, our result has tight dependencies in terms of the dimension d.

Moreover, existing algorithms based on using merge-and-reduce with the optimal coreset constructions for L_p subspace embeddings require poly $(d, \log n)$ bits of space, e.g., $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right) \cdot \text{polylog}(n)$ for $p \leq 2$ [DMM06a, DMM06b, Sar06] and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right) \cdot \text{polylog}(n)$ for p > 2 [CP15, WY23]. Similarly, schemes based on online sampling also require poly $(d, \log n)$ bits of space, e.g., $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right) \cdot \text{polylog}(n)$ for $p \leq 2$ [CMP20] and $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right) \cdot \text{polylog}(n)$ for p > 2 [CP15, BDM⁺20, WY23]. Hence, it was unknown whether there is an inherent separation between the offline setting and the streaming model for coreset constructions for L_p subspace embeddings. Theorem 1.3 resolves this question, showing that there is no overhead for the streaming model – we can perform L_p subspace embedding in $\mathcal{O}_{d,\varepsilon}(1)$ bits of space. See Figure 4 and Theorem 4.19 for more details.

| Streaming algorithm | Words of Memory | |
|--|--|--|
| Merge-and-reduce with coreset of [DMM06a, DMM06b, Sar06], $p \leq 2$ | $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right) \cdot \operatorname{polylog}(n)$ | |
| Online leverage score sampling [CMP20], $p = 2$ | $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\log n\right)$ | |
| Online sensitivity sampling [BDM ⁺ 20], $p \leq 2$ | $	ilde{\mathcal{O}}\left(rac{d^3}{arepsilon^2}\log n ight)$ | |
| Online Lewis weight sampling [WY23], $p \leq 2$ | $\tilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\log n\right)$ | |
| Theorem 1.1 (this work), $p \leq 2$ | $	ilde{\mathcal{O}}\left(rac{d^2}{arepsilon^2} ight)$ | |
| Merge-and-reduce with coreset of [DDH+09], $p \ge 1$ | $\widetilde{\mathcal{O}}\left(\frac{d^{\max(p/2+1,p)+2}}{\varepsilon^2}\log^3 n\right)$ | |
| Merge-and-reduce with coreset of [CP15], $p \ge 2$ | $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\operatorname{poly}(\varepsilon)}\right) \cdot \operatorname{polylog}(n)$ | |
| Online Lewis weight sampling [WY23], $p > 2$ | $\tilde{\mathcal{O}}\left(\frac{d^{p/2+1}}{\varepsilon^2}\right) \cdot \operatorname{polylog}(n)$ | |
| Theorem 1.1 (this work), $p > 2$ | $	ilde{\mathcal{O}}\left(rac{d^{p/2+1}}{arepsilon^2} ight)$ | |

Fig. 4: Table of L_p subspace embedding algorithms on insertion-only streams. We summarize existing results with $\kappa = \text{poly}(n)$ for the purpose of presentation.

1.2 Technical Overview

In this section, we provide a high-level intuition behind our framework and how to efficiently implement each step in Figure 1. Recall that given a stream S of length n, we first use a crude sampling scheme to generate a stream S' of length $n^{1-\Omega(1)}$. Then, we use a refined sampling scheme on S', which provides a significantly more efficient compression of the input, but requires longer time to compute, resulting in a stream S''. However, because these operations are performed on the input stream S' of length o(n), then these slower operations can be amortized into lower-order terms that do not multiply a linear term in n. Finally, we run merge-and-reduce on S'' to produce a coreset C, which we store using an efficient encoding to optimize the final space complexity.

1.2.1 Fast Update Time for Clustering

For any point x in a fixed dataset X, the (k, z)-clustering sensitivity of x is defined by $s(x) = \max_{C \subset \mathbb{R}^d: |C| \leq k} \frac{\operatorname{cost}(x,C)}{\operatorname{cost}(X,C)}$, where the cost function is a sum of the z-th power of the distances [FL11, FS12, BFL⁺21, CWZ23, WZZ23]. When the dataset is evolving, the online sensitivity of x_t is the (k, z)-clustering sensitivity of x with respect to $X_t = \{x_1, \ldots, x_t\}$, so that it quantifies the "importance" of x_t with respect to the points of the stream that have already arrived. A standard approach in streaming algorithms for (k, z)-clustering is to sample each point x with probability proportional to its online sensitivity, resulting in poly $\left(k, \frac{1}{\varepsilon}, \log(n\Delta)\right)$ samples.

To efficiently implement Figure 1, we first compute a crude but fast approximation to the sensitivity of each point. Note that since the total sensitivity sums to $\mathcal{O}(k)$, then even n^{α} -approximation to the sensitivities, where $\alpha \in (0,1)$, will sample $n^{\alpha} \cdot \text{poly}\left(k, \frac{1}{\varepsilon}, \log(n\Delta)\right) = o(n)$ points, forming the new insertion-only stream \mathcal{S}' of length o(n). Unfortunately, it is not clear how

to efficiently acquire even crude approximations to the online sensitivities; this is the main runtime bottleneck in achieving o(k) amortized update time.

(k, z)-clustering sensitivity to (k, z)-medoids sensitivity. To that end, we first define the (k, z)-medoids sensitivity of x with respect to the dataset X for (k, z)-clustering sensitivity by $\tau(x) = \max_{C \subset X: |C| \le k} \frac{\cot(x, C)}{\cot(X, C)}$; we define the online sensitivities analogously. Note that for the medoids formulation, the k centers must be among the input set X, which for our purposes will be the coreset maintained by the algorithm, rather than the original input. Hence, we can assume $|X| = \text{poly}\left(k, d, \log(n\Delta), \frac{1}{\varepsilon}\right)$. We show that the (k, z)-medoids sensitivity $\tau(x)$ is a constant-factor approximation to the (k, z)-clustering sensitivity s(x) as follows.

First, note that the maximization of the ratio $\frac{\cot(x,C)}{\cot(X,C)}$ is over a smaller search space for the possible values of C in the medoids formulation and thus $\tau(x) \leq s(x)$. To show $s(x) \leq \tau(x)$, we first recall that the optimal (k, z)-clustering and (k, z)-medoids clustering costs are within a factor of $2^{\mathcal{O}(z)}$ due to the generalized triangle inequality. Hence if we fix C to be a set of centers that maximizes s(x), let c be the closest center of C to x and let R be the set of points of X served by c, then it suffices to show that there exists $c' \in X$ such that $\frac{\cot(x,c)}{\cot(R,c)} \approx \frac{\cot(x,c')}{\cot(R,c')}$, as we can find a $2^{\mathcal{O}(z)}$ -approximate clustering of $X \setminus R$ using k-1 other medoids.

For k-medoids, we perform casework on the size of the set P of points that are closer to x than to c. If |P| < 0.99|R|, then by a simple Markov-type argument, we can show there exists some $y \in R \setminus P$ such that $\cot(y, c) \leq \frac{100}{r} \cdot \cot(R, c)$, but since $y \notin P$, then y is closer to c than y is to x. Thus by setting y to be a k-medoids center, it follows from the triangle inequality that the cost of clustering R with y instead of c cannot change by much. Moreover, the distance between x and y is similar to the distance between x and c, so that $\tau(x) \approx s(x)$. On the other hand, if $|P| \geq 0.99|R|$, then there is a large number of points which are closer to x than to c by definition of P and hence $\frac{\cot(x,c)}{\cot(R,c)} = \mathcal{O}(1)|R|$. However, we can choose the point c' of P farthest away from x, so that $\frac{\cot(x,c')}{\cot(R,c')} \geq \frac{1}{2|R|}$ and thus $\tau(x) \approx s(x)$ again.

Constrained (k, z)-medoids clustering with a fixed center. We next show how to approximately solve the constrained medoids clustering problem min cost(X, C) across all sets $C \subset X$ of k centers containing a center at p. Suppose C is the optimal such constrained clustering. Let S be a constant-factor approximation to the optimal unconstrained (k, z)-medoids clustering on X, which can be efficiently computed by local search [GT08]. We show that the best clustering obtainable from swapping a center in S with $\{p\}$ is a good approximation to C.

Specifically, let W be the set of k-1 centers $W \subseteq S$ such that $W \cup \{p\}$ has the minimum (k, z)medoids clustering cost on X, and let $C' = W \cup \{p\}$. Let Q be the set of at most k centers of $S \cup \{p\}$ consisting of the nearest neighbors to C. For each $y \in X$, let $\pi_C(y)$ be the center in C closest to yThen we can use the triangle inequality to charge $\operatorname{cost}(y, Q)$ in terms of $\operatorname{cost}(y, \pi_C(y)) + \operatorname{cost}(\pi_C(y), Q)$.
Since $\operatorname{cost}(\pi_C(y), Q) \leq \operatorname{cost}(\pi_C(y), S \cup \{p\}) \leq \operatorname{cost}(\pi_C(y), S)$, we can use the triangle inequality to
charge $\operatorname{cost}(\pi_C(y), Q)$ to $\operatorname{cost}(y, C) + \operatorname{cost}(y, S)$. Finally, because C' is the k centers among $S \cup \{p\}$,
summing across all $y \in X$ we can upper bound $\operatorname{cost}(X, C') \leq \operatorname{cost}(X, Q)$ in terms of $\operatorname{cost}(X, C)$ and $\operatorname{cost}(X, S)$ and hence, simply in terms of $\operatorname{cost}(X, C)$.

However, finding C' requires finding W, the best set of k-1 centers $W \subseteq S$, which does not immediately seem like an easy algorithmic task. To that end, we approximate cost(X, C') by first performing the following bookkeeping on S. For each center $s \in S$, we compute the number n_s (or weight) of the points of X served by s. We then compute the distance r_s from s to its closest center in $S \setminus \{s\}$ and show the clustering cost after removing s would approximately increase by $n_s \cdot (r_s)^z$.

Next, we need to update this information for each query p that we guess to serve x in the optimal constrained solution. Rather than updating the information for each center $s \in S$, we instead show that it suffices to update the information for just the nearest neighbor $u \in S$ to p. We then remove the center s with the smallest $n_s \cdot (r_s)^z$ to find W and thus C'. Finally, we show that $\operatorname{cost}(X, S) + n_c \cdot (r_c)^z$ is both efficiently computable given our pre-processing information and also a good approximation to $\operatorname{cost}(X, C')$.

Constrained (k, z)-medoids clustering with a fixed closest/serving center. Although C' is a set of k centers that contains our guess p for the center that realizes the sensitivity of x, it may not hold that p is the closest center in C' to x. Thus we must further utilize C' to approximately solve the constrained clustering problem min $\cot(X, C)$ across all sets $C \subset X$ of k centers containing a center at p and no other centers that are closer to x than p. Let C be the optimal solution to this constrained problem and let $r := ||x - p||_2$. Our first observation is that any points served by some center $c \in C'$ in the ball $B_{r/2}(x)$ of radius $\frac{r}{2}$ around x must be served in C by a center outside the interior of the ball $B_r(x)$ of radius r around x. In fact, if n_b is the number (or weight) of points served by centers of C' in $B_{r/2}(x)$, then we would expect these points to incur additional $\cot n_b \cdot r^z$ in $\cot(X, C)$.

Unfortunately, it is not true that $cost(X, C') + n_b \cdot r^z$ is a good approximation to cost(X, C) because there may be a large number of points served by a single center q at some distance $d_q \in [\frac{r}{2}, r)$ from x. However, since we did not include these points in n_b , then they are effectively not moved to outside of $B_r(x)$. Hence, it is possible that cost(X, C) is significantly larger than $cost(X, C') + n_b \cdot r^z$.

The natural approach would be to include these points in the computation of n_b , so that perhaps we instead compute n'_b to be the number of points served by centers of C' inside $B_r(x)$ and then effectively move them to a center $B_r(x)$ via $cost(X, C') + n'_b \cdot r^z$. This still does not work because there can be a large number of points served by a center q inside but arbitrarily close to the boundary of $B_r(x)$; in this case, $cost(X, C') + n'_b \cdot r^z$ is a significant over-estimate of cost(X, C).

The crucial observation is that we do not need to handle either of these cases because in both of these cases, there exists a different point p' in the annulus between $B_r(x)$ and $B_{r/2}(x)$ that approximately realizes the (k, z)-medoids sensitivity of x, such that these bad cases do not hold for p'. In particular, these problematic centers would be outside the ball of radius $||p' - x||_2$ centered at x and so the resulting clustering is valid for the constraint of p' serving x.

On the other hand, we must still ensure our approximation $\cot(X, C') + n'_b \cdot r^z$ does not significantly overestimate the sensitivity. Thus if n_a is number of points served by centers in the annulus between $B_r(x)$ and $B_{r/2}(x)$, we show that if $n_b \ge n_a$, then $\frac{r^z}{\cot(X,C')}$ is a good estimate to $\frac{\cot(x,C)}{\cot(X,C)}$. Otherwise if $n_b < n_a$, then we show that $\frac{r^z}{\cot(X,C')}$ is an overestimate of $\frac{\cot(x,C)}{\cot(X,C)}$ but still upper bounded by the desired (k, z)-medoids sensitivity $\tau(x)$, up to a constant factor.

Crude quadtree for crude approximations. Finally, to implement our crude sampling procedure in Figure 1, we create a quadtree for the input set, e.g. [IT03, BIRW16, CLN⁺20]. Traditionally, each cell of a level of a quadtree has half the side length of a cell of an adjacent level in the quadtree. However, because we only seek n^{α} -approximation, we permit each cell of a level to have side length $\frac{1}{n^{\alpha}}$ fraction of the side length of the adjacent level. This significantly decreases the number of levels in the quadtree to a constant number of levels overall, provided that $\Delta = \text{poly}(n)$, rather than $\mathcal{O}(\log \Delta)$ levels in the standard quadtree. We then run the same algorithm above searching up the levels of the quadtree rather than the radially outward ball.

Specifically, we estimate the distance of the center serving the point x whose sensitivity we wish to approximate. Each estimate of a possible distance corresponds to a separate level in the quadtree. For a fixed level β in the quadtree, we first take a constant-factor approximation to the unconstrained (k, z)-clustering problem to find a crude-approximation to the optimal (k, z)-clustering constrained to having a center in the same cell as x at level β . Our algorithm provides an estimate Ψ of this cost by deleting the center that increases the overall cost the least, among the centers in the constant-factor solution adjoined with a center in the same cell as x at level β . We then ensure that x is served by a center at level β by moving all centers in the same cell as x before level β "upward" until level β by computing the number (or weight) n_{β} of points served by these centers in Ψ and again increasing Ψ by $n_{\beta} \cdot r^{z}$, where r is the distance estimated by the quadtree for a center serving x at level β .

1.2.2 Optimal Space Clustering in the Streaming Model

The two most common approaches for (k, z)-clustering on insertion-only streams are merge-andreduce and online sensitivity sampling. Informally, merge-and-reduce partitions the stream into consecutive blocks, builds a binary tree on the sequence of blocks, and then maintains a coreset for the points corresponding to each node in the tree. On the other hand, the online sensitivity approach samples each point x_t of the stream with probability proportional to its online sensitivity, a quantity that measures the importance of x_t with respect to the points of the stream that have already arrived. However, it is known these approaches cannot immediately be used to achieve our goal of streaming algorithms with space (in words) independent of n. we describe these approaches and others, as well as their shortcomings in more detail in Section 2.1.

Recent insight by [CWZ23] observed that merge-and-reduce requires space that is polylogarithmic in the length of the input stream, while running online sensitivity sampling induces an insertion-only stream \mathcal{S}' of length poly $\left(k, d, \log(n\Delta), \frac{1}{\varepsilon}\right)$ of weighted points that forms a $(1 + \varepsilon)$ -coreset of the input points. By running merge-and-reduce on \mathcal{S}' , we acquire a $(1 + \mathcal{O}(\varepsilon))$ -coreset for the original stream using an algorithm that stores $\operatorname{poly}_{k,d,\varepsilon}(\log \log(n\Delta))$ weighted points.

Efficient encoding of coreset points. To achieve our goal of $\mathcal{O}_{k,d,\varepsilon}(1)$ space (in words), we thus need a more efficient encoding of each point, rather than representing each point using $\mathcal{O}(d\log(n\Delta))$ bits of space. Suppose we have a set C' of $\mathcal{O}(k)$ centers that is a constant-factor approximation to the optimal (k, z)-clustering on X. We can noisily encode X by writing each $x \in X$ as $x = \pi_{C'}(x) + (x - \pi_{C'}(x))$, where $\pi_{C'}(x)$ is the closest center of C' to x. We then round each coordinate of $x - \pi_{C'}(x)$ by rounding each coordinate to a power of $(1 + \varepsilon')$ for $\varepsilon' = \text{poly}\left(\varepsilon, \frac{1}{d}, \frac{1}{\log(n\Delta)}\right)$ to form a vector y'. Hence, given C', each vector $x' = \pi_{C'}(x) + y'$ can be encoded using $\mathcal{O}\left(\log k + d\log \frac{1}{\varepsilon'}\right)$, since we can store the *exponents* of the offsets. It is known [CWZ23, ZTHH24] that if X' is the set of all points of X rounded in this manner, then X' is a $(1 + \varepsilon)$ -coreset for X. Moreover, we can round the weights of X' to powers of $(1 + \varepsilon')$, thereby efficiently encoding each coreset in the merge-and-reduce framework to improve the overall space complexity of the problem.

Unfortunately, storing C' itself uses $\mathcal{O}(kd \log n)$ bits and the merge-and-reduce tree on \mathcal{S}' has height $\mathcal{O}\left(\log_{k,d,\varepsilon}(|S|)\right) = \operatorname{polylog}_{k,d,\varepsilon}(\log(n\Delta))$ and therefore requires at least the same number of

coresets to be simultaneous stored. Thus the total space remains $\mathcal{O}(kd\log n) \cdot \operatorname{polylog}_{k,d,\varepsilon}(\log(n\Delta))$.

Global encoding. To overcome this issue, we provide a single constant-factor approximation to the global dataset, rather than providing a constant-factor approximation for each level of the merge-and-reduce tree. Namely, we observe that whenever we need to do a merge operation at some time t, we use the stored coreset to recompute a constant-factor approximation C' to X_t . For each coreset for a subset S_v of data points representing a node v in the tree, we can then perform the efficient encoding with respect to C' instead of a separate constant-factor approximation. Although each encoding no longer guarantees $(1 + \varepsilon)$ -multiplicative approximation to $\cot(S_v, C)$ for a query set C of k centers, it still guarantees $\varepsilon' \cdot (X_t, C)$ additive error to $\cot(S_v, C)$, which suffices for a $(1 + \varepsilon)$ -approximation to $\cot(X_t, C)$ after summing across all subsets S_v at time t.

1.2.3 Subspace Embeddings

Finally, we briefly describe how our framework in Figure 1 can achieve fast space-efficient algorithms for subspace embeddings. However, it is not immediately obvious how to utilize a constant-factor subspace embedding $\mathbf{M} \in \mathbb{R}^{m \times d}$ of the input matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ to achieve an efficient encoding, especially since the dimensions are different. One possible approach is to project \mathbf{A} onto \mathbf{M} and round the resulting rows, but the rounding process could result in large error to due cancellation of rounded entries. For example, suppose that after the projection, there is a row $\mathbf{r} = (A, B)$ for some large values of A, B > 0. By rounding each coordinate of \mathbf{r} to the nearest power of $(1 + \varepsilon')$ for some $\varepsilon' = \text{poly}\left(\frac{\varepsilon}{d,\log(nM)}\right)$, we obtain a row $\mathbf{r}' = (A', B')$. Now for $\mathbf{v} = (B, -A)$, we have $\langle \mathbf{v}, \mathbf{r} \rangle = 0$, but $\langle \mathbf{v}, \mathbf{r}' \rangle$ can be as large as $\varepsilon' \cdot (|A| + |B|)$. Moreover, this error can compound across all rows of the matrix \mathbf{A} so that the resulting additive error to the estimate $\|\mathbf{Av}\|_p^p$ could be as large as $\varepsilon'^p \cdot \|\mathbf{A}\|_p^p$.

We instead observe that the correct procedure is to first multiply the matrix \mathbf{A} with a preconditioner so that none of its rows can contribute a large amount to the error. To that end, we first use a well-conditioned basis to compute a preconditioner $\mathbf{P} \in \mathbb{R}^{d \times d}$ and multiply each row of \mathbf{A} , so that for all vectors $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{A}\mathbf{x}\|_p = 1$, we have that $\frac{1}{\text{poly}(d)} \leq \|\mathbf{A}\mathbf{P}\mathbf{x}\|_p \leq \text{poly}(d)$. In particular, the maximum possible additive error $\langle \mathbf{v}, \mathbf{r}' \rangle$ for each row \mathbf{r} can be charged to the sensitivity of \mathbf{r} , and it follows that the sum of the errors due to the rounding can be at most $\varepsilon' \cdot \text{poly}(d)$. Since $\varepsilon' = \text{poly}\left(\frac{\varepsilon}{d,\log(nM)}\right)$, then the overall error is $\mathcal{O}(\varepsilon)$, which achieves a subspace embedding because $\|\mathbf{A}\mathbf{x}\|_p = 1$. For a row $\mathbf{b}_t = \mathbf{a}_t \mathbf{P}$, we then round each entry of \mathbf{b}_t to the nearest power of $(1 + \varepsilon')$ and store the exponent as before. We can then also achieve a streaming algorithm by compressing the entire merge-and-reduce tree, as before.

Fast runtime. Finally, we implement the approach in Figure 1 to achieve fast runtime. We first produce crude but fast n^{α} -approximations to the L_p sensitivities using the *root* leverage scores. These can be quickly produced using a quadratic form of the constant-factor approximation that we maintain at each time. In particular, given a constant-factor subspace embedding **B** to **A**, the leverage score of a row \mathbf{a}_i with respect to **A** is a constant factor multiple of $\mathbf{a}_i^{\top} (\mathbf{B}^{\top} \mathbf{B})^{-1} \mathbf{a}_i$. Now we can approximate this quantity using $\|\mathbf{gZa}_i\|_2^2$, where **g** is a random Gaussian vector and $\mathbf{Z} = (\mathbf{B}^{\top} \mathbf{B})^{-1/2}$. To relate this quantity to the L_p sensitivities, we show that the square root of the leverage score is within poly(n) factors of the true L_p sensitivities. Thus by sampling each row with the crude approximations, we then induce a stream of length o(n) for which we can amortize existing approaches that use poly(d) update time. For more details, see the intuition in Section 4.

1.3 Preliminaries

Given an integer n > 0, we use the notation [n] to represent the set $\{1, \ldots, n\}$. We use poly(n) to represent a fixed polynomial in n and polylog(n) to represent $poly(\log n)$. We say an event occurs with high probability if it occurs with probability at least $1 - \frac{1}{poly(n)}$.

In this paper, we will focus on the Euclidean clustering, as opposed to inputs from a general metric space. As is often standard in the streaming literature, we assume $\Delta = \text{poly}(n)$ and thus we allow each word of space to use $\Theta(\log(nd\Delta))$ bits of storage, so that we can store the weight and each coordinate of each point using $\mathcal{O}(1)$ words of space. Thus for vectors $x, y \in \mathbb{R}^d$, we use dist(x, y) to represent the Euclidean distance $||x - y||_2$, so that $||x - y||_2^2 = \sum_{i=1}^d (x_i - y_i)^2$. More generally, given a point x and a set S, we abuse notation by representing $\text{dist}(x, S) := \min_{s \in S} \text{dist}(x, s)$. We also recall the L_z norm of x is defined by $||x||_z$, where $||x||_z^2 = \sum_{i=1}^d x_i^z$. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, we use $||\mathbf{A}||_F$ to denote the Frobenius norm, so that

$$\|\mathbf{A}\|_{F}^{2} = \sum_{i \in [n]} j \in [d] A_{i,j}^{2}$$

For a fixed $z \ge 1$ and sets $X, C \subset \mathbb{R}^d$ with $X = \{x_1, \ldots, x_n\}$ we use $\operatorname{cost}(X, C)$ to denote $\sum_{i=1}^n \operatorname{dist}(x_i, C)^z$.

We first recall the generalized triangle inequality:

Fact 1.4 (Generalized triangle inequality). For any $z \ge 1$ and $x, y, z \in \mathbb{R}^d$, we have

$$dist(x,y)^{z} \le 2^{z-1}(dist(x,w)^{z} + dist(w,y)^{z}).$$

Next, we recall the definition of a strong coreset for (k, z)-clustering.

Definition 1.5 (Coreset). Given an approximation parameter $\varepsilon > 0$, and a set X of points $x_1, \ldots, x_n \in \mathbb{R}^d$ with distance function dist, a coreset for (k, z) clustering is a set S with weight function w such that for any set C of k points, we have

$$(1-\varepsilon)\sum_{t=1}^n dist(x_t,C)^z \le \sum_{q\in S} w(q)dist(q,S)^z \le (1+\varepsilon)\sum_{t=1}^n dist(x_t,C)^z.$$

We use the following coreset construction for (k, z)-clustering:

Theorem 1.6. [CLSS22, HLW23, CLS⁺22] Given an accuracy parameter $\varepsilon \in (0, 1)$, there exists a coreset construction for Euclidean (k, z)-clustering that samples $\tilde{\mathcal{O}}\left(\min\left(\frac{1}{\varepsilon^2} \cdot k^{2-\frac{z}{z+2}}, \frac{1}{\min(\varepsilon^4, \varepsilon^{2+z})} \cdot k\right)\right)$ weighted points of the input dataset.

We next recall the standard Johnson-Lindenstrauss transformation:

Theorem 1.7 (Johnson-Lindenstrauss lemma). [JL84] Let $\varepsilon \in (0, \frac{1}{2})$ and $m = \mathcal{O}\left(\frac{1}{\varepsilon^2} \log n\right)$. Let $X \subset \mathbb{R}^d$ be a set of n points. There exists a family of random linear maps $\Pi : \mathbb{R}^d \to \mathbb{R}^m$ such that with high probability over the choice of $\pi \sim \Pi$,

$$(1-\varepsilon)\|x-y\|_{2} \le \|\pi x - \pi y\|_{2} \le (1+\varepsilon)\|x-y\|_{2},$$

for all $x, y \in X$.

We next recall the following standard concentration inequality:

Theorem 1.8 (Hoeffding's inequality). Let X_1, \dots, X_n be independent random variables such that $a_i \leq X_i \leq b_i$, and let $S_n = \sum_{i=1}^n X_i$. Then

$$\mathbf{Pr}\left[\left|S_n - \mathbb{E}\left[S_n\right]\right| > t\right] \le 2\exp\left(-\frac{t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

2 Clustering

In this section, we describe the efficient encoding for coresets for (k, z)-clustering, as well as a global encoding that be utilized to achieve a one-pass streaming algorithm on insertion-only streams for (k, z)-clustering using $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space. We begin with a number of preliminaries.

We first formally define the sensitivity and online sensitivity of each point x in a dataset X for (k, z)-clustering.

Definition 2.1 (Sensitivities for (k, z)-clustering). The sensitivity of a point $x \in X$ for (k, z)clustering in a metric space equipped with metric dist is

$$\max_{C:|C| \le k} \frac{\cot(x,C)}{\cot(X,C)} = \max_{C:|C| \le k} \frac{\operatorname{dist}(x,C)^z}{\sum_{x \in X} \operatorname{dist}(x,C)^z}.$$

Definition 2.2 (Online sensitivity for (k, z)-clustering). Let x_1, \ldots, x_n be a sequence of points with metric dist and define $X_t := \{x_1, \ldots, x_t\}$ for all $t \in [n]$. The online sensitivity of x_t for (k, z)-clustering is

$$\max_{C:|C|\leq k} \frac{\operatorname{cost}(x_t, C)}{\operatorname{cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\operatorname{dist}(x_t, C)^z}{\sum_{i=1}^t \operatorname{dist}(x_i, C)^z}.$$

We have the following upper bound on the sum of the online sensitivities, i.e., the total online sensitivity.

Theorem 2.3. [CWZ23] Let $X = \{x_1, \ldots, x_n\} \subset [\Delta]^d$ be a sequence of n points and let $\sigma(x_t)$ denote the online sensitivity of x_t for $t \in [n]$ for (k, z)-clustering, where $z \ge 1$. Then

$$\sum_{t=1}^{n} \sigma(x_t) = \mathcal{O}\left(2^{2z}k\log^2(nd\Delta)\right)$$

We next recall the guarantees of online sensitivity sampling, in terms of both correctness and sample complexity.

Theorem 2.4 (Online sensitivity sampling). [CWZ23] Given a sequence x_1, \ldots, x_n of points, suppose each point x_t is sampled with probability $p_t \ge \min(1, \gamma \cdot \sigma(x_t))$, where σ_t is the online sensitivity of x_t and $\gamma = \mathcal{O}\left(\frac{dk}{\varepsilon^2}\log\frac{n\Delta}{\varepsilon}\right)$ and weighted $\frac{1}{p_t}$ if x_t is sampled. Then with high probability, the weighted sample is a $(1 + \varepsilon)$ -strong coreset for (k, z)-clustering that contains at most $\mathcal{O}\left(\frac{dk^2}{\varepsilon^2}\log^3\frac{n\Delta}{\varepsilon}\right)$ points.

2.1 Background for Optimal Space Clustering in the Streaming Model

We first describe common approaches for clustering in the streaming setting. Even though they incur logarithmic overheads in the size n of the dataset compared to offline coreset constructions, their intuition will nevertheless be helpful towards our main algorithm.

Merge-and-reduce. Merge-and-reduce [BS80, HM04] is a standard approach on insertion-only streams for (k, z)-clustering, as well as many other problems for which there exist coreset constructions. Given a dataset $X \subset [\Delta]^d$ of size n, the number k of clusters, an accuracy parameter ε , a failure probability δ , suppose there exists a coreset construction algorithm for (k, z)-clustering that samples and reweights $f(n, d, k, \varepsilon, \delta)$ points of X. The merge-and-reduce framework first partitions the stream into consecutive blocks of size $f(n, d, k, \varepsilon', \delta')$, where $\varepsilon' = \frac{\varepsilon}{\mathcal{O}(\log n)}$ and $\delta' = \frac{\delta}{\operatorname{poly}(n)}$. It then creates a coreset with accuracy $(1 + \varepsilon')$ and failure probability δ' for each block of the stream. Each of these coresets uses space $f(n, d, k, \varepsilon', \delta')$ and can be viewed as the leaves of a binary tree with height $\mathcal{O}(\log n)$. For each node in the tree at depth ℓ , the merge-and-reduce framework then takes the points that are in the coresets in its children nodes at depth $\ell + 1$ and constructs a coreset with accuracy $(1 + \varepsilon')$ for these points.

Observe that the coreset at the root of the tree is a coreset for the entire dataset of the stream. Moreover, each node is the coreset of the coresets of its children nodes, so the entire merge-and-reduce process can be performed on-the-fly during the evolution of an insertion-only stream. Although each level of the merge-and-reduce tree induces a multiplication distortion of $(1 + \varepsilon')$, the accuracy of the root node is $(1 + \varepsilon')^{\mathcal{O}(\log n)} = (1 + \varepsilon)$ due to the setting of $\varepsilon' = \frac{\varepsilon}{\mathcal{O}(\log n)}$. Unfortunately, the optimal coreset constructions sample $\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^2}\right) \cdot \min\left(k, \frac{1}{\min(\varepsilon^2, \varepsilon^z)}\right)$ points [CSS21, CLSS22] and in fact for certain regimes of ε , coreset constructions for (k, z)-clustering provably require $\Omega\left(\frac{k}{\varepsilon^{z+2}}\right)$ points [HLW23, CLS⁺22]. Therefore setting $\varepsilon' = \frac{\varepsilon}{\mathcal{O}(\log n)}$ in the merge-and-reduce approach would incur log *n* factors that are prohibitive for our goal.

Offline sensitivity sampling. Another common approach is to adapt offline coreset constructions to the streaming model. In particular, recent efforts [CWZ23, WZZ23] have been made to adapt the sensitivity sampling framework [FL11, FS12, BFL+21] to data streams. The *sensitivity* of each point x among a dataset $X \subset \mathbb{R}^d$ of n points measures the "importance" of the point x with respect to X, for the purposes of (k, z)-clustering and, as per Definition 2.1, is defined by

$$\max_{C \subset \mathbb{R}^d: |C| \le k} \frac{\operatorname{cost}(x, C)}{\operatorname{cost}(X, C)} = \max_{C \subset \mathbb{R}^d: |C| \le k} \frac{\operatorname{dist}(x, C)^z}{\sum_{x \in X} \operatorname{dist}(x, C)^z}.$$

There exist many variants of the sensitivity sampling framework, but the most relevant approach is sampling each point independently without replacement with probability proportional to its sensitivity, a process that can be shown to admit a coreset construction with high probability.

In fact, the argument is relatively straightforward. The analysis first fixes a set C of k centers and shows that sensitivity sampling preserves the cost of clustering X with C in expectation. It then upper bounds the variance of the cost of clustering C on the sampled points to show concentration. Although there can be arbitrary number of choices for C, for k centers among \mathbb{R}^d , it suffices to only show correctness-of-approximation on a net of size $\left(\frac{n}{\varepsilon}\right)^{\mathcal{O}(kd)}$, which can be handled by adjusting the probability of failure for each choice of C and then applying a union bound.

Unfortunately, the sensitivity of a point x is defined with respect to the entire dataset X, which is unknown at the time of arrival of x in a data stream, and thus the sensitivity of x cannot be computed or even well-approximated at that point.

Online sensitivity sampling. Instead, recent works have focused on *online* sensitivity sampling, where the elements of the dataset X are ordered by the time of their arrival in the data stream,

so that $X = \{x_1, \ldots, x_n\}$, where x_1 is the first item in the stream and x_n is the last item in the stream. The online sensitivity of a point x_t is then, as per Definition 2.2, defined by

$$\max_{C \subset \mathbb{R}^d: |C| \le k} \frac{\operatorname{cost}(x_t, C)}{\operatorname{cost}(X_t, C)} = \max_{C \subset \mathbb{R}^d: |C| \le k} \frac{\operatorname{dist}(x_t, C)^z}{\sum_{i=1}^t \operatorname{dist}(x_i, C)^z}$$

where X_t is the subset consisting of the first t points of X. Note that while the sensitivity of a point x_t measures the importance of x_t with respect to the entire dataset X, the online sensitivity of x_t measures the importance of x_t with respect to the prefix of size t, a quantity that can be computed at the time of arrival of x_t . In particular, since online sensitivity sampling produces a $(1 + \varepsilon)$ -coreset of X_{t-1} , then the previously sampled points can be used to approximate the online sensitivity of x_t .

The online sensitivity framework then samples each point with probability proportional to its online sensitivity. We remark that although this process seemingly induces dependencies in the analysis, a standard martingale and coupling argument shows correctness of online sensitivity sampling. Unfortunately, as a result of the sampling probability, the total number of points sampled is proportional to the sum of the online sensitivities of the points, which can be shown to be at least $\Omega(k \log(n\Delta))$. Thus online sensitivity sampling would again incur log *n* factors that are prohibitive for our goal.

2.2 Efficient Encoding for Coreset Construction for (k, z)-Clustering

We now give our efficient encoding for a given coreset for (k, z)-clustering. Given a dataset X, which can be viewed as either the original input or a set of weighted points that forms a coreset of some underlying dataset, we first acquire a constant-factor approximation C' for (k, z)-clustering on X. For each $x \in X$, let $\pi_{C'}(X)$ be the closest center of C' to x. We then write each $x \in X$ as $x = \pi_{C'}(x) + (x - \pi_{C'}(x))$, thereby decomposing x into the closest center $\pi_{C'}(x)$ and its offset $x - \pi_{C'}(x)$ from the center. We show that for the purposes of $(1 + \varepsilon)$ -approximation for (k, z)-clustering, we can afford to round each coordinate of $x - \pi_{C'}(x)$ to a power of $(1 + \varepsilon')$, where $\varepsilon' = \text{poly}\left(\varepsilon, \frac{1}{d}, \frac{1}{\log(n\Delta)}\right)$, forming a vector y'. Finally, we can store C' and thus encode the vector $x' = \pi_{C'}(x) + y'$ using $\mathcal{O}\left(\log k + d\log\left(\frac{1}{\varepsilon}, d, \log(n\Delta)\right)\right)$ bits, by storing the identity of $\pi_{C'}(x)$ and the exponent of the offset for each of the d coordinates. We give the algorithm in full in Algorithm 1.

To show correctness of our encoding, we first recall the following fact.

Fact 2.5 (Claim 5 in [SW18]). Suppose $z \ge 1$, $x, y \ge 0$, and $\varepsilon \in (0, 1]$. Then

$$(x+y)^z \le (1+\varepsilon) \cdot x^z + \left(1+\frac{2z}{\varepsilon}\right)^z \cdot y^z.$$

Given a constant-factor approximation C', we now show that the rounding X' of the dataset X by representing each point as a rounded offset from its closest center in C' is a strong coreset of X.

Lemma 2.6. Let $\varepsilon \in (0, \frac{1}{2})$ and let X' be the weighted dataset S defined by their offsets from the set C' of centers from Algorithm 1. Then for all $C \subset [\Delta]^d$ with $|C| \leq k$,

$$(1 - \varepsilon) \cdot \cot(C, X) \le \cot(C, X') \le (1 + \varepsilon) \cdot \cot(C, X).$$

Proof. Let C be any fixed set of at most k centers. We abuse notation so that for each $x \in X$, we use x' to denote the corresponding point c(x) + y, where y' = x - c'(x) is the offset, and y is y' with

Algorithm 1 Efficient Encoding for Coreset Construction for (k, z)-Clustering

- **Input:** Data set $X \subset [\Delta]^d$ with weight $w(\cdot)$, accuracy parameter $\varepsilon \in (0, 1)$, number of clusters k, parameter $z \ge 1$, failure probability $\delta \in (0, 1)$
- **Output:** $(1 + \varepsilon)$ -coreset for (k, z)-clustering
- 1: $\varepsilon' \leftarrow \frac{\operatorname{poly}(\varepsilon^z)}{\operatorname{poly}(k, \log(nd\Delta))}$
- 2: Find a set C' of k centers that is a constant-factor approximation to (k, z)-clustering on X
- 3: for each $x \in X$ do
- 4: Let c'(x) be the closest center of C' to x
- 5: Let y' be the offset x c'(x)
- 6: Let y be y' with coordinates rounded to a power of $(1 + \varepsilon')$
- 7: Let x' = (c'(x), y), storing the *exponent* for each coordinate of y
- 8: $X' \leftarrow X' \cup \{x'\}$
- 9: return (C', X')

its coordinates rounded to a power of $(1 + \varepsilon')$. Similarly, we use c' to denote $c(x) \in C'$. Thus, we have $||x - x'||_2 \le \varepsilon' \cdot ||c' - x||_2$ for all $x \in X$.

For ease of presentation, we first write X' so that the weights are not rounded to a power of $(1 + \varepsilon')$. By the triangle inequality,

$$\operatorname{cost}(C, X') = \sum_{x' \in X'} (\operatorname{dist}(x', C))^z \le \sum_{x' \in X'} (\operatorname{dist}(x', x) + \operatorname{dist}(x, C))^z.$$

Since dist $(x, x') = ||x - x'||_2 \le \varepsilon' \cdot ||c' - x||_2 = \varepsilon' \cdot \operatorname{dist}(x, C')$, then

$$\operatorname{cost}(C, X') \le \sum_{x' \in X'} \left(\varepsilon' \cdot \operatorname{dist}(x, C') + \operatorname{dist}(x, C) \right)^z$$

By Fact 2.5,

$$\operatorname{cost}(C, X') \leq \sum_{x' \in X'} \left(\left(1 + \frac{\varepsilon}{2} \right) \cdot (\operatorname{dist}(x, C))^z + \left(1 + \frac{4z}{\varepsilon} \right)^z \cdot (\varepsilon' \cdot \operatorname{dist}(x, C'))^z \right).$$

Thus,

$$\operatorname{cost}(C, X') \le \left(1 + \frac{\varepsilon}{2}\right) \cdot \operatorname{cost}(C, X) + \left(1 + \frac{4z}{\varepsilon}\right)^z (\varepsilon')^z \cdot \operatorname{cost}(C', X).$$

Since C' is a constant-factor approximation to the optimal (k, z)-clustering of X, then there exists a constant $\gamma \ge 1$ such that $\operatorname{cost}(C, X') \le \gamma \cdot \operatorname{cost}(C, X)$. Hence,

$$cost(C, X') \le \left(\left(1 + \frac{\varepsilon}{2} \right) + \left(1 + \frac{4z}{\varepsilon} \right)^z (\varepsilon')^z \cdot \gamma \right) \cdot cost(C, X).$$

Then for $\varepsilon' = \frac{\operatorname{poly}(\varepsilon^z)}{\operatorname{poly}(k, \log(nd\Delta))}$, we have that

$$cost(C, X') \le \left(1 + \frac{2}{3}\varepsilon\right) \cdot cost(C, X),$$

which almost gives the right hand side of the inequality (due to the notation X' representing the output prior to the rounding of the weights).

Similarly, by triangle inequality and $dist(x, x') \leq \varepsilon' \cdot dist(x, C')$,

$$\operatorname{cost}(C, X) = \sum_{x \in X} (\operatorname{dist}(x, C))^{z}$$
$$\leq \sum_{x \in X} (\operatorname{dist}(x, x') + \operatorname{dist}(x', C))^{z}$$
$$\leq \sum_{x' \in X'} (\varepsilon' \cdot \operatorname{dist}(x, C') + \operatorname{dist}(x', C))^{z}$$

By Fact 2.5,

$$\operatorname{cost}(C, X) \leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \operatorname{cost}(C, X') + \left(1 + \frac{4z}{\varepsilon}\right)^{z} (\varepsilon')^{z} \cdot \operatorname{cost}(C', X)$$
$$\leq \left(1 + \frac{\varepsilon}{2}\right) \cdot \operatorname{cost}(C, X') + \left(1 + \frac{4z}{\varepsilon}\right)^{z} (\varepsilon')^{z} \cdot \gamma \cdot \operatorname{cost}(C, X).$$

Then for $\varepsilon' = \frac{\operatorname{poly}(\varepsilon^z)}{\operatorname{poly}(k, \log(nd\Delta))}$, we have that

$$\operatorname{cost}(C, X) \le \left(1 + \frac{\varepsilon}{2}\right) \cdot \operatorname{cost}(C, X') + \frac{\varepsilon}{100} \cdot \operatorname{cost}(C, X),$$

so that $\left(1 - \frac{2}{3} \cdot \varepsilon\right) \operatorname{cost}(C, X) \leq \operatorname{cost}(C, X')$, which almost gives the left-hand side of the inequality (due to the notation X' representing the output prior to the rounding of the weights).

Finally, since the weights are rounded to a power of $(1 + \varepsilon')$, then the cost can only change by $(1 + \varepsilon')$. Since $\varepsilon' = \frac{\operatorname{poly}(\varepsilon^z)}{\operatorname{poly}(k, \log(nd\Delta))}$, then it follows that

$$(1 - \varepsilon) \cdot \operatorname{cost}(C, X) \le \operatorname{cost}(C, X') \le (1 + \varepsilon) \cdot \operatorname{cost}(C, X).$$

We now give the full guarantees of Algorithm 1, which gives an efficient encoding of an input set X.

Lemma 2.7. Let X be a coreset construction with weights bounded by $[1, \text{poly}(nd\Delta)]$. Then X' is a $(1 + \varepsilon)$ -strong coreset for X that uses $\mathcal{O}(dk \log(n\Delta)) + |X| \cdot \text{polylog}\left(k, \frac{1}{\varepsilon}, \log(nd\Delta), \log \frac{1}{\delta}\right)$ bits of space.

Proof. Algorithm 1 then outputs a set (C', X') that encodes X' using C'. By Lemma 2.6, we have that for all $C \subset [\Delta]^d$ with $|C| \leq k$,

$$(1 - \varepsilon) \cdot \operatorname{cost}(C, X) \le \operatorname{cost}(C, X') \le (1 + \varepsilon) \cdot \operatorname{cost}(C, X).$$

Thus, X' is a strong coreset for X.

It remains to analyze the space complexity of (C', X'). We first require C' to encode S. Since C' is a set of k centers, then C' can be represented using $\mathcal{O}(dk \log(n\Delta))$ bits of space. Additionally, each point x' in X' is encoded using $\mathcal{O}(d\log(\frac{1}{\varepsilon} + \log n + \log \Delta))$ bits of space due to storing the d coordinates of the rounded offset y. In particular, we do not store the explicit coordinates, but rather their exponents. Moreover, we use $\log k$ bits to store the closest center c'(x). Finally, each weight has weight $\operatorname{poly}(nd\Delta)$ and thus can be approximated to $(1 + \varepsilon)$ -multiplicative factor by storing the exponent, using $\mathcal{O}(\log\log(nd\Delta))$ bits of space. Therefore, the output of the algorithm is encoded using $\mathcal{O}(dk \log(n\Delta)) + |X| \cdot \operatorname{polylog}\left(k, \frac{1}{\varepsilon}, \log(nd\Delta), \log \frac{1}{\delta}\right)$ total bits of space. \Box

2.3Clustering in the Streaming Model

In this section, we show how to use a global encoding combined with the efficient encoding from the previous section in order to achieve our main algorithm for (k, z)-clustering on insertion-only data streams.

The main shortcoming of immediately applying the previous efficient encoding to each node of a merge-and-reduce tree is that the constant-factor approximation requires $\mathcal{O}(kd\log(n\Delta))$ bits of storage per efficient encoding, and there can be $polylog(log(n\Delta))$ such efficient encodings due to the height of the merge-and-reduce tree on a data stream produced by online sensitivity sampling. Instead, we provide a single constant-factor approximation to the global dataset and show that the error of maintaining such a global constant-factor approximation does not compound too much over the course of the stream.

The algorithm appears in full in Algorithm 2.

Algorithm 2 (k, z)-Clustering on Insertion-Only Stream

Input: Data set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that arrives as a data stream, $\varepsilon \in (0, 1)$, number of clusters k, parameter $z \ge 1$

Output: $(1 + \varepsilon)$ -coreset for (k, z)-clustering

1:
$$Z \leftarrow \emptyset, \lambda \leftarrow \mathcal{O}\left(\frac{k}{\varepsilon^2} \cdot \log k \log n\right)$$

- 2: for each $t \in [n] \overset{\varepsilon}{\operatorname{do}}$
- Let \hat{x}_t be the image of x_t after applying a JL transform into $\mathcal{O}(\log n)$ dimensions 3:
- Use $Z \cup \{\hat{x}_t\}$ to compute a $\mathcal{O}(1)$ -approximation $\sigma(x_t)$ to the online sensitivity of x_t 4:
- $p(x_t) \leftarrow \min(1, \lambda \cdot \hat{\sigma}(x_t))$ 5:
- With probability $p(x_t)$, sample x_t into a stream \mathcal{S}' with weight $\frac{1}{p(x_t)}$ 6:
- Let Z be the running output of merge-and-reduce on \mathcal{S}' using the efficient encoding for 7: coreset construction in Algorithm 1 with a global constant-factor approximation, for accuracy $\frac{\varepsilon}{\operatorname{poly}(\log\log n\Delta)}$ and failure probability $\frac{1}{\operatorname{poly}(\frac{1}{\varepsilon},\log(n\Delta))}$
- 8: return Z

We first show that given a partition $X_1 \sqcup \ldots \sqcup X_m$ of X, we can apply Algorithm 1 and although the output X'_i from our efficient encoding will no longer necessarily be a strong coreset for each corresponding X_i , the resulting error will only be an additive $\varepsilon' \cdot \operatorname{cost}(X, C)$, which suffices for our purposes. Specifically, we will have $\varepsilon' = \frac{\varepsilon}{\operatorname{poly}(\log\log(nd\Delta))}$ and ultimately have $m = \operatorname{poly}(\log\log(nd\Delta))$.

Lemma 2.8. Let $X_1 \sqcup \ldots \sqcup X_m = X$ be a partition of X for $m = \text{poly}(\log \log(nd\Delta))$. Let C' be a set of k centers that is a set C' of k centers that is a constant-factor approximation to (k, z)-clustering on X. For each $i \in [m]$, let X'_i be the set of rounded points corresponding to X_i from Algorithm 1. Then for all $i \in [m]$ and all $C \subset [\Delta]^d$ with $|C| \leq k$,

$$|\operatorname{cost}(X'_i, C) - \operatorname{cost}(X_i, C)| \le \frac{\varepsilon}{\operatorname{poly}(\log \log nd\Delta)} \cdot \operatorname{cost}(X, C).$$

Proof. Note that by Lemma 2.6 with accuracy $\frac{\varepsilon}{\operatorname{poly}(\log \log nd\Delta)}$, we have

$$|\operatorname{cost}(X', C) - \operatorname{cost}(X, C)| \le \frac{\varepsilon}{\operatorname{poly}(\log \log nd\Delta)} \cdot \operatorname{cost}(X, C).$$

Since

$$\operatorname{cost}(X'_i, C) - \operatorname{cost}(X_i, C)| \le \sum_{i=1}^m |\operatorname{cost}(X'_i, C) - \operatorname{cost}(X_i, C)| \le \varepsilon \cdot \operatorname{cost}(X, C),$$

then the claim follows.

It remains to give the full guarantees of Algorithm 2 by showing correctness of the global encoding and in particular, that there is no compounding error across the poly($\log \log(nd\Delta)$) iterations.

Theorem 2.9. Given a set X of n points on $[\Delta]^d$, let $f\left(n, d, \Delta, k, \frac{1}{\varepsilon}, z\right)$ be the number of points of a coreset construction with weights $[1, \text{poly}(nd\Delta)]$ for (k, z)-clustering. Then Algorithm 2 outputs a $(1 + \varepsilon)$ -strong coreset of X and uses $\mathcal{O}(dk \log(n\Delta)) + f\left(n, d, \Delta, k, \frac{\text{polylog}(\log(nd\Delta))}{\varepsilon}, z\right) \cdot$ polylog $\left(\frac{1}{\varepsilon}, \log(nd\Delta)\right)$ bits of space.

Proof. We first show that with high probability, the following invariants are satisfied at each time:

- (1) $\widehat{\sigma(x_1)}, \ldots, \widehat{\sigma(x_t)}$ are constant-factor approximations to the online sensitivities $\sigma(x_1), \ldots, \sigma(x_t)$ with respect to (k, z)-clustering
- (2) At most poly $\left(k, \frac{1}{\varepsilon}, \log(nd\Delta)\right)$ points have been sampled into \mathcal{S}'
- (3) Z at time t is a $(1 + \varepsilon)^2$ -coreset to $X_t := \{x_1, \dots, x_t\}$

We prove correctness by induction on t. The online sensitivity of the first point is 1 and it will be correctly computed by the data structure. Thus we have that after the first step, we have $S' = Z = \{x_1\}$ and the base case is complete.

Let \mathcal{E}_t be the event that the invariants hold at time t. Conditioning on \mathcal{E}_{t-1} and in particular the correctness of Z at time t-1 being a $(1+\varepsilon)$ -strong coreset of $X_{t-1} := \{x_1, \ldots, x_{t-1}\}$, we have that $Z \cup \{x_t\}$ is a strong coreset of $X_t := \{x_1, \ldots, x_t\}$. Thus we can use $Z \cup \{x_t\}$ to compute a constant-factor approximation to the online sensitivity $\sigma(x_t)$ of x_t . By Theorem 1.7, it follows that with high probability, we can instead use $Z \cup \{\hat{x}_t\}$ to compute a constant-factor approximation $\widehat{\sigma(x_t)}$ to the online sensitivity $\sigma(x_t)$. Then by Theorem 2.4, \mathcal{S}' after time t will be a $(1+\varepsilon)$ -strong coreset for X_t with high probability.

By Theorem 2.4 through Theorem 2.3, we have that by sampling points with probabilities that are constant-factor approximations to their online sensitivities, then the total number of sampled points into S' is $\mathcal{O}\left(\frac{dk^2}{\varepsilon^2}\log^3\frac{n\Delta}{\varepsilon}\right)$ with high probability. Thus, $\mathbf{Pr}\left[\mathcal{E}_t\right] \geq 1 - \frac{1}{\operatorname{poly}(n)}$.

Observe that merge-and-reduce will be correct if S' is a stream with length at most $k \, \cdot \, \text{poly}\left(\frac{1}{\varepsilon}, \log(nd\Delta)\right)$, since we set the failure probability to be $\frac{1}{\text{poly}\left(\frac{1}{\varepsilon}, \log(n\Delta)\right)}$ and merge-and-reduce will consider at least k stream updates before applying a new coreset construction. Thus conditioned on (1) the correctness of S' after time t being a $(1 + \varepsilon)$ -strong coreset for X_t and (2) the correctness of merge-and-reduce on S', we have that Z after time t will be a $(1 + \varepsilon)$ -strong coreset for S', and thus a $(1 + \varepsilon)^2$ -strong coreset for X_t , which completes the induction. That is, $\Pr[\mathcal{E}_t \mid \mathcal{E}_1, \ldots, \mathcal{E}_{t-1}] \geq 1 - \frac{1}{\text{poly}(n)}$. The correctness guarantee then follows by rescaling ε and a union bound over all $t \in [n]$.

To analyze the space complexity, note that by the guarantees of the invariants, the total number of sampled points into \mathcal{S}' is $\mathcal{O}\left(\frac{dk^2}{\varepsilon^2}\log\frac{nd\Delta}{\varepsilon}\right)$. However, \mathcal{S}' is not maintained explicitly, but instead given

as input to the merge-and-reduce subroutine, which uses the efficient encoding for coreset construction given in Algorithm 1 with accuracy $\frac{\varepsilon}{\operatorname{poly}(\log\log(nd\Delta))}$ and failure probability $\frac{1}{\operatorname{poly}(\frac{1}{\varepsilon},\log(nd\Delta))}$. Thus by Lemma 2.7, the total representation of Z uses $\mathcal{O}(dk\log(n\Delta)) + \frac{dk^2}{\varepsilon^2} \cdot 2^{2z} \cdot \operatorname{polylog}(\frac{1}{\varepsilon},\log(n\Delta))$ bits of space.

3 Fast (k, z)-Clustering

In this section, we describe how our one-pass streaming algorithm on insertion-only streams for (k, z)-clustering that uses $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space can further be implemented using fast amortized update time. In Section 3.2, we first implement our streaming algorithm using $dk \cdot \text{polylog}(\log(n\Delta))$ amortized update time. The $o(\log(n\Delta))$ amortized update time of this algorithm is not only faster than the $k \cdot \text{polylog}(n\Delta)$ amortized update time of [BCLP23], but also provides crucial structural properties that will be ultimately used for our algorithm that uses $d \log k \cdot \text{polylog}(\log(n\Delta))$ amortized update time in Section 3.3. One of the such crucial properties is the proof in Section 3.1 that the (k, z)-clustering sensitivities and the (k, z)-mediods sensitivities are within constant factors of each other.

3.1 (k, z)-Clustering Sensitivities and (k, z)-medoids Sensitivities

In this section, we show that the sensitivity $\tau(x)$ for a point x with respect to the dataset X for k-medoids is a constant-factor approximation to the sensitivity s(x) for a point x with respect to the dataset X for (k, z)-clustering. Thus to acquire a constant-factor approximation to the sensitivity s(x), it suffices to approximate the (k, z)-medoids sensitivity $\tau(x)$, and vice versa.

We first recall the well-known fact that the optimal (k, z)-clustering and (k, z)-medoids costs are within a constant factor of each other. For the sake of completeness, we include the proof.

Lemma 3.1. For each $X \subset [\Delta]^d$, the optimal (k, z)-clustering cost is a 2^{z+1} -approximation of the optimal (k, z)-medoid clustering cost.

Proof. Let C be an optimal (k, z)-clustering of X and for a fixed $c \in C$, let P be the subset of X served by c. Let $c' \in P$ be the point closest to c. Then we have for all $p \in P$,

$$||c'-p||_2^z \le 2^z (||c'-c||_2^2 + ||p-c||_2^2) \le 2^{z+1} ||p-c||_2^2.$$

Therefore, $\cot(P, c') \leq 2^{z+1} \cdot \cot(P, c)$. The conclusion then follows by iterating over all $c \in C$. \Box

We now show that the (k, z)-clustering and (k, z)-medoids sensitivities are within a constant factor of each other.

Theorem 3.2. There exists a constant $\gamma \leq 2^{z+1} \cdot 101$ such that the sensitivity of a point x with respect X for (k, z)-clustering is a γ -approximation to the sensitivity of x with respect to X for (k, z)-medoids.

Proof. Let C be any set of k centers that achieves the sensitivity s(x) for (k, z)-clustering and let S be any set of k centers that achieves the sensitivity $\tau(x)$ for (k, z)-medoids. Since $S \subseteq X \subseteq [\Delta]^d$ and $|S| \leq k$, then by the maximality characterization of sensitivity, we have $\tau(x) \leq s(x)$.

Now, let $c \in C$ be the closest center to x. Let $R \subset X$ be the set of points served by c and let r = |R|. Firstly, let $w \in R$ be the farthest point from x. Then

$$\max_{c' \in X} \frac{\cot x, c'}{\cot(R, c')} \ge \frac{\|w - x\|_2}{\sum_{y \in R} \|w - y\|_2} \ge \frac{\|w - x\|_2}{\sum_{y \in R} \|w - x\|_2 + \|y - x\|_2} \ge \frac{1}{2|R|}$$

since $||w - x||_2 \ge ||y - x||_2$ for all $y \in R$. Thus, $\frac{||w - x||_2}{\sum_{y \in R} ||w - y||_2} \ge \frac{1}{2r}$.

Let $P \subseteq R$ be the set of points y that are closer to x than to c. We perform casework on the size |P| of P.

First, suppose |P| < 0.99r. Note that there are at most 0.01r points y such that $\cot(y,c) > \frac{100}{r} \cdot \cot(R,c)$. Thus there are at least 0.99r points y such that $\cot(y,c) \le \frac{100}{r} \cdot \cot(R,c)$. If |P| < 0.99r, then there exists $y \in R$ such that $y \notin P$, so that $\cot(y,c) \le \frac{100}{r} \cdot \cot(R,c)$ but $||y-c||_2 \le ||y-x||_2$. But then if we set c' = y, we have that $||x-c'||_2 \ge \frac{1}{2}||x-c||_2$ and moreover,

$$\operatorname{cost}(R,c') \le 2^{z}(\operatorname{cost}(R,c) + |R| \cdot \operatorname{cost}(C,c')) \le 2^{z}(101 \cdot \operatorname{cost}(R,c)).$$

Hence we have

$$\frac{\operatorname{cost}(x,c')}{\operatorname{cost}(R,c')} \ge \frac{1}{2^{z+1} \cdot 101} \frac{\operatorname{cost}(x,c)}{\operatorname{cost}(R,c)}$$

The claim then follows by clustering $X \setminus R$ with the remaining k-1 centers using Lemma 3.1.

Otherwise, suppose $|P| \ge 0.99r$. Furthermore, it suffices to assume that there exist 0.99r points $y \in P$ such that $\cos(y, c) \le \frac{100}{r} \cdot \cos(R, c)$, or else we could reach the same conclusion as the previous case where |P| < 0.99r by moving c to such a point y. Since these points are closer to x than to c by definition of P, then we have $||y - c||_2 \ge \frac{1}{2} \cdot ||x - c||_2$. Thus we have $\cos(R, c) \ge 0.99r \cdot \frac{1}{2} \cdot ||x - c||_2^2$. Hence, $\frac{\cos(x, c)}{\cos(R, c)} \le \frac{4}{r}$. On the other hand, by the above argument, we have $\max_{c' \in X} \frac{\cos x, c'}{\cos(R, c')} \ge \frac{1}{2r}$ and thus $\tau(x)$ is a $2^{z+1} \cdot 8$ -approximation to s(x), after clustering the remaining points in $X \setminus R$ with the other k - 1 centers.

Finally, we show that the sensitivities of points are preserved under coresets. That is, the (k, z)-clustering sensitivity of x with respect to X is well-approximated by the sensitivity of x with respect to a coreset Z of X.

Lemma 3.3. Let $\gamma \geq 1$ be fixed and let Z be a γ -strong coreset for X. For $p \in X$, let $s_X(p) = \max_{C:|C| \leq k} \frac{\operatorname{cost}(p,C)}{\operatorname{cost}(X,C)}$ and let $s_Z(p) = \max_{C:|C| \leq k} \frac{\operatorname{cost}(p,C)}{\operatorname{cost}(Z,C)}$. Then

$$\frac{1}{\gamma} \cdot s_Z(p) \le s_X(p) \le \gamma \cdot s_Z(p).$$

Proof. Let S be a set of k centers in C that achieves the sensitivity with respect to Z, so that $s_Z(p) = \frac{\cot(p,S)}{\cot(X,S)}$. Since Z is a γ -strong coreset for X, then we have $\frac{1}{\gamma} \cdot \cot(Z,S) \leq \cdot \cot(X,S) \leq \gamma \cdot (Z,S)$. Therefore, we have

$$\frac{1}{\gamma} \cdot s_Z(p) \le s_X(p) \le \gamma \cdot s_Z(p).$$

3.2 Fast Update Algorithm

In this section, we show that our streaming algorithm can be implemented using $dk \cdot \text{polylog}(\log(n\Delta))$ amortized update time. We remark that by comparison, the algorithm of [BCLP23] uses $k \cdot \text{polylog}(n\Delta)$ amortized update time, which is exponentially larger in terms of the dependency in the n and Δ factors. On the other hand, we remark that the main point of [BCLP23] is to handle the fully dynamic case, where insertions and deletions of points are both permitted, which we cannot handle, though they also require the entire dataset to be stored.

We first use the following guarantee about quadratic runtime for the local search algorithm for (k, z)-clustering.

Theorem 3.4. [GT08] For each constant $z \ge 1$, there exists a polynomial time algorithm that outputs a $\mathcal{O}(z)$ -approximation to (k, z)-clustering in time $\mathcal{O}(dn^2)$.

We now show how the approximately solve the constrained clustering problem min cost(X, C) across all sets $C \subset X$ of k centers containing a center at a fixed center x. The algorithm is quite simple. We compute a constant-factor approximation C to the optimal (k, z)-medoids clustering on X. We then swap one of the centers of C with $\{x\}$, choosing the swap with the best subsequent clustering cost.

Lemma 3.5. For a fixed constant $\gamma \geq 1$, let C be a γ -approximation to the optimal (k, z)-medoids clustering on $X \subset [\Delta]^d$. Let S be the optimal (k, z)-medoids clustering on $X \subset [\Delta]^d$ containing a center at a fixed $x \in X$. Let W be the set of k - 1 centers $W \subseteq C$ such that $W \cup \{x\}$ has the minimum (k, z)-medoids clustering cost on X, and let $C' = W \cup \{x\}$. Then

$$\cot(X, S) \le \cot(X, C') \le (2^z + 2^{2z} + 2^{2z} \cdot \gamma) \cdot \cot(X, S).$$

Proof. Since C' is a set of k centers containing x and S is the optimal clustering among such sets of k centers containing x, then $cost(X, S) \leq cost(X, C')$. It thus remains to prove the right-hand side of the inequality.

Let $S' = C \cup \{p\}$ so that |S'| = k + 1 and also $C' \subset S'$ with $|S' \setminus C'| = 1$. Let $\pi_{S'} : S \to C'$ be the mapping that assigns each center $s \in S$ to the closest center in S', breaking ties arbitrarily. Let $Q = \{\pi_{S'}(s) \mid s \in S\}$ and note that $x \in Q$ since $x \in S$ and $x \in S'$. Moreover, since |S| = k, then |Q| = k, and since $Q \subset S'$ and C' is the best subset of S', then $\operatorname{cost}(X, C') \leq \operatorname{cost}(X, Q)$.

Now for any $p \in X$, consider its contribution to cost(X, Q). Let $\pi_S : X \to S$ be the mapping that assigns each point $x \in X$ to the closest center in S, breaking ties arbitrarily. Then by generalized triangle inequality, c.f., Fact 1.4,

$$\operatorname{cost}(p,Q) \le 2^{z} \cdot \operatorname{cost}(p,\pi_{S}(p)) + 2^{z} \cdot \operatorname{cost}(\pi_{S}(p),Q).$$

Since $\pi_{S'}$ maps each center in S to its closest center in S', then we have

$$\operatorname{cost}(\pi_S(p), Q) = \operatorname{cost}(\pi_S(p), S').$$

Thus,

$$\operatorname{cost}(p,Q) \le 2^z \cdot \operatorname{cost}(p,\pi_S(p)) + 2^z \cdot \operatorname{cost}(\pi_S(p),S').$$

We also have by generalized triangle inequality, c.f., Fact 1.4,

$$\operatorname{cost}(\pi_S(p), S') \le 2^z \cdot \operatorname{cost}(\pi_S(p), p) + 2^z \cdot \operatorname{cost}(p, S') = 2^z \cdot \operatorname{cost}(p, S) + 2^z \cdot \operatorname{cost}(p, S').$$

Therefore,

$$\operatorname{cost}(p,Q) \le 2^{z} \cdot \operatorname{cost}(p,\pi_{S}(p)) + 2^{2z} \cdot \operatorname{cost}(p,S) + 2^{2z} \cdot \operatorname{cost}(p,S')$$
$$= 2^{z} \cdot \operatorname{cost}(p,S) + 2^{2z} \cdot \operatorname{cost}(p,S) + 2^{2z} \cdot \operatorname{cost}(p,S').$$

Summing across $p \in X$, we thus have

$$\operatorname{cost}(X,Q) \le 2^{z} \cdot \operatorname{cost}(X,S) + 2^{2z} \cdot \operatorname{cost}(X,S) + 2^{2z} \cdot \operatorname{cost}(X,S').$$

Since C provides a γ -approximation to the optimal (k, z)-medoids clustering on $X \subset [\Delta]^d$ and S is the optimal (k, z)-medoids clustering on X containing a center at x, then $\operatorname{cost}(X, C) \leq \gamma \cdot \operatorname{cost}(X, S)$. Since $S' = C \cup \{p\}$, then $\operatorname{cost}(X, S') \leq \operatorname{cost}(X, C) \leq \gamma \cdot \operatorname{cost}(X, S)$ and thus we have

$$\cot(X, Q) \le (2^z + 2^{2z} + 2^{2z} \cdot \gamma) \cdot \cot(X, S).$$

Finally, since $cost(X, C') \leq cost(X, Q)$, then the desired claim follows.

Algorithm 3 Approximation algorithm for finding (k, z)-medoids clustering containing a fixed center

Input: Constant-factor approximation $C \subset X \subset [\Delta]^d$ for (k, z)-medoids clustering on X, query point $x \in X$

Output: Constant-factor approximation for (k, z)-medoids clustering on X with a center at x 1: $\psi \leftarrow nd\Delta, \ \psi_x \leftarrow \emptyset$

2: for $c \in C$ do

3: Let n_c be the number of points in X assigned to c by C

4: **if** $n_c \cdot ||c - x||_2^z < \psi$ **then**

5:
$$\psi \leftarrow n_c \cdot \min_{p \in C \cup \{x\} \setminus \{c\}} \|c - p\|_2^z, \psi_x \leftarrow c$$

6: return $(\psi_x, \operatorname{cost}(X, C) + \psi)$

Unfortunately, we cannot afford to exactly compute the cost of the best replacement of a center in C with $\{x\}$. Thus we show that by snapping the points served by each center $c \in C$ and moving these points all to the closest center if c is removed gives a good approximation to the cost of the best swap. In particular, if n_c is the number of points served by c and p is the closest center of $C \cup \{x\}$ to c, then $cost(X, C) + n_c \cdot ||c - p||_2^z$ is a good approximation to the cost of replacing c with x in C.

Lemma 3.6. Let $\Psi = \operatorname{cost}(X, C) + \psi$ be the output of Algorithm 3. Let S be the optimal (k, z)medoids clustering on $X \subset [\Delta]^d$ containing a center at a fixed $x \in X$. Then there exists a constant $\gamma = 2^{\Theta(z)}$ such that

$$\cot(X, S) \le \Psi \le \gamma \cdot \cot(X, S).$$

Proof. Let W be the set of k-1 centers $W \subseteq C$ such that $W \cup \{x\}$ has the minimum (k, z)-medoids clustering cost on X, and let $C' = W \cup \{x\}$. By Lemma 3.5, there exists a constant $\gamma_1 \ge 1$ such that

$$\cot(X, S) \le \cot(X, C') \le (2^z + 2^{2z} + 2^{2z} \cdot \gamma_1) \cdot \cot(X, S).$$

We claim Ψ gives a constant-factor approximation to cost(X, C').

By generalized triangle inequality,

$$\operatorname{cost}(X,C') \le 2^z \cdot \operatorname{cost}(X,C) + 2^z \cdot \operatorname{cost}(C,C') \le 2^z \cdot \operatorname{cost}(X,C) + 2^{2z} \cdot \operatorname{cost}(X,C) + 2^{2z} \cdot \operatorname{cost}(X,C').$$

Let C be a γ_2 -approximation to the optimal unconstrained (k, z)-medoids clustering. Then $\cot(X, C) \leq \gamma_2 \cdot \cot(X, C')$, so that

$$\cot(X, C') \le 2^z \cdot \cot(X, C) + 2^z \cdot \cot(C, C') \le (2^z \gamma_2 + 2^{2z} \gamma_2 + 1) \cdot \cot(X, C').$$

Hence, it suffices to show that Ψ gives a constant-factor approximation to cost(X, C) + cost(C, C').

Note that $|C \setminus C'| = 1$, then $\operatorname{cost}(C, C') = \min_{c \in C} n_c \cdot \min_{p \in C'} ||c - p||_2^z$, where n_c is the number of points assigned to c by X. Thus, we have $\psi = \operatorname{cost}(C, C')$, so that

$$\Psi = \cot(X, C) + \cot(C, C').$$

Putting things together, we have that there exists a constant $\gamma = 2^{\Theta(z)}$ such that

$$\cot(X, S) \le \Psi \le \gamma \cdot \cot(X, S).$$

| r. | | |
|----|--|--|
| н | | |
| L | | |
| н | | |
| L | | |

Unfortunately, we may not have time to compute the closest center p to each center c among $C \cup \{x\} \setminus \{c\}$. Thus we show that it suffices to just update the information for the center $u \in C$ that is closest to x.

Lemma 3.7. Let C be a constant-approximation to the optimal (k, z)-medoids clustering on $X \subset [\Delta]^d$. For a point $x \in X$, let u be the center of C closest to x, breaking ties arbitrarily. Then for any center $c \in C \setminus \{u\}$,

$$\min_{p \in C \cup \{x\} \setminus \{c\}} \|c - p\|_2^z \le \min_{p \in C \setminus \{c\}} \|c - p\|_2^z \le 2^{z+1} \min_{p \in C \cup \{x\} \setminus \{c\}} \|c - p\|_2^z.$$

Proof. Note that the left-hand side of the inequality immediately holds because the minimization is taken over a larger superset on the left-most term. We now justify the right-hand side of the inequality.

Let p be the closest center of $S \setminus \{c\}$ to c. Observe that if $||c - p||_2 \le ||c - x||_2$, then p remains the closest center of $S \cup \{x\} \setminus \{c\}$ to c and thus the claim holds.

Hence, it remains to consider the setting where $||c - p||_2 < ||c - x||_2$. In this case, we have by the optimality of p in S, $||c - p||_2 \le ||c - u||_2$. Thus by generalized triangle inequality,

$$||c - p||_2^z \le ||c - u||_2^z \le 2^z \cdot ||u - x||_2^z + 2^z \cdot ||c - x||_2^z$$

Since u is the closest center in S to x and $c \in S$, then it follows that $||u - x||_2 \leq ||c - x||_2$ and thus

$$||c-p||_2^z \le 2^{z+1} \cdot ||c-x||_2^z.$$

To estimate the sensitivity of a point x, we enumerate over all possible centers p that serve x. To that end, we show that $\Phi + n_b \cdot r^z$ can only be an underestimate to the optimal (k, z)-medoids clustering constrained to the center p being the closest to x. **Lemma 3.8.** Let $x \in X$ be a fixed point and $p \in X$ be a fixed center with $r = ||x - p||_2$. Let Q be a constant-factor approximation to the optimal (k, z)-medoids clustering on X containing a center at p and let Ψ be a constant-factor approximation to cost(X,Q). Let S be the optimal (k,z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$. Let n_b be the number of points assigned to centers in $B_{r/2}(x)$. There exists a constant $\gamma \ge 1$ such that

$$\Psi + n_b \cdot r^z \le \gamma \cdot \operatorname{cost}(X, S).$$

Proof. Let Q be a ζ_1 -approximation to the optimal (k, z)-medoids clustering on X containing a center at p. Since S is the optimal solution for a more constrained search space, i.e., sets of at most k centers that contain a center at p and no center in the interior of $B_r(x)$, then we have

$$\cot(X, Q) \le \zeta_1 \cdot \cot(X, S).$$

Let Ψ be a ζ_2 -approximation to cost(X, Q), so that

$$\Psi \le \zeta_1 \zeta_2 \cdot \operatorname{cost}(X, S).$$

Now, consider a point y served by a center $q \in Q$ inside $B_{r/2}(x)$. If $||y - q||_2 \ge \frac{r}{4}$, then we have

$$||y - q||_2 + r \le 5 \cdot ||y - q||_2$$

Otherwise, if $||y-q||_2 < \frac{r}{4}$, then by triangle inequality, $dist(y, S) \ge dist(q, S) - ||y-q||_2 \ge \frac{3r}{4}$. Thus,

$$||y - q||_2 + r \le ||y - q||_2 + 2 \cdot \operatorname{dist}(y, S).$$

Taking both cases together, we have

$$||y - q||_2 + r \le 5 \cdot ||y - q||_2 + 2 \cdot \operatorname{dist}(y, S).$$

Summing across all y and q, we have

$$\operatorname{cost}(X,Q) + n_b \cdot r^z \le 5 \operatorname{cost}(X,Q) + 2 \operatorname{cost}(X,S) \le (5\gamma + 2) \cdot \operatorname{cost}(X,S).$$

Since Ψ is a ζ_2 -approximation to cost(X, Q), then there exists a constant $\gamma_2 \ge 1$ such that

$$\Psi + n_b \cdot r^z \le \gamma \cdot \cot(X, S).$$

Now we show that if there is a large number of points served by centers inside the ball $B_{r/2}(x)$ of radius $\frac{r}{2}$ around x, where $r := ||x - p||_2$, then $\Phi + n_b \cdot r^z$ is a good estimate to the cost of the optimal (k, z)-medoids clustering constrained to the center p being the closest to x.

Lemma 3.9. Let $x \in X$ be a fixed point and $p \in X$ be a fixed center with $r = ||x - p||_2$. Let Q be a constant-factor approximation to the optimal (k, z)-medoids clustering on X containing a center at p and let Ψ be a constant-factor approximation to cost(X, Q). Let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$. Let n_b be the number of points assigned to centers in $B_{r/2}(x)$ and let n_a be the number of points assigned to centers in $B_{r/2}(x)$. If $n_b \ge n_a$, then there exist constants $\gamma_1, \gamma_2 \ge 1$ such that

$$\operatorname{cost}(X,S) \le \gamma_1(\Psi + n_b \cdot r^z) \le (\gamma_1 \gamma_2) \cdot \operatorname{cost}(X,S)$$

Proof. We show that there exists a constant $\gamma_1 \geq 1$ with $\operatorname{cost}(X, S) \leq \gamma_1(\Psi + n_b \cdot r^z)$. Observe that this would imply by Lemma 3.8 that there also exists a constant $\gamma_2 \geq 1$ such that $\gamma_1(\Psi + n_b \cdot r^z) \leq (\gamma_1 \gamma_2) \cdot \operatorname{cost}(X, S)$.

Recall that S is the optimal solution across sets of at most k centers that contain a center at p and no center in the interior of $B_r(x)$.

Consider a point y served by a center $q \in Q$ inside $B_{r/2}(x)$. Let $\pi_S(y)$ be the closest center in S to y, breaking ties arbitrarily. By generalized triangle inequality,

$$cost(y, S) \le 2^{z} \cdot ||y - q||_{2}^{z} + 2^{z} \cdot cost(q, \pi_{S}(y)).$$

Summing across all y served by centers $q \in B_{r/2}(x)$ and noting that $dist(q, S) \leq 2r$, we have

$$\sum_{y:\pi_Q(y)\in B_{r/2}(x)}\operatorname{cost}(y,S) \le 8^z \cdot n_b \cdot \left(\frac{r}{2}\right)^z + 2^z \cdot \sum_{y:\pi_Q(y)\in B_{r/2}(x)}\operatorname{cost}(y,Q).$$

Next, consider a point y served by a center $q \in Q$ inside $B_r(x) \setminus (B_{r/2}(x))$. Note that if $\|y-q\|_2 \geq \frac{r}{4}$, then $\|y-p\|_2 \leq \|y-q\|_2 + \|q-p\|_2 \leq 9 \cdot \|y-q\|_2$. Otherwise if $\|y-q\|_2 < \frac{r}{4}$, then $\|y-p\|_2 \leq \|y-q\|_2 + \|q-p\|_2 < 3r$. Summing across all $y \in X$ served by centers $q \in B_{r/2}(x)$ and noting that $n_b \geq n_a$, we have

$$\sum_{y:\pi_Q(y)\in B_r(x)\setminus (B_{r/2}(x))} \operatorname{cost}(y,S) \le 12^z \cdot n_b \cdot \left(\frac{r}{2}\right)^z + 9 \cdot 2^z \cdot \sum_{y:\pi_Q(y)\in B_r(x)\setminus (B_{r/2}(x))} \operatorname{cost}(y,Q).$$

Finally, for the points y served by a center $q \in Q$ outside $B_r(x)$, note that S can only have more centers outside $B_r(x)$ than C because S cannot have centers inside $B_r(x)$ whereas C can. Thus by optimality of S,

$$\sum_{y:\pi_Q(y)\in X\backslash (B_r(x))} \operatorname{cost}(y,S) \leq \sum_{y:\pi_Q(y)\in X\backslash (B_r(x))} \operatorname{cost}(y,C).$$

Putting these together by summing across all y served by centers $q \in B_{r/2}(x)$, $q \in B_r(x) \setminus B_{r/2}(x)$, and $q \in X \setminus B_r(x)$, we have that there exists a constant $\zeta \ge 1$ such that

$$\operatorname{cost}(X,S) \le \zeta \cdot 2^{\mathcal{O}(z)} \cdot \left(\operatorname{cost}(X,Q) + n_b \cdot \left(\frac{r}{2}\right)^z \right).$$

Since Φ is a ζ_2 -factor approximation to cost(X,Q), then there exists a constant γ_1 such that

$$\operatorname{cost}(X,S) \le \gamma_1(\Psi + n_b \cdot r^z).$$

Therefore, we have

$$\operatorname{cost}(X,S) \le \gamma_1(\Psi + n_b \cdot r^z) \le (\gamma_1 \gamma_2) \cdot \operatorname{cost}(X,S),$$

as desired.

On the other hand, if the number of points served by centers inside the ball $B_{r/2}(x)$ of radius $\frac{r}{2}$ around x is small, then $\Phi + n_b \cdot r^z$ may not be a good estimate to the cost of the optimal (k, z)-medoids clustering S constrained to the center p being the closest to x. Nevertheless, we show that it can provide an upper bound on $\frac{\cot(x,S)}{\cot(X,S)}$, while still being at most the sensitivity s(x) of x.

Lemma 3.10. Let $x \in X$ be a fixed point and $p \in X$ be a fixed center with $r = ||x - p||_2$. Let s(x)be the sensitivity of (k, z)-medoids clustering with respect to X. Let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$. Let n_b be the number of points assigned to centers in $B_{r/2}(x)$ and let n_a be the number of points assigned to centers in the annulus $B_r(x) \setminus B_{r/2}(x)$. If $n_b < n_a$, then

$$\frac{1}{\gamma_1} \frac{\operatorname{cost}(x, S)}{\operatorname{cost}(X, S)} \le \frac{r^z}{\Psi + n_b \cdot r^z} \le 2^z \cdot \gamma_2 \cdot s(x),$$

for fixed constants $\gamma_1, \gamma_2 \geq 1$.

Proof. Let Q be a ζ_1 -factor approximation to the optimal (k, z)-medoids clustering on X containing a center at p and let Ψ be a ζ_2 -factor approximation to cost(X,Q). By Lemma 3.8, we have that there exists a constant $\gamma_1 \geq 1$ such that

$$\Psi + n_b \cdot r^z \le \gamma_1 \cdot \operatorname{cost}(X, S).$$

Since S contains a center at p and no centers in the interior of $B_r(x)$, then $cost(x, S) = r^z$. Then we have

$$\frac{1}{\gamma_1} \cdot \frac{\cot(x,S)}{\cot(X,S)} \le \frac{r^z}{\Psi + n_b \cdot r^z}$$

On other hand, for a set W of k centers that contains no center in the interior of $B_{r/2}(x)$ and a center at p, so that we have $\frac{r}{2} \leq \operatorname{dist}(x, W) \leq r$ and $\operatorname{cost}(X, W) \leq \gamma_2(\Psi + n_b \cdot r^z)$ for some constant $\gamma_2 \geq 1$. Thus by the maximality of sensitivity, we have

$$\frac{r^z}{\Psi + n_b \cdot r^z} \le 2^z \cdot \gamma_2 \cdot \frac{\operatorname{cost}(x, W)}{\operatorname{cost}(X, w)} \le 2^z \cdot \gamma_2 \cdot s(x).$$

Algorithm 4 BATCHSENS: Fast batch approximation of sensitivities

Input: Input set $Z \subset [\Delta]^d$ for (k, z)-medoids clustering, batch B of k query points **Output:** Constant-factor approximation for the sensitivity of x for (k, z)-medoids clustering on

- $Z \cup B$, for all $x \in B$
- 1: Compute S to be a constant-factor solution on $Z \cup B$ for (k, z)-medoids clustering

2: for $x \in B$ do

- $s(x) \leftarrow 0$ 3:
- for $p \in Z \cup B$ do 4:

Let Q be a constant-factor approximation of the optimal (k, z)-medoids clustering on X 5:containing a center at p \triangleright Only need to pre-process and compute once per p across all x Use S to compute a constant-factor approximation Ψ of cost(X,Q)⊳Algorithm 3 6: 7: $r \leftarrow \|x - p\|_2$ (x)

8: Let
$$n_b$$
 be the number of points assigned by Q to centers of Q in $B_{r/2}(x)$

9:
$$s(x) \leftarrow \max\left(s(x), \frac{r^z}{\Psi + n_b \cdot r^z}\right)$$

return s(x)10:

It follows that our algorithm in BATCHSENS provides constant-factor approximations to all sensitivities x in the batch B.

Lemma 3.11. For a fixed $x \in X$, let s(x) be the sensitivity of (k, z)-medoids clustering with respect to X. Let $\widehat{s(x)}$ be the output of Algorithm 4, for each $x \in B$, where B is the input batch. Then there exist constants $\gamma_1, \gamma_2 \ge 1$ such that

$$\frac{1}{\gamma_1} \cdot s(x) \le \widehat{s(x)} \le 2^z \cdot \gamma_2 \cdot s(x).$$

Proof. For each fixed $p \in X$, let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$. Let n_b be the number of points assigned to centers in $B_{r/2}(x)$ and let n_a be the number of points assigned to centers in the annulus $B_r(x) \setminus B_{r/2}(x)$. We perform casework on whether $n_b < n_a$ or $n_b \ge n_a$.

In the case where $n_b \ge n_a$, then by Lemma 3.9, we have

$$\operatorname{cost}(X,S) \le \gamma_1(\Psi + n_b \cdot r^z) \le (\gamma_1 \gamma_2) \cdot \operatorname{cost}(X,S).$$

Since $\cot(x, S) = r^z$, then it holds that $\frac{r^z}{\Psi + n_b \cdot r^z}$ is a constant-factor approximation to $\frac{\cot(x,S)}{\cot(x,S)}$. Otherwise if $n_b < n_a$, then by Lemma 3.10,

$$\frac{1}{\gamma_1} \frac{\cot(x, S)}{\cot(X, S)} \le \frac{r^z}{\Psi + n_b \cdot r^z} \le 2^z \cdot \gamma_2 \cdot s(x),$$

for fixed constants $\gamma_1, \gamma_2 \geq 1$. In particular, for the center $p \in X$ that realizes the sensitivity s(x), we have that $\frac{r^z}{\Psi + n_b \cdot r^z}$ is an $\mathcal{O}(2^z)$ -approximation to s(x). For centers $q \in X$ that do not realize the sensitivity, we have that $\frac{r^z}{\Psi + n_b \cdot r^z} \leq \gamma_2 \cdot s(x)$. Therefore,

$$\frac{1}{\gamma_1} \cdot s(x) \le \widehat{s(x)} \le 2^z \cdot \gamma_2 \cdot s(x).$$

It remains to analyze the amortized runtime of the algorithm BATCHSENS.

Lemma 3.12. Given a constant-factor coreset Z to $X \subset [\Delta]^d$, there exists an absolute constant C > 1 and an algorithm BATCHSENS that outputs C-approximations to the sensitivities to each of the points $x \in B$ in a batch B of k points of X, using amortized $\frac{(|Z|+k \log k)^2}{k} \cdot \operatorname{poly}(d)$ time.

Proof. Consider BATCHSENS in Algorithm 4. By Lemma 3.3, to compute the sensitivity of $x \in B$ with respect to X, it suffices to compute the sensitivity of $x \in Z \cup B$, since Z is a constant-factor coreset to Y and thus $Z \cup B$ is a constant-factor coreset to $X = Y \cup B$. By Theorem 3.2, it suffices to compute a constant-factor approximation to the (k, z)-medoids clustering sensitivity of x with respect to $Z \cup B$. Hence, we want to compute $\max_{C \subset Z \cup B: |C| \le k} \frac{\cot(x, C)}{\cot(X, C)}$. Correctness thus follows by Lemma 3.11.

It remains to analyze the amortized runtime of BATCHSENS. First, note that it takes time $\mathcal{O}(d(|Z|+k)^2)$ to compute a set S of k centers that serve as a constant-factor approximation to $Z \cup B$. We can then pre-process $(Z \cup B)$ in $\mathcal{O}(d(|Z|+k)^2)$ time so that for any $x \in Z \cup B$, we can find the set W of k-1 centers of S that after adjoining with $\{x\}$ achieves the best k-medoids clustering on $(Z \cup B)$ in $\mathcal{O}(k)$ time. Specifically, for each center $s \in S$, we can store the number η_s of points assigned to s, as well as the distance d_s to the closest center in $S \setminus \{s\}$.

We can then update the information in time $\mathcal{O}(d(k+|Z|))$ for each query $\{p\}$ to add to S, to ultimately form Q. In particular, we find $u \in S$ that is the nearest center of S to p and update d_u to be $||u - p||_2$ if $||u - p||_2 < d_u$. By Lemma 3.7, the points served by the remaining centers all have the distances to their nearest centers preserved up to a 2^{z+1} -factor. Thus, we can approximate ψ in Algorithm 3 up to a 2^{z+1} -factor, along with the corresponding center ψ_p to be removed from S. By Lemma 3.5, the resulting set Q of k centers is a $2^{\mathcal{O}(z)}$ -approximation to the optimal set S of centers for (k, z)-medoids clustering that contains a center at x.

It remains to pre-process $(Z \cup B)$ and S to efficiently compute the number of centers n_b assigned to centers of Q in $B_{r/2}(x)$ for each $x \in B$ and each radius $r := ||x - p||_2$ for all $p \in (Z \cup B)$. To that end, we can first sort the points $y \in (Z \cup B)$ by their distances from x. Now we can scan radially outward from x, finding the closest center in $Z \cup B$ and iterating outward.

Given two centers $u, v \in (Z \cup B)$, let Q_u (resp. Q_v) be the clustering Q induced by removing the output of ϕ_u (resp. ϕ_v) from $S \cup \{u\}$ (resp. $S \cup \{v\}$) by Algorithm 3 with query center u (resp. v). In particular, suppose that after scanning radially outward from x, we first see u, followed immediately by v. Since $B_{r/2}(x)$ is monotonically increasing as r increases, it suffices to simply consider the additional centers in $Q_v \setminus Q_u$. Since Q_u and Q_v are both formed by swapping a center of S with $\{u\}$ and $\{v\}$ respectively, then we have $|Q_v \setminus Q_u| \leq 2$. Hence, to compute n_b for v, it suffices to use the previous computation of n_b for u and consider the number of points assigned to at most two centers of Q_u , which can be done in $\mathcal{O}(1)$ time.

Therefore, the total time to compute S and perform additional pre-processing is $\mathcal{O}(d(|Z|+k)^2)$. For each possible center $p \in (Z \cup B)$, we use $\mathcal{O}(k + |Z|)$ time to update the pre-processing information to obtain Ψ . For each point $x \in B$, we use $\mathcal{O}(dk \log k)$ time to sort the points by their distances from x. For each point $x \in B$ and possible center $p \in (Z \cup B)$, we use $\mathcal{O}(1)$ time to compute n_b . Hence, the amortized runtime is $\frac{d(|Z|+k \log k)^2}{k}$.

Algorithm 5 Fast (k, z)-clustering

Input: Set $X = \{x_1, ..., x_n\}$ of points in $[\Delta]^d$ that arrive as a stream **Output:** (k, z)-clustering coreset for X1: $S \leftarrow \emptyset, \lambda \leftarrow \mathcal{O}\left(\frac{kd}{\varepsilon^2}\log(n\Delta)\right)$ 2: Batch $x_1, ..., x_n$ into blocks $B_1, ..., B_{n/k}$ of k updates 3: **for** $b \in [n/k]$ **do**

- 4: Let Z be a coreset for block B_{b-1}
- 5: Call BATCHSENS to batch approximation of sensitivities for x_t with $t \in B_b$
- 6: Sample points of B_{b-1} into a stream \mathcal{S}' using sensitivity sampling
- 7: Update Z by running merge-and-reduce on \mathcal{S}'

Putting together Theorem 2.9 and Lemma 3.12, we have:

Theorem 3.13. Given a set X of n points on $[\Delta]^d$, let $f\left(n, d, \Delta, k, \frac{1}{\varepsilon}, z\right)$ be the number of points of a coreset construction with weights $[1, \operatorname{poly}(nd\Delta)]$ for (k, z)-clustering. There exists an algorithm that uses $dk \cdot \operatorname{polylog}(\log(nd\Delta))$ amortized update time and $\mathcal{O}(dk \log(n\Delta)) + f\left(n, d, \Delta, k, \frac{\operatorname{polylog}(\log(nd\Delta))}{\varepsilon}, z\right) + \operatorname{polylog}\left(\frac{1}{\varepsilon}, \log(nd\Delta)\right)$ bits of space, and outputs a $(1 + \varepsilon)$ -strong coreset of X.

3.3 Filtering of Low-Sensitivity Points

In this section, we show that our one-pass streaming algorithm can be implemented in $d \log k \cdot \text{polylog}(\log(n\Delta))$ amortized update time.

We first define a quadtree embedding. Due to the desiderata of fast update time, our construction is somewhat non-standard. Given $s = (s_1, \ldots, s_d) \in \mathbb{Z}^d$, $t \in \{0, 1, \ldots, \ell\}$, and a parameter z > 1, we define the axis-aligned grid $\mathcal{G}_{s,t,\zeta}$ over \mathbb{Z}^d with side length ζ^t , so that $s = (s_1, \ldots, s_d)$ lies on one of the corners of the grid. Then for $X \in \mathbb{R}^{[\Delta]^d}$, we define $G_{s,t,\zeta}(X)$ as the frequency vector over the hypercubes of the grid $\mathcal{G}_{s,t,\zeta}$ that counts the total number or weight of points in each hypercube. Specifically, each cell of a grid of length ζ^t has closed boundaries on one side and open boundaries on the other side, i.e., a cell containing (u_1, \ldots, u_d) cannot also contain (v_1, \ldots, v_d) for any $v_i \ge u_i + \zeta^t$. We define TreeDist_{s,\zeta}(x, y) to be $\sqrt{d} \cdot \zeta^{\alpha}$, where α is the smallest integer such that x and y are in two different cells of $\mathcal{G}_{s,\alpha,\zeta}$ or TreeDist_{s,\zeta}(x, y) = 0 if no such grid exists. For our purposes, we will set $\zeta = n^{1-c}$ for a fixed constant $c \in (0, 1)$ so that $\ell = \mathcal{O}\left(\frac{1}{c}\right)$ for $\Delta = \text{poly}(n)$.

We first show that the distance between a pair of points cannot be underestimated by the quadtree.

Lemma 3.14. For every $x, y \in [\Delta]^d$ and every $s = (s_1, \ldots, s_d) \in \mathbb{Z}^d$ and $\zeta > 1$, we have

$$dist(x,y) \leq \operatorname{TreeDist}_{s,\zeta}(x,y)$$

Proof. Since $x, y \in [\Delta]^d$, then by an averaging argument there exists some coordinate for which x and y are separated by distance $\frac{\|x-y\|_2}{\sqrt{d}}$. Then x and y must be separated in grid $\mathcal{G}_{s,\alpha,\zeta}$ for $\zeta^{\alpha} \leq \frac{\|x-y\|_2}{\sqrt{d}}$. Therefore, TreeDist_{s,\zeta} $(x, y) = \sqrt{d} \cdot \zeta^{\beta}$ for $\zeta^{\beta} > \frac{\|x-y\|_2}{\sqrt{d}}$. Then it follows dist $(x, y) \leq$ TreeDist_{s,\zeta}(x, y).

We next bound the probability the distance between a pair of points is overestimated by the quadtree for a fixed distortion rate.

Lemma 3.15. Given a set S and a parameter t > 2, let \mathcal{E} be the event where no points of S are within $\frac{1}{t}$ fraction toward the boundary of any cell in the grid that induces TreeDist. Then conditioned on \mathcal{E} , we have that for all $x, y \in S$,

TreeDist
$$(x, y) \le t \cdot \sqrt{d} \cdot \zeta \cdot dist(x, y).$$

Moreover, we have

$$\mathbf{Pr}\left[\mathcal{E}\right] \ge 1 - \frac{d|S|}{t}.$$

Proof. Observe that if x and y are first in the same cell at level i, then TreeDist_{s, ζ}(x, y) = $\sqrt{d} \cdot \zeta^i$. However, because the cells at level i - 1 have length ζ^{i-1} and conditioned on \mathcal{E} , both of x and y are at least $\frac{1}{t}$ fraction away from the boundary, then we have dist $(x, y) \geq \frac{\zeta^{i-1}}{t}$, which implies that

TreeDist
$$(x, y) \le t \cdot \sqrt{d} \cdot \zeta \cdot \operatorname{dist}(x, y)$$

Next, we note that for each coordinate, the probability that x is contained a distance that is within $\frac{1}{t}$ fraction toward the boundary is $\frac{1}{t}$. The desired claim then follows from a union bound over all $x \in S$ and all d dimensions.

We now require the following generalization of the fast Euclidean k-means approximation algorithm by [CLN⁺20] to (k, z)-clustering. We remark that the statement in [CLN⁺20] provides a $\mathcal{O}_{\varepsilon}(\log k)$ -approximation to k-means in runtime $\tilde{\mathcal{O}}(nd + (n \log \Delta)^{1+\varepsilon})$. We briefly describe the algorithm of [CLN⁺20] and the necessary adjustments to provide Theorem 3.16.

The algorithm of $[\text{CLN}^+20]$ first applies the Johnson-Lindenstrauss transformation [JL84] to all the points to reduce the dimension to $\mathcal{O}(\log k)$, which preserves all clustering costs within a constant factor [MMR19, ISZ21]. It then constructs a standard quadtree embedding, which it uses to adaptively sample a set C of k centers. Specifically, adaptive sampling randomly selects a point of the input set to be the first center of C. It then iterates, iteratively choosing k - 1 additional points from the input set, with probability proportional to the squared distance of the point from the current set of centers. To do this quickly, $[\text{CLN}^+20]$ uses multiple independent instances of the quadtree embedding to estimate the distance of each point from the current set of centers. Finally, to roughly estimate the clustering cost induced by C, $[\text{CLN}^+20]$ applies a locality-sensitivity hashing procedure to assign each point to its approximate nearest neighbor in C.

For (k, z)-clustering, we instead perform adaptive sampling by iteratively choosing each of the input points with probability proportional to the z-th power of distances of the point from the centers already opened in C. The $\mathcal{O}(\log \Delta)$ runtime of the algorithm of $[\text{CLN}^+20]$ is due to the $\mathcal{O}(\log \Delta)$ levels in the quadtree used to estimate these distances. To achieve runtime independent of $\mathcal{O}(\log \Delta)$, we instead use our crude quadtree with $\mathcal{O}(1)$ levels, to provide $n^{\Theta(1)}$ -approximations to these distances. Once C is acquired from adaptive sampling, we then estimate the clustering cost by again using the crude quadtree to assign each point to its closest center in the quadtree, which uses constant time per point, since the crude quadtree has $\mathcal{O}(1)$ levels. Finally, we remark that unlike $[\text{CLN}^+20]$, we can perform the quadtree embedding multiple times to avoid any cases where the distortion between pairs of points is too large. Thus we have:

Theorem 3.16. [CLN⁺ 20] For any constant $\alpha \in (0, 1)$ and $N \gg n$, there exists a N^{α}-approximation algorithm to (k, z)-clustering on X with n weighted points, that uses $O(nd \log(nd))$ expected runtime.

To estimate the sensitivity of a point x, we now enumerate over the distances induced by all levels of the quadtree for centers that serve x. To that end, we show that $\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}$ can only be an underestimate to the optimal (k, z)-medoids clustering constrained to β being the first level in which the center serving x is in the same cell as x in the quadtree. In particular, the following lemma should be considered the quadtree analog to Lemma 3.8 for the radial search.

Lemma 3.17. Let $x \in X$ be a fixed point and $p \in X$. Suppose TreeDist preserves all pairwise distances in X up to a factor of κ and suppose TreeDist $(p, x) = \sqrt{d} \cdot \zeta^{\beta}$. Let $\kappa \geq 2$ be a parameter, so that Q is a κ -factor approximation to the optimal (k, z)-medoids clustering on X containing a center at p and let Ψ be is a κ -factor approximation to $\operatorname{cost}(X, Q)$. Let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$, where $r := ||x - p||_2$. Let n_β be the weight of the points assigned to centers $q \in Q$ such that $\operatorname{TreeDist}(x, q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$. There exists a constant $\gamma \geq 1$ such that

$$\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z} \le \kappa^{\mathcal{O}(z)} \cdot \operatorname{cost}(X, S).$$

Proof. Since S is the optimal solution for a more constrained search space than Q, i.e., sets of at most k centers that contain a center at p and no center in the interior of $B_r(x)$, then we have

$$\kappa^2 \cdot \operatorname{cost}(X, S) \ge \kappa \cdot \operatorname{cost}(X, Q) \ge \Psi.$$

Now, consider a point y served by a center $q \in Q$ such that $\text{TreeDist}(x,q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$. If $\|y-q\|_2 \geq \frac{r}{4}$, then

$$||y - q||_2 + r \le 5 \cdot ||y - q||_2$$

Otherwise, for $||y - q||_2 < \frac{r}{4}$, then by triangle inequality, we have

$$dist(y, S) \ge dist(q, S) - ||y - q||_2 \ge \frac{3r}{4}.$$

Therefore,

$$||y - q||_2 + r \le ||y - q||_2 + 2 \cdot \operatorname{dist}(y, S)$$

Putting the two cases together, it follows that

$$||y - q||_2 + r \le 5 \cdot ||y - q||_2 + 2 \cdot \operatorname{dist}(y, S).$$

Summing across all $y \in X$ served by q such that $\text{TreeDist}(x,q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$,

$$\operatorname{cost}(X,Q) + n_{\beta} \cdot r^{z} \leq 5 \operatorname{cost}(X,Q) + 2 \operatorname{cost}(X,S) \leq (5\kappa + 2) \cdot \operatorname{cost}(X,S).$$

Since Ψ is a κ -approximation to $\operatorname{cost}(X, Q)$ and conditioned on the TreeDist procedure preserving all pairwise distances in X up to a factor of κ and thus $\sqrt{d} \cdot \zeta^{\beta}$ is a κ -approximation to r, then

$$\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z} \le \kappa^{\mathcal{O}(z)} \cdot \operatorname{cost}(X, S),$$

for $\kappa \geq 2$, i.e., bounded away from 1.

We next show that if there is a large number of points served by centers that are first in the same cell as x in a level of the quadtree before β , then $\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}$ is a good estimate to the cost of the optimal (k, z)-medoids clustering constrained to x being served at a center that is first in the same cell as x at level β in the quadtree. In particular, the following lemma should be considered the quadtree analog to Lemma 3.9 for the radial search.

Lemma 3.18. Let $x \in X$ be a fixed point and $p \in X$. Suppose TreeDist preserves all pairwise distances in X up to a factor of κ and suppose TreeDist $(p, x) = \sqrt{d} \cdot \zeta^{\beta}$. Let $\kappa \geq 2$ be a parameter, so that Q is a κ -factor approximation to the optimal (k, z)-medoids clustering on X containing a center at p and let Ψ be is a κ -factor approximation to $\operatorname{cost}(X, Q)$. Let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$, where $r := ||x - p||_2$. Let n_β be the weight of the points assigned to centers $q \in Q$ such that $\operatorname{TreeDist}(x, q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$ and let n_α be the weight of the points assigned to centers $q \in Q$ such that $\operatorname{TreeDist}(x, q) \geq \sqrt{d} \cdot \zeta^{\beta}$. If $n_\beta \geq n_\alpha$, then there exist constants $\gamma_1, \gamma_2 \geq 1$ such that

$$\operatorname{cost}(X,S) \le \kappa^{\mathcal{O}(\gamma_1 z)} \left(\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z} \right) \le \kappa^{\mathcal{O}(\gamma_1 \gamma_2 z)} \cdot \operatorname{cost}(X,S).$$

Proof. We first show that there exists a constant $\gamma_1 \geq 1$ such that $\cot(X, S) \leq \kappa^{\mathcal{O}(\gamma_1 z)} \left(\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}\right)$. Observe that this would imply by Lemma 3.17 that there also exists a constant $\gamma_2 \geq 1$ such that $\kappa^{\mathcal{O}(\gamma_1 z)} \left(\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}\right) \leq \kappa^{\mathcal{O}(\gamma_1 \gamma_2 z)} \cdot \cot(X, S)$.

Consider a point y served by a center $q \in Q$ with $\text{TreeDist}(x,q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$. Let $\pi_S(y)$ be the closest center in S to y, breaking ties arbitrarily. By generalized triangle inequality,

$$\cot(y, S) \le 2^z \cdot ||y - q||_2^z + 2^z \cdot \cot(q, \pi_S(y)).$$

Summing across all y served by centers q with TreeDist $(x,q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$ and noting that dist $(s,q) \leq 2\sqrt{d} \cdot \zeta^{\beta}$ conditioned on the correctness of TreeDist, as well as and noting that $n_{\beta} \geq n_{\alpha}$, we have

$$\sum_{y: \text{TreeDist}(x, \pi_Q(y)) \le \sqrt{d} \cdot \zeta^{\beta-1}} \cot(y, S) \le 2^z \cdot n_\beta \cdot \left(2\sqrt{d} \cdot \zeta^{\beta}\right)^z + 2^z \cdot \sum_{y: \text{TreeDist}(x, \pi_Q(y)) \le \sqrt{d} \cdot \zeta^{\beta-1}} \cot(y, Q).$$

Next, consider a point y served by a center $q \in Q$ with $\text{TreeDist}(x,q) \geq \sqrt{d} \cdot \zeta^{\beta}$. We have that if $\|y - q\|_2 \geq \frac{r}{4}$, then $\|y - p\|_2 \leq \|y - q\|_2 + \|q - p\|_2 \leq 9 \cdot \|y - q\|_2$. Else if $\|y - q\|_2 < \frac{r}{4}$, then $\|y - p\|_2 \leq \|y - q\|_2 + \|q - p\|_2 < 3r$. Summing across all y served by centers $q \in Q$ with $\text{TreeDist}(x,q) \geq \sqrt{d} \cdot \zeta^{\beta}$,

$$\sum_{y: \text{TreeDist}(x, \pi_Q(y)) \ge \sqrt{d} \cdot \zeta^{\beta}} \cot(y, S) \le 12^z \cdot n_{\beta} \cdot \left(\frac{r}{2}\right)^z + 9 \cdot 2^z \cdot \sum_{y: \text{TreeDist}(x, \pi_Q(y)) \ge \sqrt{d} \cdot \zeta^{\beta}} \cot(y, Q).$$

Conditioned on the correctness of TreeDist, we have that $\sqrt{d} \cdot \zeta^{\beta}$ is a κ -approximation to r. Thus summing the two cases across all $y \in X$, we have that there exists a constant $\zeta \geq 1$ such that

$$\operatorname{cost}(X,S) \le \kappa^{\zeta z} \cdot \left(\operatorname{cost}(X,Q) + n_{\beta} \cdot \left(\sqrt{d} \cdot \zeta^{\beta} \right)^{z} \right).$$

Since Φ is a κ -factor approximation to cost(X, Q), then there exists a constant γ_1 such that

$$\operatorname{cost}(X,S) \le \kappa^{\mathcal{O}(\gamma_1 z)} \left(\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z} \right).$$

Thus,

$$\operatorname{cost}(X,S) \le \kappa^{\mathcal{O}(\gamma_1 z)} \left(\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z} \right) \le \kappa^{\mathcal{O}(\gamma_1 \gamma_2 z)} \cdot \operatorname{cost}(X,S).$$

On the other hand, if the number of points served by centers that are first in the same cell as x in a level of the quadtree before β is small, then $\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}$ may not be a good estimate to the cost of the optimal (k, z)-medoids clustering constrained to x being served at a center that is first in the same cell as x at level β in the quadtree. However, we next show that it can nevertheless provide an upper bound on the $\frac{\cos(x,S)}{\cot(X,S)}$ for any S where x is served by a center that is first in the same cell as x at level β in the quadtree. Importantly, the quantity is at most the sensitivity s(x) of x. In particular, the following lemma should be considered the quadtree analog to Lemma 3.10 for the radial search.

Lemma 3.19. Let $x \in X$ be a fixed point and $p \in X$. Suppose TreeDist preserves all pairwise distances in X up to a factor of κ and suppose TreeDist $(p, x) = \sqrt{d} \cdot \zeta^{\beta}$. Let s(x) be the sensitivity of (k, z)-medoids clustering with respect to X. Let $\kappa \geq 2$ be a parameter, so that Q is a κ -factor approximation to the optimal (k, z)-medoids clustering on X containing a center at p and let Ψ be is a κ -factor approximation to cost(X, Q). Let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$, where $r := ||x - p||_2$. Let n_β be the weight of the points assigned to centers $q \in Q$ such that TreeDist $(x, q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$ and let n_α be the weight of the points assigned to centers $q \in Q$ such that TreeDist $(x, q) \geq \sqrt{d} \cdot \zeta^{\beta}$. If $n_b < n_a$, then

$$\frac{1}{\kappa^{\mathcal{O}(\gamma_1 z)}} \frac{\cot(x, S)}{\cot(X, S)} \le \frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z}} \le \kappa^{\mathcal{O}(\gamma_2 z)} \cdot s(x),$$

for fixed constants $\gamma_1, \gamma_2 \geq 1$.

Proof. By Lemma 3.17, there exists a constant $\gamma \geq 1$ such that

$$\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z} \le \kappa^{\gamma z} \cdot \operatorname{cost}(X, S).$$

Since S contains a center at p and no centers in the interior of $B_r(x)$, then $cost(x, S) = r^z$. Then conditioned on the correctness of TreeDist giving $d^{z/2} \cdot \zeta^{\beta z}$ as a κ^z -approximation to r^z , we have that

$$\frac{1}{\kappa^{\mathcal{O}(\gamma_1 z)}} \frac{\operatorname{cost}(x, S)}{\operatorname{cost}(X, S)} \le \frac{r^z}{\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z}}.$$

However, for a set W of k centers that contains a center at p but no center q with TreeDist(x,q) < $\sqrt{d} \cdot \zeta^{\beta-1}$, then we have $\frac{\sqrt{d} \cdot \zeta^{\beta-1}}{\kappa} \leq \operatorname{dist}(x, W)$ and $\operatorname{cost}(X, W) \leq \kappa^{\gamma_2 z} (\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z})$ for some constant $\gamma_2 \geq 1$. Thus by the maximality of sensitivity, we have

$$\frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_b \cdot r^z} \le \kappa^{\gamma_2 z} \cdot \frac{\operatorname{cost}(x, W)}{\operatorname{cost}(X, w)} \le \kappa^{\gamma_2 z} \cdot s(x).$$

Hence if TreeDist provides a κ -approximation to the pairwise distance of all points $x \in X$, then the algorithm in ROUGHSENS provides a $\kappa^{\mathcal{O}(1)}$ -factor approximations to all sensitivities x in the batch B.

Lemma 3.20. For a fixed $x \in X$, let s(x) be the sensitivity of (k, z)-medoids clustering with respect to X. Suppose TreeDist preserves all pairwise distances in X up to a factor of κ . Let s(x) be the output of Algorithm 6, for each $x \in B$, where B is the input batch. Then there exist constants $\gamma_1, \gamma_2 \geq 1$ such that

$$\frac{1}{\kappa^{\gamma_1 z}} \cdot s(x) \le \widehat{s(x)} \le \kappa^{\gamma_2 z} \cdot s(x).$$

Proof. For each fixed $p \in X$, let S be the optimal (k, z)-medoids clustering on X containing a center at p and no center in the interior of $B_r(x)$. Let n_β be the weight of the points assigned to centers $q \in Q$ such that TreeDist $(x,q) \leq \sqrt{d} \cdot \zeta^{\beta-1}$ and let n_{α} be the weight of the points assigned to centers $q \in Q$ such that TreeDist $(x, q) \ge \sqrt{d} \cdot \zeta^{\beta}$. We perform casework on whether $n_{\beta} < n_{\alpha}$ or $n_{\beta} \ge n_{\alpha}$.

In the case where $n_{\beta} \ge n_{\beta}$, then by Lemma 3.18,

$$\operatorname{cost}(X,S) \le \kappa^{\mathcal{O}(\gamma_1 z)} \left(\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z} \right) \le \kappa^{\mathcal{O}(\gamma_1 \gamma_2 z)} \cdot \operatorname{cost}(X,S).$$

Since $cost(x, S) = r^z$, which is a κ^z -approximation to $(d^{z/2} \cdot \zeta^{\beta z})$ conditioned on the correctness of TreeDist, then it holds that $\frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}}$ is a $\kappa^{\mathcal{O}(z)}$ approximation to $\frac{\operatorname{cost}(x,S)}{\operatorname{cost}(X,S)}$. Otherwise if $n_{\beta} < n_{\alpha}$, then by Lemma 3.19,

$$\frac{1}{\kappa^{\mathcal{O}(\gamma_1 z)}} \frac{\operatorname{cost}(x, S)}{\operatorname{cost}(X, S)} \le \frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_\beta \cdot d^{z/2} \cdot \zeta^{\beta z}} \le \kappa^{\mathcal{O}(\gamma_2 z)} \cdot s(x),$$

for fixed constants $\gamma_1, \gamma_2 \ge 1$. In particular, for the center $p \in X$ that realizes the sensitivity s(x), we have that $\frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}}$ is a $\kappa^{\mathcal{O}(z)}$ approximation to $\frac{\cot(x,S)}{\cot(X,S)}$. Furthermore, for centers $q \in X$ that do not realize the sensitivity, we have that $\frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_{\beta} \cdot d^{z/2} \cdot \zeta^{\beta z}} \leq \kappa^{\mathcal{O}(\gamma_2 z)} \cdot s(x)$. Thus,

$$\frac{1}{\kappa^{\gamma_1 z}} \cdot s(x) \le \widehat{s(x)} \le \kappa^{\gamma_2 z} \cdot s(x).$$

Algorithm 6 ROUGHSENS: Rough batch approximation of sensitivities

- **Input:** Input set $Z \subset [\Delta]^d$ for (k, z)-medoids clustering, batch B of k query points, approximation parameter $\alpha \in (0, 1)$
- **Output:** n^{α} approximation for the sensitivity of x for (k, z)-medoids clustering on $Z \cup B$, for all $x \in B$
- 1: $\kappa \leftarrow n^{\alpha}, \zeta \leftarrow n^{\iota}$ for sufficiently small $\iota < \alpha \in (0, 1)$
- 2: Compute S to be a constant-factor solution on $Z \cup B$ for (k, z)-medoids clustering
- 3: for $x \in B$ do
- $s(x) \leftarrow 0$ 4:
- Build a tree-embedding on $Z \cup B$ 5:
- while there is a point of $Z \cup B$ within $\frac{1}{\kappa}$ -fraction of the distance to a cell in the tree 6: embedding **do**
- Create a tree-embedding on $Z \cup B$ 7:
- 8: for each level ℓ_{β} in the tree do
- Let Q be a κ -factor approximation Q of the optimal (k, z)-medoids clustering on X 9: initially separated from x at level ℓ_{β}
- Use TreeDist to compute a constant-factor approximation Ψ of cost(X, Q)10:
- Let n_{β} be the number of points served by centers of Q in the same cell as x before ℓ_{β} 11: $\left(\widehat{s(x)}, \frac{d^{z/2} \cdot \zeta^{\beta z}}{\Psi + n_{\beta} \cdot (d^{z/2} \cdot \zeta^{\beta z})}\right)$

12:
$$\widehat{s(x)} \leftarrow \max$$

return s(x)13:

We now analyze the properties of ROUGHSENS. Since correctness is mostly given by Lemma 3.20, the main consideration is the amortized update time.

Lemma 3.21. Given any constant $\alpha \in (0,1)$ and a constant-factor coreset Z of size $k \cdot \text{polylog}(\log n)$ to $X \subset [\Delta]^d$, there exists an algorithm ROUGHSENS that outputs n^{α} -approximations to the sensitivities to each of the points $x \in B$ in a batch B of k points of X, using amortized $d\log(k)$. polylog(log($n\Delta$)) time, with probability at least $1 - \frac{1}{\sqrt{n}}$.

Proof. By Lemma 3.3, we can compute the sensitivity of $x \in Z \cup B$ as a good approximation to computing the $x \in B$ with respect to X, since Z is a constant-factor coreset to Y so that $Z \cup B$ is a constant-factor coreset to $X = Y \cup B$. Moreover, it suffices to compute a constant-factor approximation to the (k, z)-medoids clustering sensitivity of x with respect to $Z \cup B$, by Theorem 3.2. Therefore, the goal is to compute a crude approximation to $\max_{C \subset Z \cup B: |C| \leq k} \frac{\operatorname{cost}(x,C)}{\operatorname{cost}(X,C)}$. Let \mathcal{E} be

the event that TreeDist approximates all pairwise distances within a factor of $\frac{1}{\text{poly}(d)} \cdot n^{\frac{1}{100p}}$. Then conditioned on \mathcal{E} , correctness follows by Lemma 3.20. We remark that the algorithm iterates until \mathcal{E} occurs, so the dependency on \mathcal{E} is not in the correctness, but rather than runtime.

Thus, we now analyze the amortized runtime of ROUGHSENS. By Theorem 3.16, we use $kd \log(kd) \cdot \operatorname{polylog}(\log(n\Delta))$ time to find a κ -approximation S of k centers that serve as a constantfactor approximation to $Z \cup B$. For $\Delta = \text{poly}(n)$, the tree has height $\mathcal{O}\left(\frac{1}{t}\right)$ and thus by pre-processing the number of points in each cell, the total time form Q across all levels is $\mathcal{O}\left(\frac{1}{\iota}\right)$. Similarly, we can approximate Ψ up to a κ -factor, along with the corresponding center ψ_{β} to be removed from S that is consistent with Ψ .

It remains to pre-process $(Z \cup B)$ and S to efficiently compute the number of centers n_{β} assigned to centers of $q \in Q$ in such that $\text{TreeDist}(x,q) < \sqrt{d} \cdot \zeta^{\beta}$. Again, this is handled by pre-processing all the points $y \in (Z \cup B)$ in the quadtree by pre-computing the number of points in each cell at each level, which takes total time $k \cdot \text{polylog}(\log n)$ due to the size of X. Then by starting at $\beta = 0$ and iterating to $\beta = \mathcal{O}\left(\frac{1}{\iota}\right)$, we proceed up the tree to compute each n_{β} , using $\mathcal{O}\left(\frac{1}{\iota}\right)$ time in total. Therefore, the total time to compute S and perform additional pre-processing is $kd \log(kd) \cdot$

Therefore, the total time to compute S and perform additional pre-processing is $kd \log(kd) \cdot polylog(\log(n\Delta))$. Across all possible levels β , we use $\mathcal{O}\left(\frac{1}{\iota}\right)$ time to update the pre-processing information to obtain Ψ and compute each value of n_{β} . Hence, the amortized runtime is $d \log(kd) \cdot polylog(\log(n\Delta))$. Finally, we remark that if $d \geq \mathcal{O}(\log n)$, then we can apply the standard Johnson-Lindenstrauss dimensionality reduction technique to achieve dimension $\mathcal{O}(\log n)$, and thus the resulting amortized runtime is $d \log(k) \cdot polylog(\log(n\Delta))$.

Putting everything together, we have:

Theorem 3.22. Given a set X of n points on $[\Delta]^d$, let $f(n, d, \Delta, k, \frac{1}{\varepsilon}, z)$ be the number of points of a coreset construction with weights $[1, \text{poly}(nd\Delta)]$ for (k, z)-clustering. There exists an algorithm that uses $d\log(k) \cdot \text{polylog}(\log(n\Delta))$ amortized update time and $\mathcal{O}(dk\log(n\Delta)) + f(n, d, \Delta, k, \frac{\text{polylog}(\log(nd\Delta))}{\varepsilon}, z) \cdot \text{polylog}(\frac{1}{\varepsilon}, \log(nd\Delta))$ bits of space, and outputs a $(1 + \varepsilon)$ -strong coreset of X.

Proof. We perform induction on the time t throughout the stream. Correctness at the first step is immediate, since the first point is inserted into the batch. Now we suppose for our inductive hypothesis that at time t - 1, we have both a constant-factor coreset and a $(1 + \varepsilon)$ -coreset at time t - 1.

Consider time t for our inductive step. Either the new point is added to the batch of size k, in which case both coreset invariants are maintained, or we perform a sampling step on the batching using crude approximations to the sensitivities as an initial filtering step, because the batch has gotten too large. In the latter case, we first use the constant-factor coreset through Lemma 3.21 to compute rough approximations to the (k, z)-sensitivities in a batch of k points. This uses $d \log(k) \cdot \text{polylog}(\log n)$ amortized runtime and produces an insertion-only stream S of length $\mathcal{O}\left(n^{1-\Omega(1)}\right)$. We feed this insertion-only stream S as the input to Algorithm 5. By Theorem 3.13, the algorithm uses dk polylog $(\log(n\Delta))$ update time on the stream S of length o(n) to generate both a constant-factor coreset and a $(1 + \varepsilon)$ -factor coreset at time t, which completes our induction. Since this is a lower-order term, then the total amortized runtime is $d \log(k) \cdot \text{polylog}(\log n)$.

Finally, we note the implications of Theorem 3.22 to dynamic (k, z)-clustering in the incremental setting.

Theorem 3.23. Given an insertion-only stream of n points that defines a dataset X on $[\Delta]^d$, there exists a one-pass streaming algorithm that uses $d \log(k) \cdot \operatorname{polylog}(\log(n\Delta))$ amortized update time and $\tilde{\mathcal{O}}(dk \log(n\Delta))$ bits of space, and outputs a $\mathcal{O}(z)$ -approximation to (k, z)-clustering at all times in the stream.

Proof. By Theorem 1.6, there exists a coreset construction for Euclidean (k, z)-clustering that samples $\tilde{\mathcal{O}}(k)$ weighted points of the input dataset. Thus, Theorem 3.22 guarantees a constant-approximation coreset for X using $d \log(k) \cdot \operatorname{polylog}(\log(n\Delta))$ amortized update time. In fact, because

the proof of Theorem 3.22 uses induction over the course of the stream, it actually achieves a constantapproximation coreset to every prefix of the stream. We can thus achieve a $\mathcal{O}(z)$ -approximation at all times in the stream by applying a standard polynomial-time clustering approximation algorithm such as local search, c.f., Theorem 3.4, each time the merge-and-reduce data structure is updated. Moreover, note that by Theorem 2.4, the final input stream induced by online sensitivity sampling to the merge-and-reduce data structure has length $\mathcal{O}\left(dk^2\log^3(n\Delta)\right)$. By applying a standard Johnson-Lindenstrauss transformation, c.f., Theorem 1.7, we can view $d = \mathcal{O}(\log n)$. Therefore, the total runtime of all iterations of local search is $\mathcal{O}\left(k^2\log^4(n\Delta)\right) \cdot \tilde{\mathcal{O}}\left(k^2\log n\right)$. Thus for $n \gg k^4$, the amortized runtime remains $d\log(k) \cdot \operatorname{polylog}(\log(n\Delta))$, due to the computation of the constantapproximation coreset.

4 Subspace Embeddings

In this section, we describe our one-pass streaming algorithm on insertion-only streams for L_p subspace embeddings that uses $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space and amortized runtime $\mathcal{O}(d)$ per update. Recall that in problem of L_p subspace embeddings, the rows of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ arrive one-by-one, and the goal is to produce a matrix $\mathbf{M} \in \mathbb{R}^{m \times d}$ such that for a fixed $\varepsilon \in (0, 1)$ and for all $\mathbf{x} \in \mathbb{R}^d$, we have

$$(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_p \le \|\mathbf{M}\mathbf{x}\|_p \le (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_p.$$

 L_p subspace embeddings are widely used in data compression applications where preserving the geometry of data is critical, e.g., dimensionality reduction, compressed sensing, and linear/robust regression. Hence, there is a long line of active work studying L_p subspace embeddings [DMM06a, DMM06b, Sar06, DDH⁺09, CW09, CP15, CMP20, BDM⁺20, PPP21, MWZ22, MMM⁺22, MMWY22, CSWZ23, MMM⁺23, WY23].

We first describe an efficient encoding for coresets for L_p embeddings, as well as a global encoding that be utilized to achieve our streaming algorithm that uses $\mathcal{O}_{k,d,\varepsilon}(1)$ words of space. We then describe how our algorithm can be implemented using $\mathcal{O}(d)$ amortized update time.

We first recall the following definition of leverage scores, which intuitively quantifies the importance of a row for L_2 subspace embeddings.

Definition 4.1 (Leverage scores). The leverage score of a row $\mathbf{a}_i \in \mathbb{R}^d$ of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ is

$$\max_{\mathbf{x}:\|\mathbf{A}\mathbf{x}\|_2=1} \langle \mathbf{a}, \mathbf{x} \rangle^2$$

It is known that the leverage score for \mathbf{a}_i also admits the closed form $\mathbf{a}_i^{\top} (\mathbf{A}^{\top} \mathbf{A})^{-1} \mathbf{a}_i$.

We first recall one of the possible generalizations of leverage scores to L_p subspace embeddings.

Definition 4.2 (L_p Lewis weights). The Lewis weight of a row $\mathbf{a}_i \in \mathbb{R}^d$ of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ for L_p subspace embedding is the *i*-th diagonal entry of the unique diagonal matrix \mathbf{W} such that for all $i \in [n]$

$$W_{i,i} = \ell_i(\mathbf{W}^{\frac{1}{2} - \frac{1}{p}}\mathbf{A}),$$

where ℓ_i denotes the *i*-th leverage score, *i.e.*, the leverage score of the *i*-th row of the matrix $\mathbf{W}^{\frac{1}{2}-\frac{1}{p}}\mathbf{A}$.

The intuition for L_p Lewis weights is not obvious; they are the leverage scores of the matrix after applying a proper change-of-density, where each row is weighted the number of times necessary to preserve $\|\mathbf{A}\mathbf{x}\|_p$ when consider $\|\mathbf{A}'\mathbf{x}\|_2$, where \mathbf{A}' is the reweighted matrix [CP15]. An advantage of using Lewis weights to sample rows over other quantities such as the L_p leverage scores [DDH⁺09] or the L_p sensitivities [BDM⁺20, CD21] is that the L_p Lewis weights are known to admit the optimal sampling complexity. On the other hand, for the special case of p = 2, [Sar06] showed the existence of an oblivious subspace embedding with $\mathcal{O}\left(\frac{d}{\varepsilon^2}\right)$ rows. Combining these methods, we have:

Theorem 4.3. [Sar06, CP15, WY23] Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, accuracy $\varepsilon \in (0, 1)$, and $p \ge 1$, let

$$f\left(d,\frac{1}{\varepsilon},p\right) = \begin{cases} \frac{d}{\varepsilon^2}\log\frac{d}{\varepsilon}\operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1,2)\\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2\\ \frac{d^{p/2}}{\varepsilon^2} \cdot \operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2 \end{cases}$$

Then there exists a coreset construction for L_p subspace embeddings that consists of $f\left(d, \frac{1}{\varepsilon}, p\right)$ rows, each with entry at most $\|\mathbf{A}\|_{\infty} \cdot \operatorname{poly}(n)$, which can be constructed in time polynomial in n and d.

We next define the generalization of L_p Lewis weights to the online setting.

Definition 4.4 (Online L_p Lewis weights). The online L_p Lewis weight of a row $\mathbf{a}_i \in \mathbb{R}^d$ of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ is the Lewis weight of \mathbf{a}_i with respect to the matrix $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$, i.e., the submatrix of \mathbf{A} consisting of the first *i* rows of \mathbf{A} .

We recall the following upper bounds on the sum of the online Lewis weights.

Theorem 4.5. [BDM⁺20, WY23] Let $\mathbf{A} = {\mathbf{a}_1, \ldots, \mathbf{a}_n} \in \mathbb{R}^{n \times d}$ be a matrix of n rows and let $\sigma(\mathbf{a}_t)$ denote the online Lewis weight of \mathbf{a}_t for $t \in [n]$ for L_p -subspace embedding, where $p \ge 1$. Then for $p \in [1, 2]$,

$$\sum_{t=1}^{n} \sigma(\mathbf{a}_t) = d \cdot \operatorname{polylog}(n\kappa),$$

and for p > 2,

$$\sum_{t=1}^{n} \sigma(\mathbf{a}_t) = d^{p/2} \cdot \operatorname{polylog}(n\kappa),$$

where κ is the online condition number.

We now recall the full guarantees of online Lewis weight sampling.

Theorem 4.6 (Online Lewis weight sampling). [WY23] Given a sequence $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^{n \times d}$ of rows, suppose each point \mathbf{a}_t is sampled with probability $p_t \geq \min(1, \gamma \cdot \sigma(\mathbf{a}_t))$, where $\sigma(\mathbf{a}_t)$ is the online Lewis weight of \mathbf{a}_t and $\gamma = \mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ and multiplied by $\frac{1}{p_t}$ if \mathbf{a}_t is sampled. Then with high probability, the rescaled sample is a $(1 + \varepsilon)$ -strong coreset for L_p -subspace embedding that has a number of rows that is at most:

$$g\left(n,d,\frac{1}{\varepsilon},\kappa,p\right) = \begin{cases} \frac{d}{\varepsilon^2} \cdot \operatorname{polylog}(n\kappa), & p \in [1,2]\\ \frac{d^{p/2}}{\varepsilon^2} \cdot \operatorname{poly}(\log^{p/2}(n\kappa)), & p > 2 \end{cases}$$

We use standard approaches, e.g., Lemma 4.1 in [CW09] to upper bound the magnitude of the logarithm of any nonzero singular value of a matrix with bounded integer entries. This in turn upper bounds the logarithm of the online condition number.

Lemma 4.7. Suppose $\mathbf{A} \in \mathbb{R}^{n \times d}$ has integer entries bounded in magnitude by M. Then for any nonzero singular value σ_i of \mathbf{A} , we have $|\log \sigma_i| = \mathcal{O}(d \log(nM))$.

Proof. We have that the characteristic polynomial of $\mathbf{A}^{\top}\mathbf{A}$ is $p(x) = \det(x \cdot \mathbb{I}_d - \mathbf{A}^{\top}\mathbf{A})$. Note that if \mathbf{A} has rank r, then $p(x) = x^{d-r}\prod_{i=1}^{r}(x-\lambda_i)$, where λ_i is the *i*-th eigenvalue of $\mathbf{A}^{\top}\mathbf{A}$, writing $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$. Since all entries of $\mathbf{A}^{\top}\mathbf{A}$ are integers, then the coefficients of p(x) are also integers. Moreover, since the eigenvalues of $\mathbf{A}^{\top}\mathbf{A}$ are non-negative, then $\prod_{i=1}^{r}\lambda_i \geq 1$. Because all entries of $\mathbf{A}^{\top}\mathbf{A}$ are at most poly(ndM) in magnitude, then

$$\lambda_i \leq \lambda_1 \leq \|\mathbf{A}^\top \mathbf{A}\|_F \leq \text{poly}(ndM).$$

Since $\prod_{i=1}^{r} \lambda_i \ge 1$, then we have that $\lambda_i \ge \frac{1}{\operatorname{poly}(ndM)^d}$. Since $\sigma_i^2 = \lambda_i$, then $|\log \sigma_i| = \mathcal{O}(d\log(nM))$.

We next recall the following definition of a well-conditioned basis.

Definition 4.8 (Well-conditioned basis). Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ of rank r, let $p \in [1, \infty)$ and q be its dual norm, so that $\frac{1}{p} + \frac{1}{q} = 1$. Then matrix $\mathbf{U} \in \mathbb{R}^{n \times d}$ is an (α, β, p) -well-conditioned basis for the column space of \mathbf{A} if:

- (1) The column space of \mathbf{A} is the column space of \mathbf{U}
- (2) $\sum_{i \in [n], j \in [d]} U_{i,j}^p \leq \alpha^p$
- (3) For all $\mathbf{x} \in \mathbb{R}^d$, we have $\|\mathbf{z}\|_q \leq \beta \|\mathbf{U}\mathbf{z}\|_p$

One such construction of a well-conditioned basis gives the following properties:

Theorem 4.9. [DDH⁺09] Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ of rank r, let $p \in [1, \infty)$ and q be its dual norm, so that $\frac{1}{p} + \frac{1}{q} = 1$. There exists an (α, β, p) -well-conditioned basis \mathbf{U} for the column space of \mathbf{A} such that:

- If p < 2, then $\alpha = r^{\frac{1}{p} + \frac{1}{2}}$ and $\beta = 1$
- If p = 2, then $\alpha = \sqrt{r}$ and $\beta = 1$
- If p > 2, then $\alpha = r^{\frac{1}{p} + \frac{1}{2}}$ and $\beta = r^{\frac{1}{q} \frac{1}{2}}$

Moreover, **U** can be computed in time $\mathcal{O}(ndr + nr^2 \log n)$.

For completeness, we briefly describe the construction of the well-conditioned basis of Theorem 4.9, given by $[DDH^+09]$. Given a QR decomposition $\mathbf{A} = \mathbf{QR}$, so that \mathbf{Q} is any $n \times r$ matrix that is an orthonormal basis for the span of \mathbf{A} and \mathbf{R} is an $r \times d$ matrix, define the set $S = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{Qx}\|_p \leq 1\}$ and the $r \times r$ matrix \mathbf{F} so that $\mathcal{E}_{LJ} = \{\mathbf{x} \in \mathbb{R}^r \mid \mathbf{x}^\top \mathbf{Fx} \leq 1\}$ is the Löwner-John ellipsoid of S, let $\mathbf{G} \in \mathbb{R}^{d \times r}$ be the full rank and upper triangular matrix such that $\mathbf{F} = \mathbf{G}^\top \mathbf{G}$. Then $\mathbf{U} := \mathbf{QG}^{-1}$ is the desired (α, β, p) -well-conditioned basis for the column span of \mathbf{A} .

We now define the L_p sensitivity of a row of a matrix, for the purposes of L_p subspace embeddings.

Definition 4.10 (L_p sensitivity). For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, the L_p sensitivity of row \mathbf{a}_t is the quantity

$$\max_{\mathbf{y} \in \mathbb{R}^d: \|\mathbf{A}\mathbf{y}\|_p = 1} |\langle \mathbf{a}_t, \mathbf{y} \rangle|^p$$

Finally, we recall the following upper bound on the total L_p sensitivity for L_p subspace embedding.

Lemma 4.11 (Upper bounds on the sum of the L_p sensitivities). [CP15, CD21] For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, let $s(\mathbf{a}_t)$ denote the L_p sensitivity of the t-th row of \mathbf{A} for L_p subspace embedding. Then $\sum_{t=1}^n s(\mathbf{a}_t) \leq d$ for $p \leq 2$ and $\sum_{t=1}^n s(\mathbf{a}_t) \leq d^{p/2}$ for p > 2.

Efficient Encoding for Coreset Construction for L_p Subspace Embedding 4.1

We now give our efficient encoding for a given coreset for L_p subspace embeddings. Given a matrix **A**, which can be viewed as either the original matrix or a set of reweighted rows that forms a coreset of some underlying matrix, we first acquire a constant-factor approximation \mathbf{M} for L_p subspace embedding on A. We then use the existence of a well-conditioned basis to compute a deterministic preconditioner $\mathbf{P} \in \mathbb{R}^{d \times d}$, so that \mathbf{MP}^{-1} has condition number $\operatorname{poly}(d)$. Now for each row \mathbf{a}_t of \mathbf{A} , let \mathbf{b}'_t be $\mathbf{a}_t \mathbf{P}$ with each coordinate rounded to a power of $(1 + \varepsilon')$ for $\varepsilon' = \frac{\operatorname{poly}(\varepsilon)}{\operatorname{poly}(d^p, \log(nd))}$. We then store the exponent of each rounded coordinate of \mathbf{B}' , as well as \mathbf{M} . The algorithm appears in full in Algorithm 7.

Algorithm 7 Efficient Encoding for Coreset Construction for L_p Subspace Embedding

Input: Matrix $\mathbf{A} \subset \{-M, \ldots, -1, 0, 1, \ldots, M\}^{n \times d}$, accuracy parameter $\varepsilon \in (0, 1)$, failure probability $\delta \in (0,1)$

Output: $(1 + \varepsilon)$ -coreset for L_p subspace embedding

- 1: $\varepsilon' \leftarrow \frac{\text{poly}(\varepsilon)}{\text{poly}(d^p,\log(nd\Delta))}$ 2: Find a matrix **M** that is a constant-factor L_p subspace embedding on **A** ⊳Theorem 4.3
- 3: Compute a deterministic preconditioner \mathbf{P} of \mathbf{M} using well-conditioned bases, such that \mathbf{MP}^{-1} has condition number poly(d)
- 4: for each row $\mathbf{a} \in \mathbf{A}$ do
- Let **b'** be **aP** with each coordinate rounded to a power of $(1 + \varepsilon')$ 5:
- $\mathbf{B}' \leftarrow \mathbf{B}' \circ \mathbf{b}'$, storing the *exponent* for each entry of \mathbf{b}' 6:
- 7: return $(\mathbf{M}, \mathbf{B}')$, from which estimate \mathbf{A}' can be constructed

Given a constant-factor subspace embedding \mathbf{M} , we now show that the matrix \mathbf{B}' acquired by rounding each row of AP can be used to construct a matrix A' that is a strong coreset of A.

Lemma 4.12. Let $\varepsilon \in (0, \frac{1}{2})$ and let $\mathbf{A}' = \mathbf{B}'\mathbf{P}^{-1}$, where \mathbf{B}' is the output of Algorithm 7 and \mathbf{P} is the deterministic preconditioner of **M**. Then for all $\mathbf{x} \in \mathbb{R}^d$,

$$(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_p^p \le \|\mathbf{A}'\mathbf{x}\|_p \le (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_p^p.$$

Proof. Without loss of generality, we assume that A is full rank. Note that we can write Ax = $APP^{-1}x$ and thus by rewriting $y = P^{-1}x$, it suffices to show that for all $y \in \mathbb{R}^d$

$$(1-\varepsilon) \|\mathbf{A}\mathbf{P}\mathbf{y}\|_p^p \le \|\mathbf{A}'\mathbf{P}\mathbf{y}\|_p^p \le (1+\varepsilon) \|\mathbf{A}\mathbf{P}\mathbf{y}\|_p^p.$$

We remark that if **A** is not full rank, the task would be to show claim for all **y** in the column span of \mathbf{P}^{-1} .

Let $\mathbf{B} = \mathbf{AP}$ and observe that Algorithm 7 rounds all coordinates of each row $\mathbf{b}_t = \mathbf{a}_t \mathbf{P}$ to their closest power of $(1 + \varepsilon')$. Then we have

$$\left| \langle \mathbf{b}_t, \mathbf{y} \rangle^p - \langle \mathbf{b}_t', \mathbf{y} \rangle^p \right| \le \varepsilon' \cdot \max_{\mathbf{y} \in \mathbb{R}^d, \|\mathbf{y}\|_p = 1} |\langle \mathbf{b}_t, \mathbf{y} \rangle|^p.$$

For each $t \in [n]$, let $s(\mathbf{b}_t)$ denote the sensitivity of \mathbf{b}_t with respect to **B**, so that

$$s(\mathbf{b}_t) = \max_{\mathbf{y} \in \mathbb{R}^d} \frac{|\langle \mathbf{b}_t, \mathbf{y} \rangle|^p}{\|\mathbf{B}\mathbf{y}\|_p^p}.$$

Recall that $\mathbf{B} = \mathbf{AP}$ and \mathbf{P} is a preconditioner for \mathbf{M} , so that \mathbf{MP} is well-conditioned. That is, for any unit vector $\mathbf{y} \in \mathbb{R}^d$, we have

$$\frac{1}{d^{pC}} \cdot \|\mathbf{MPy}\|_p^p \le \|\mathbf{y}\|_p^p \le d^{pC} \cdot \|\mathbf{MPy}\|_p^p,$$

for a fixed constant C > 0. Since **M** is a constant-factor subspace embedding for **A**, then we certainly have

$$\frac{1}{d^{p(C+1)}} \cdot \|\mathbf{APy}\|_p^p \le \|\mathbf{y}\|_p^p \le d^{p(C+1)} \cdot \|\mathbf{APy}\|_p^p.$$

Thus it follows that

$$\frac{1}{d^{p(C+1)}} \cdot \|\mathbf{B}\mathbf{y}\|_p^p \le \|\mathbf{y}\|_p^p \le d^{p(C+1)} \cdot \|\mathbf{B}\mathbf{y}\|_p^p.$$

Therefore,

$$\max_{\mathbf{y}\in\mathbb{R}^d}|\langle \mathbf{b}_t,\mathbf{y}\rangle|^p \le s(\mathbf{b}_t)\cdot d^{p(C+1)},$$

so that

$$\left| \left\| \langle \mathbf{b}_t, \mathbf{y} \rangle^p - \langle \mathbf{b}'_t, \mathbf{y} \rangle^p \right| \le \varepsilon' \cdot s(\mathbf{b}_t) \cdot d^{p(C+1)}. \right.$$

Then by triangle inequality, we have

$$\left| \|\mathbf{B}\mathbf{y}\|_{p}^{p} - \|\mathbf{B}'\mathbf{y}\|_{p}^{p} \right| \leq \sum_{t=1}^{n} \left| \|\langle \mathbf{b}_{t}, \mathbf{y} \rangle - \langle \mathbf{b}'_{t}, \mathbf{y} \rangle \right|^{p} \leq \varepsilon' \cdot \sum_{t=1}^{n} s(\mathbf{b}_{t}) \cdot d^{p(C+1)}.$$

Since the sum of L_p sensitivities is at most d by Lemma 4.11, then

$$\left| \|\mathbf{B}\mathbf{y}\|_p^p - \|\mathbf{B}'\mathbf{y}\|_p^p \right| \le \varepsilon' \cdot d^{p(C+2)}$$

The desired claim then follows by the setting of $\varepsilon' = \frac{\text{poly}(\varepsilon)}{\text{poly}(d^p, \log(nd\Delta))}$, recalling that $\mathbf{B}\mathbf{y} = \mathbf{A}\mathbf{P}\mathbf{P}^{-1}\mathbf{x}$ and $\mathbf{B}'\mathbf{y} = \mathbf{A}'\mathbf{P}\mathbf{P}^{-1}\mathbf{x}$.

We now give the full guarantees of Algorithm 7, which gives an efficient encoding of an input matrix \mathbf{A} .

Lemma 4.13. Let $\mathbf{A} \in \mathbb{R}^{m \times d}$ be a coreset construction with entries in $\{0\} \cup \left[\frac{1}{\operatorname{poly}(n)}, \operatorname{poly}(n)\right]$. Let

$$f\left(d,\frac{1}{\varepsilon},p\right) = \begin{cases} \frac{d}{\varepsilon^2}\log\frac{d}{\varepsilon}\operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1,2)\\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2\\ \frac{d^{p/2}}{\varepsilon^2} \cdot \operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2 \end{cases}$$

Then \mathbf{A}' is a $(1 + \varepsilon)$ -coreset for \mathbf{A} for L_p subspace embedding that uses $\mathcal{O}(f(d, 1, p)) \cdot (d \log n) + md \cdot \operatorname{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log \frac{1}{\delta}\right)$ bits of space.

Proof. Firstly, note that Algorithm 7 outputs a pair $(\mathbf{M}, \mathbf{B}')$ that encodes \mathbf{B}' using \mathbf{M} . By Lemma 4.12, we have that for all $\mathbf{x} \in \mathbb{R}^d$,

$$(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_{1} \le \|\mathbf{A}'\mathbf{x}\|_{1} \le (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_{1},$$

where $\mathbf{A}' = \mathbf{B}'\mathbf{P}^{-1}$. Thus, \mathbf{A}' is a coreset for \mathbf{A} .

It remains to analyze the space complexity of $(\mathbf{M}, \mathbf{B}')$. Since \mathbf{M} is a constant-factor L_p subspace embedding, then by Theorem 4.3, \mathbf{M} can be represented using $\mathcal{O}(f(d, 1, p)) \cdot d\log(n)$ bits of space. Furthermore, each offset \mathbf{b}' in \mathbf{B}' is encoded using $\mathcal{O}\left(d\log\left(\frac{1}{\varepsilon} + \log(nd\kappa)\right)\right)$ bits of space due to storing the d coordinates of the offset rounded to the power of $(1 + \varepsilon')$. Specifically, we store the exponent of each offset after the rounding, rather than the explicit coordinates. Therefore, the output of the algorithm is encoded using $\mathcal{O}(f(d, 1, p)) \cdot (d\log n) + md \cdot \operatorname{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log \frac{1}{\delta}\right)$. \Box

4.2 Subspace Embedding in the Streaming Model

In this section, we show how to use a global encoding combined with the efficient encoding from the previous section in order to achieve our main algorithm for L_p subspace embeddings in the row-arrival model.

Similar to (k, z)-clustering, the main downfall of immediately applying the previous efficient encoding to each node of a merge-and-reduce tree is that the constant-factor approximation requires $\mathcal{O}(d^2 \log(nd))$ bits of storage per efficient encoding, and there can be polylog(log(nd)) such efficient encodings due to the height of the merge-and-reduce tree on a data stream produced by online Lewis weight sampling. As before, we circumvent this issue by showing that we can instead consider a single constant-factor approximation to the global dataset. We then show that the error of maintaining such a global constant-factor approximation does not compound too much over the course of the stream.

We give the algorithm in full in Algorithm 8.

We first show that given disjoint matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_m$ of \mathbf{A} , we can apply Algorithm 7 and although the output \mathbf{Q}'_i from our efficient encoding will no longer necessarily be a strong coreset for each corresponding \mathbf{Q}_i , the resulting error will only be an additive $\varepsilon' \cdot \|\mathbf{A}\mathbf{x}\|_p^p$ for a fixed $\mathbf{x} \in \mathbb{R}^d$, which suffices for our purposes. Specifically, we will set $\varepsilon' = \frac{\varepsilon}{\text{poly}(\log \log(nd))}$ and $m = \text{poly}(\log \log(nd))$, provided the entries of \mathbf{A} are integers bounded in magnitude by poly(nd).

Lemma 4.14. Let $\mathbf{Q}_1 \circ \ldots \circ \mathbf{Q}_m = \mathbf{A}$ be a partition of \mathbf{A} . Let \mathbf{M} be a constant-factor L_p subspace embedding on \mathbf{A} . For each $i \in [m]$, let \mathbf{Q}'_i be the resulting matrix after applying the inverse of the preconditioner to the rounded matrix resulting from \mathbf{Q}_i . Then for all $i \in [m]$ and all $\mathbf{x} \in \mathbb{R}^d$,

$$\left| \|\mathbf{Q}_i'\mathbf{x}\|_p^p - \|\mathbf{Q}_i\mathbf{x}\|_p^p \right| \le \frac{\varepsilon}{\operatorname{poly}(\log\log nd)} \cdot \|\mathbf{A}\mathbf{x}\|_p^p.$$

Algorithm 8 L_p -Embedding on Insertion-Only Stream

Input: Matrix $\mathbf{A} = {\mathbf{a}_1, \dots, \mathbf{a}_n} \in \mathbb{R}^{n \times d}$ that arrives as a data stream, $\varepsilon \in (0, 1)$, parameter $p \ge 1$ **Output:** $(1 + \varepsilon)$ -coreset for L_p -subspace embedding

- 1: $\mathbf{Z} \leftarrow \emptyset, \ \lambda \leftarrow \mathcal{O}\left(\frac{d}{\varepsilon^2} \cdot \log d \log n\right)$
- 2: for each $t \in [n]$ do
- 3: Use $\mathbf{Z} \cup \{\widehat{\mathbf{a}}_t\}$ to compute a $\mathcal{O}(1)$ -approximation $\widehat{\sigma}(\mathbf{a}_t)$ to the online Lewis weight of \mathbf{a}_t
- 4: $p(\mathbf{a}_t) \leftarrow \min(1, \lambda \cdot \sigma(\mathbf{a}_t))$
- 5: With probability $p(\mathbf{a}_t)$, sample $\frac{1}{p(\mathbf{a}_t)} \cdot \mathbf{a}_t$ into a stream \mathcal{S}'
- 6: Let **Z** be the running output of merge-and-reduce on S' using the efficient encoding for coreset construction in Algorithm 7 with a *global* constant-factor embedding, for accuracy $\frac{\varepsilon}{\operatorname{poly}(\log \log nd)}$ and failure probability $\frac{1}{\operatorname{poly}(\frac{1}{\varepsilon}, \log(nd))}$

7: return \mathbf{Z}

Proof. Note that by Lemma 4.12 with accuracy $\frac{\varepsilon}{\operatorname{poly}(\log \log nd)}$, we have

$$\left| \|\mathbf{A}'\mathbf{x}\|_p^p - \|\mathbf{A}\mathbf{x}\|_p^p \right| \le \frac{\varepsilon}{\operatorname{poly}(\log\log nd)} \cdot \|\mathbf{A}\mathbf{x}\|_p^p.$$

Since

$$\left| \| \mathbf{Q}_i' \mathbf{x} \|_p^p - \| \mathbf{Q}_i \mathbf{x} \|_p^p
ight| \le \sum_{i=1}^m \left| \| \mathbf{Q}_i' \mathbf{x} \|_p^p - \| \mathbf{Q}_i \mathbf{x} \|_p^p
ight| = \left| \| \mathbf{A}' \mathbf{x} \|_p^p - \| \mathbf{A} \mathbf{x} \|_p^p
ight|,$$

then the claim follows.

We now give the full guarantees of Algorithm 8 by showing correctness of the global encoding and specifically, that there is no compounding error across the poly(log log(nd)) iterations.

Theorem 4.15. Given a matrix **A** of *n* rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$, with M = poly(n) and online condition number κ , there exists a one-pass streaming algorithm in the row arrival model that outputs a $(1 + \varepsilon)$ -strong coreset of **A** and uses $\mathcal{O}(d) \cdot f\left(d, \frac{1}{\varepsilon}, p\right)$ words of space, where

$$f\left(d,\frac{1}{\varepsilon},p\right) = \begin{cases} \frac{d}{\varepsilon^2}\log\frac{d}{\varepsilon}\operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1,2)\\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2\\ \frac{d^{p/2}}{\varepsilon^2} \cdot \operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2 \end{cases}$$

Proof. Consider Algorithm 8. We prove correctness by induction on t, claiming that after $j \, \cdot \, g\left(n, d, \frac{\operatorname{polylog}(\log(n\kappa))}{\varepsilon}, p\right)$ samples into \mathcal{S}' , then \mathbf{Z} is a $\left(1 + \mathcal{O}\left(\frac{\varepsilon}{\operatorname{poly}(\log\log nd)}\right)\right)^{j+1}$ coreset of \mathbf{A}_t . The online Lewis weight of the first row is always 1 and thus $\mathbf{Z} = \mathbf{a}_1$ after the first step, and the base case is complete. For the inductive hypothesis, we condition on the correctness of \mathbf{Z} being a $\left(1 + \mathcal{O}\left(\frac{\varepsilon}{\operatorname{poly}(\log\log nd)}\right)\right)^j$ -coreset of $\mathbf{A}_{t-1} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_{t-1}$ after j-1 merges have occured. Then $\mathbf{Z} \circ \mathbf{a}_t$ is a $\left(1 + \frac{\varepsilon}{\operatorname{poly}(\log\log nd)}\right)$ -coreset of $\mathbf{A}_t = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_t$, so we can use $\mathbf{Z} \circ \mathbf{a}_t$ to compute a constant-factor approximation to the online Lewis weight $\sigma(\mathbf{a}_t)$ of \mathbf{a}_t . Note that merge-and-reduce will be correct if \mathcal{S}' is a stream with length at most $g\left(n, d, \frac{\operatorname{poly}(\log(\log(n\kappa))}{\varepsilon}, p\right)$, since the failure

probability is set to be $\frac{1}{\operatorname{poly}(\frac{1}{\varepsilon}, \log(n\Delta))}$ and merge-and-reduce will consider at least $g\left(n, d, \frac{1}{\varepsilon}, p\right)$ stream updates before applying a new coreset construction. Thus by Lemma 4.14 the correctness of merge-and-reduce on \mathcal{S}' , we have that \mathbf{Z} after time t will be a $\left(1 + \mathcal{O}\left(\frac{\varepsilon}{\operatorname{poly}(\log\log nd)}\right)\right)^{j+1}$ -strong coreset for X_t after j merges, which completes the induction. Finally, we have that by Theorem 4.5, at most $g\left(n, d, \frac{\operatorname{polylog}(\log(n\kappa))}{\varepsilon}, p\right)$ samples will be inserted into \mathcal{S} , so that $j \leq \operatorname{polylog}(n\kappa)$. The correctness guarantee then follows by rescaling ε .

To analyze the space complexity, note that by Theorem 4.6 through Theorem 4.5, we have that the total number of sampled rows into S' is $n' = \mathcal{O}\left(g\left(n, d, \frac{1}{\varepsilon}, p\right)\right)$, with high probability. We do not maintain S', but instead feed S' as input to the merge-and-reduce subroutine. Hence, the input to merge-and-reduce has size n' and so each coreset implementation has size $f\left(d, \frac{1}{\varepsilon}, p\right)$. polylog(n'). We use the efficient encoding for coreset construction given in Algorithm 7 with accuracy $\frac{\varepsilon}{\text{poly}(\log \log(n\kappa))}$ and failure probability $\frac{1}{\text{poly}(\frac{1}{\varepsilon},\log(n\Delta))}$. Similar to the analysis of Lemma 4.13, we recall that total representation of \mathbf{Z} requires a constant-factor L_p subspace embedding \mathbf{M} that uses $\mathcal{O}\left(f\left(d, 1, p\right)\right) \cdot (d \log n)$ bits of space. We then use Lemma 4.14 to encode each of the coresets of size $f\left(d, \frac{1}{\varepsilon}, p\right) \cdot \text{polylog}(n')$, using $d \cdot \text{polylog}\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log \frac{1}{\delta}\right)$ bits per row. Since $n' = \mathcal{O}\left(g\left(n, d, \frac{1}{\varepsilon}, p\right)\right)$, it follows that there are at most polylog $\left(d, \frac{1}{\varepsilon}, \log(nd\kappa), \log \frac{1}{\delta}\right)$ such coresets. Therefore, the total space required is

$$\mathcal{O}(d\log n) \cdot f(d,1,p) + f\left(d,\frac{1}{\varepsilon},p\right) \cdot \operatorname{polylog}(n') \cdot d \cdot \operatorname{polylog}\left(\frac{1}{\varepsilon},\log(nd\kappa)\right).$$

in bits, which is equivalent to $\mathcal{O}(d) \cdot f\left(d, \frac{1}{\varepsilon}, p\right)$ words of space, as desired.

4.3 Fast Algorithm for Subspace Embeddings

In this section, we show that our one-pass streaming algorithm can be implemented in $\mathcal{O}(d)$ amortized update time. We first require a crude n^{α} -approximation to the online Lewis weight for each row that arrives in the stream. To that end, we first recall that the L_p sensitivity is a poly(d) approximation to the L_p Lewis weights.

Lemma 4.16. [CP15, CD21] For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, let $s(\mathbf{a}_t) := \max_{\mathbf{y} \in \mathbb{R}^d: ||\mathbf{A}\mathbf{y}||_p = 1} |\langle \mathbf{a}_t, \mathbf{y} \rangle|^p$ denote the L_p sensitivity of the t-th row of \mathbf{A} and let $w(\mathbf{a}_t)$ denote the L_p Lewis weight of \mathbf{a}_t . Then

$$\frac{w(\mathbf{a}_t)}{d^{-1-p/2}} \le s(\mathbf{a}_t) \le w(\mathbf{a}_t).$$

We first relate the square root of the leverage score of a row \mathbf{a}_t in a matrix \mathbf{A} with the *p*-th root of the L_p sensitivity of \mathbf{a}_t .

Lemma 4.17. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, let $\xi(\mathbf{a}_t)$ denote the square root of the leverage score of \mathbf{a}_t and let $\chi(\mathbf{a}_t)$ denote the L_p sensitivity of \mathbf{a}_t raised to the $\frac{1}{p}$ power, i.e., $(s(\mathbf{a}_t))^{1/p}$, where $s(\mathbf{a}_t)$ is the L_p sensitivity of \mathbf{a}_t . Then for $p \in [1, 2]$, we have

$$s(\mathbf{a}_t) \le \xi(\mathbf{a}_t) \le n^{\frac{1}{p} - \frac{1}{2}} \cdot s(\mathbf{a}_t),$$

and for p > 2, we have

$$\xi(\mathbf{a}_t) \le s(\mathbf{a}_t) \le n^{\frac{1}{2} - \frac{1}{p}} \cdot \xi(\mathbf{a}_t).$$

– 43 –

Proof. Note that for a fixed vector $\mathbf{x} \in \mathbb{R}^d$ that is not in the kernel of \mathbf{A} , we have that $\frac{\langle \mathbf{a}_t, \mathbf{x} \rangle}{\|\mathbf{A}\mathbf{x}\|_p}$ is within a \sqrt{n} factor of $\frac{\langle \mathbf{a}_t, \mathbf{x} \rangle}{\|\mathbf{A}\mathbf{x}\|_p}$, for $p \in [1, \infty)$. The claim then follows from the definition that the square root of the leverage score is the maximum of $\frac{\langle \mathbf{a}_t, \mathbf{x} \rangle}{\|\mathbf{A}\mathbf{x}\|_2}$ across all possible such \mathbf{x} , while the $\frac{1}{p}$ -th root of the L_p sensitivity is the maximum of $\frac{\langle \mathbf{a}_t, \mathbf{x} \rangle}{\|\mathbf{A}\mathbf{x}\|_p}$ over all such possible \mathbf{x} .

We next show that we can quickly compute crude approximations to the L_p Lewis weights by instead computing crude approximations to the square root of the leverage scores of the rows of the matrix.

Lemma 4.18. Given $\mathbf{A} \in \mathbb{R}^{n \times d}$, let $\mathbf{B} \in \mathbb{R}^{m \times d}$ be a matrix such that $\frac{1}{C} \cdot \mathbf{A}^{\top} \mathbf{A} \preceq \mathbf{B}^{\top} \mathbf{B} \preceq C \cdot \mathbf{A}^{\top} \mathbf{A}$, for some constant C, there exists an algorithm that computes a n^{α} approximations to the L_p Lewis weight using $\mathcal{O}(\operatorname{nnz}(\mathbf{A}))$ runtime.

Proof. Recall that the leverage score of \mathbf{a}_t with respect to \mathbf{A} is $\mathbf{a}_t^{\top} (\mathbf{A}^{\top} \mathbf{A})^{-1} \mathbf{a}^{\top}$. Since $\frac{1}{C} \cdot \mathbf{A}^{\top} \mathbf{A} \preceq \mathbf{B}^{\top} \mathbf{B} \preceq C \cdot \mathbf{A}^{\top} \mathbf{A}$, then $\mathbf{a}_t^{\top} (\mathbf{B}^{\top} \mathbf{B})^{-1} \mathbf{a}^{\top}$ is a *C*-approximation to the leverage score. Let $\mathbf{Z} = (\mathbf{B}^{\top} \mathbf{B})^{-1/2}$, so that $\|\mathbf{Z}\mathbf{a}_t\|_2^2$ is a *C*-approximation to the leverage score of \mathbf{a}_t . Thus it suffices to find an $n^{\alpha/2}$ -approximation to $\|\mathbf{Z}\mathbf{a}_t\|_2$. To that end, observe that for a unit vector $\mathbf{v} \in \mathbb{R}^d$ a random Gaussian vector $\mathbf{g} \in \mathbb{R}^d$ whose entries are independently drawn from $\mathcal{N}(0, 1)$, we have that

$$\begin{aligned} \mathbf{Pr}\left[\langle \mathbf{v}, \mathbf{g} \rangle \geq \log^2 n\right] &\leq 1 - \frac{1}{\operatorname{poly}(n)} \\ \mathbf{Pr}\left[\langle \mathbf{v}, \mathbf{g} \rangle \leq \frac{1}{n^{1/100p}}\right] \leq \mathcal{O}\left(\frac{1}{n^{1/100p}}\right). \end{aligned}$$

Thus by the median over $\mathcal{O}\left(\frac{1}{100p}\right)$ values of $\|\mathbf{g}^{(i)}\mathbf{M}\mathbf{a}_t\|_2$, we can approximately the leverage score of \mathbf{a}_t within $n^{1/100p}$. Hence, we can also approximate the square root of the leverage score of \mathbf{a}_t within $n^{1/100p}$. By Lemma 4.17, this results in a n^{α} -approximation to the L_p Lewis weight of \mathbf{a}_t .

For the runtime, observe that $\mathbf{A}^{\top}\mathbf{A}$ over changes $\operatorname{poly}(d)$ times throughout the stream, so the computation of $\mathbf{g}^{(i)}\mathbf{M}$ is a lower order term than the stream length, i.e., total runtime over all values of \mathbf{M} is o(n). To compute $\mathbf{g}^{(i)}\mathbf{M}\mathbf{a}_t$ from \mathbf{a}_t uses $\mathcal{O}(d)$ time per row and in fact, input-sparsity time over the entire matrix

Putting everything together, we have:

Theorem 4.19. Given a matrix **A** of *n* rows on $[-M, \ldots, -1, 0, 1, \ldots, M]^d$, with M = poly(n) and online condition number κ , there exists a one-pass streaming algorithm in the row arrival model that outputs a $(1 + \varepsilon)$ -strong coreset of **A** and uses $\mathcal{O}(d) \cdot f\left(d, \frac{1}{\varepsilon}, p\right)$ words of space, where

$$f\left(d,\frac{1}{\varepsilon},p\right) = \begin{cases} \frac{d}{\varepsilon^2}\log\frac{d}{\varepsilon}\operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p \in [1,2)\\ \mathcal{O}\left(\frac{d}{\varepsilon^2}\right), & p = 2\\ \frac{d^{p/2}}{\varepsilon^2} \cdot \operatorname{polylog}\left(\frac{d}{\varepsilon}\right), & p > 2 \end{cases}$$

Moreover, the amortized runtime is $\mathcal{O}(d)$ per update.

Proof. We proceed by induction on the time t. The correctness at the first time is clear, since the initial nonzero row must be sampled. Now assuming that the correctness of both a constant-factor

coreset and a $(1 + \varepsilon)$ -coreset at time t - 1, then at time t, either the row is added to the batch of size k, in which case both coreset invariants are maintained, or the sampling procedure is performed on the batch because the batch has gotten too large. In the latter case, we use the constant-factor coreset through Lemma 4.18 to compute a crude approximation to the Lewis scores of each row in a batch of k rows. This uses $\mathcal{O}(\operatorname{nnz}(\mathbf{A}))$ total runtime and produces an insertion-only stream of length $\mathcal{O}\left(n^{1-\Omega(1/p)}\right)$. We then use this as the input to Algorithm 8. By Theorem 4.15, there is poly(d) update time on the stream of length o(n) to generate both a constant-factor coreset and a $(1 + \varepsilon)$ -factor coreset at time t, which completes our induction. Thus for $d \ll n^{1/p}$, the total runtime for Algorithm 8 is $\mathcal{O}(nd)$, and so the overall runtime is $\mathcal{O}(nd + \operatorname{nnz}(\mathbf{A}))$, which is amortized runtime $\mathcal{O}(d)$.

Acknowledgments

The work was initialized while David P. Woodruff and Samson Zhou were visiting the Institute for Emerging CORE Methods in Data Science (EnCORE) supported by the NSF grant 2217058. The work was conducted in part while David P. Woodruff and Samson Zhou were visiting the Simons Institute for the Theory of Computing as part of the Sublinear Algorithms program. David P. Woodruff is supported in part by Office of Naval Research award number N000142112647 and a Simons Investigator Award. Liudeng Wang and Samson Zhou were supported in part by NSF CCF-2335411.

References

- [AV07] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 1027–1035, 2007. 4
- [BBC⁺19] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for k-means: beyond subspaces and the johnson-lindenstrauss lemma. In Moses Charikar and Edith Cohen, editors, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019, pages 1039–1050. ACM, 2019. 3, 4
- [BCF24] Sayan Bhattacharya, Martín Costa, and Ermiya Farokhnejad. Fully dynamic *k*-median with near-optimal update time and recourse. *CoRR*, abs/2411.03121, 2024. 3
- [BCG⁺24] Sayan Bhattacharya, Martín Costa, Naveen Garg, Silvio Lattanzi, and Nikos Parotsidis.
 Fully dynamic k-clustering with fast update time and small recourse. In 65th IEEE Annual Symposium on Foundations of Computer Science, FOCS, pages 216–227, 2024.
 3
- $[BCLP23] Sayan Bhattacharya, Martín Costa, Silvio Lattanzi, and Nikos Parotsidis. Fully dynamic k-clustering in <math>\tilde{O}(k)$ update time. In Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS, 2023. 3, 5, 19, 21

- [BDM⁺20] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS, pages 517–528, 2020. 5, 6, 36, 37
- [BFL⁺21] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. Efficient coreset constructions via sensitivity sampling. In Asian Conference on Machine Learning, ACML, pages 948–963, 2021. 6, 13
- [BFLR19] Vladimir Braverman, Dan Feldman, Harry Lang, and Daniela Rus. Streaming coreset constructions for m-estimators. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM, pages 62:1–62:15, 2019. 2, 3, 4
- [BIRW16] Arturs Backurs, Piotr Indyk, Ilya P. Razenshteyn, and David P. Woodruff. Nearlyoptimal bounds for sparse recovery in generic norms, with applications to k-median sketching. In Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 318–337, 2016. 8
- [BS80] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-todynamic transformation. J. Algorithms, 1(4):301–358, 1980. 13
- [CD21] Xue Chen and Michal Derezinski. Query complexity of least absolute deviation regression via robust uniform convergence. In *Conference on Learning Theory, COLT*, volume 134, pages 1144–1179, 2021. 37, 39, 43
- [CEM⁺15] 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, STOC, pages 163–172, 2015. 4
- [Che09] 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. 2, 3, 4
- [CLN⁺20] Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. Fast and accurate \$k\$-means++ via rejection sampling. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS, 2020. 8, 30
- [CLS⁺22] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, Chris Schwiegelshohn, and Omar Ali Sheikh-Omar. Improved coresets for euclidean k-means. In *NeurIPS*, 2022. 2, 11, 13
- [CLSS22] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, pages 1038–1051, 2022. 2, 4, 11, 13
- [CMP20] Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. *Theory* Comput., 16:1–25, 2020. 5, 6, 36

- [CP15] Michael B. Cohen and Richard Peng. l_p row sampling by lewis weights. In Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC, pages 183–192, 2015. 5, 6, 36, 37, 39, 43
- [CSS21] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In STOC: 53rd Annual ACM SIGACT Symposium on Theory of Computing, pages 169–182, 2021. 2, 4, 13
- [CSWZ23] Yeshwanth Cherapanamjeri, Sandeep Silwal, David P. Woodruff, and Samson Zhou. Optimal algorithms for linear algebra in the current matrix multiplication time. In Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 4026–4049, 2023. 36
- [CW09] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC, pages 205–214, 2009. 36, 38
- [CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In Symposium on Theory of Computing Conference, STOC, pages 81–90, 2013. 2
- [CWZ23] Vincent Cohen-Addad, David P. Woodruff, and Samson Zhou. Streaming euclidean k-median and k-means with o(log n) space. In 64th IEEE Annual Symposium on Foundations of Computer Science, FOCS, pages 883–908, 2023. 2, 3, 4, 6, 9, 12, 13
- [DDH⁺09] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for l_p regression. SIAM J. Comput., 38(5):2060– 2078, 2009. 6, 36, 37, 38
- [DMM06a] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM, Proceedings, pages 316–326, 2006. 2, 5, 6, 36
- [DMM06b] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *Algorithms - ESA* 2006, 14th Annual European Symposium, Proceedings, pages 304–314, 2006. 2, 5, 6, 36
- [DSS24] Andrew Draganov, David Saulpic, and Chris Schwiegelshohn. Settling time vs. accuracy tradeoffs for clustering big data. *Proc. ACM Manag. Data*, 2(3):173, 2024. 2, 5
- [FL11] 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, pages 569–578, 2011. 2, 3, 4, 6, 13
- [FS12] Dan Feldman and Leonard J. Schulman. Data reduction for weighted and outlierresistant clustering. In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 1343–1354, 2012. 6, 13

- [GT08] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008. 5, 7, 21
- [HK07] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discret. Comput. Geom.*, 37(1):3–19, 2007. 2, 3, 4
- [HK20] Monika Henzinger and Sagar Kale. Fully-dynamic coresets. In 28th Annual European Symposium on Algorithms, ESA, volume 173, pages 57:1–57:21, 2020. 3
- [HLW23] Lingxiao Huang, Jian Li, and Xuan Wu. On optimal coreset construction for euclidean (k, z)-clustering, 2023. 2, 11, 13
- [HM04] 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, pages 291–300, 2004. 2, 3, 4, 13
- [HV20] Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in euclidean spaces: importance sampling is nearly optimal. In Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC, pages 1416–1429, 2020. 4
- [ISZ21] Zachary Izzo, Sandeep Silwal, and Samson Zhou. Dimensionality reduction for wasserstein barycenter. In Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS, 2021. 4, 30
- [IT03] Piotr Indyk and Nitin Thaper. Fast image retrieval via embeddings. In International Workshop on Statistical and Computational Theories of Vision, ICCV Workshop, volume 120, 2003. 8
- [JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. conference in modern analysis and probability (new haven, conn., 1982), 189–206. In *Contemp. Math*, volume 26, 1984. 11, 30
- [ITHS24] Max Dupré la Tour, Monika Henzinger, and David Saulpic. Fully dynamic k-means coreset in near-optimal update time. In 32nd Annual European Symposium on Algorithms, ESA, pages 100:1–100:16, 2024. 5
- [ITS24] Max Dupré la Tour and David Saulpic. Almost-linear time approximation algorithm to euclidean k-median and k-means. *CoRR*, abs/2407.11217, 2024. 5
- [LWW21] Yi Li, Ruosong Wang, and David P. Woodruff. Tight bounds for the subspace sketch problem with applications. *SIAM J. Comput.*, 50(4):1287–1335, 2021. 5
- [Mag10] Malik Magdon-Ismail. Row sampling for matrix algorithms via a non-commutative bernstein bound. *CoRR*, abs/1008.0587, 2010. 2
- [MMM⁺22] Raphael A. Meyer, Cameron Musco, Christopher Musco, David P. Woodruff, and Samson Zhou. Fast regression for structured inputs. In *The Tenth International Conference on Learning Representations, ICLR, 2022,* 2022. 36

- [MMM⁺23] Raphael A. Meyer, Cameron Musco, Christopher Musco, David P. Woodruff, and Samson Zhou. Near-linear sample complexity for L_p polynomial regression. In Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 3959–4025, 2023. 36
- [MMR19] Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC, pages 1027–1038, 2019. 4, 30
- [MMWY22] Cameron Musco, Christopher Musco, David P. Woodruff, and Taisuke Yasuda. Active linear regression for l_p norms and beyond. In 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS, pages 744–753, 2022. 36
- [MWZ22] Sepideh Mahabadi, David P. Woodruff, and Samson Zhou. Adaptive sketches for robust regression with importance sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 31:1–31:21, 2022. 36
- [PPP21] Aditya Parulekar, Advait Parulekar, and Eric Price. L1 regression with lewis weights subsampling. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM, pages 49:1–49:21, 2021. 36
- [Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), Proceedings, pages 143–152, 2006. 5, 6, 36, 37
- [SW18] Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS, pages 802–813, 2018. 14
- [Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014. 2
- [WY23] David P. Woodruff and Taisuke Yasuda. Online lewis weight sampling. In Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 4622–4666, 2023. 5, 6, 36, 37
- [WZZ23] David P. Woodruff, Peilin Zhong, and Samson Zhou. Near-optimal k-clustering in the sliding window model. In Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS 2023, 2023. 2, 3, 6, 13
- [ZTHH24] Xiaoyi Zhu, Yuxiang Tian, Lingxiao Huang, and Zengfeng Huang. Space complexity of euclidean clustering. arXiv preprint arXiv:2403.02971, 2024. 5, 9