CLUSTERING WITH OUTLIERS ON SLIDING WINDOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study sliding window algorithms for (k, z)-clustering with m outliers in Euclidean spaces, and focus on constructing ϵ-*coresets*, which is a powerful notion of data reduction for clustering. Our main result is an algorithm that maintains an ϵ-*coreset* of size $\widetilde{O}\big(k/\min\{\epsilon^4, \epsilon^{2+z}\} + 2^{O(z\log(z+1))}(d^z + kd)m/\epsilon\big)$, and this readily yields a $(1+\epsilon)$-approximate solution center set. We evaluate our algorithm for k-means with outliers on five real data sets with injected outliers, the experiment result shows that it has lower empirical cost than uniform sampling and coreset for vanilla k-means at every tested coreset size.

1 INTRODUCTION

Clustering with outliers is a standard robust formulation of clustering (Charikar et al., 2001; Chen, 2008; Krishnaswamy et al., 2018; Agrawal et al., 2023; Zhang et al., 2024; Gupta et al., 2025). It is motivated by data containing measurement errors, corrupted records, or atypical points: even a small number of such points can dominate the clustering objective and move the centers far from the structure of the remaining data. The formulation limits their influence by allowing a fixed number of input points to be discarded. Formally, for a finite point set X in a metric space, a set C of k centers, an exponent $z \geq 1$, and an outlier budget m, define $\mathrm{dist}(x, C) \coloneqq \min_{c\in C} \mathrm{dist}(x, c)$ and

$$\mathrm{cost}_z^{(m)}(X, C) \coloneqq \min_{O\subseteq X, |O|\leq m} \sum_{x\in X\setminus O} \mathrm{dist}^z(x, C), \qquad \mathrm{OPT}_z^{(m)}(X) \coloneqq \min_{|C|=k} \mathrm{cost}_z^{(m)}(X, C). \tag{1}$$

We call this the (k, z, m)-clustering problem; $z = 1$ and $z = 2$ give k-median and k-means with outliers, respectively.

The sliding-window setting. We focus on the sliding-window setting. In many applications, points arrive continuously and recent data are more relevant than old data. The sliding-window model captures this setting by retaining only the most recent updates. Clustering in this model has been studied extensively (Babcock et al., 2003; Braverman et al., 2016; Borassi et al., 2020; Epasto et al., 2022; Woodruff et al., 2023). Formally, let $p_1, \ldots, p_T$, where $T \leq n$, be an insertion-only stream, and let $W \leq n$ be the window length. After the point with timestamp t arrives, the active input is $\mathsf{Win}_t \coloneqq (p_{\max\{1,t-W+1\}}, \ldots, p_t)$; all earlier points have expired. A central goal is to maintain an approximate solution for the active window after every update without storing the entire window.

Coresets for clustering with outliers. We speicifically aim to maintain *coresets* for (k, z, m)-clustering (see Definition 2.1) in sliding windows. A coreset is a small weighted representation of the input on which clustering costs can be evaluated approximately, which is conceptually suitable for addressing the bounded space challenge of sliding window. Indeed, coresets for (k, z, m)-clustering has received substantial attention recently (Feldman & Schulman, 2012; Huang et al., 2018; 2023; Jiang & Lou, 2025), and these coreset results readily imply streaming algorithms for (k, z, m)-clustering in *insertion-only* streams.

However, efficient (low-space) constructions for coresets for (k, z, m)-clustering in the more general sliding-window setting is still open. In fact, similar sliding-window questions have been asked for (variants of) clustering, but the message is mixed. For vanilla clustering, the space bound of sliding-window algorithms is comparable to that of insertion-only ones (Braverman et al., 2016; Woodruff et al., 2023), whereas for fair clustering, Cohen-Addad et al. (2025) showed that any finite approximation that satisfies the fairness constraints requires linear space.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Main result. We make the conceptual contribution that coreset for clustering with outliers does admit an efficient sliding-window algorithm, and the space complexity is comparable to that in the insertion-only setting.

Theorem 1.1. *There is a randomized algorithm that, for any integers $z, k, m, n, d, \Delta, W \geq 1$ with $W \leq n$, any $0 < \varepsilon < 1$, and any stream of n distinct points from $[\Delta]^d$, reports an ε-coreset for (k, z, m)-clustering on the current window of size*

$$\widetilde{O}\left(\frac{k}{\min\{\varepsilon^4, \varepsilon^{2+z}\}} + 2^{O(z\log(z+1))}\frac{(d^z + kd)m}{\varepsilon}\right) \tag{2}$$

after every update with high probability, where $\widetilde{O}$ hides factors polylogarithmic in the problem parameters. The same bound applies to the space complexity, excluding the vanilla coreset black box[1].

Our result is the first nontrivial sliding-window coreset for (k, z, m)-clustering, there is no direct comparison we can make from the literature. Perhaps the closest one is sliding-window algorithms for vanilla clustering (where $m = 0$), and we achieve asymptotically the same space bounds as in Woodruff et al. (2023), which is the best in this category. Furthermore, our coreset size bound is competitive even compared with offline/non-streaming constructions. Our first term $\frac{k}{\min\{\epsilon^4, \epsilon^{2+z}\}}$, which corresponds to the bound when $m = 0$, nearly matches the general lower bound for coresets for vanilla clustering (Huang et al., 2024) in the same linear-k regime, up to an ϵ factor. This lower bound carries on to our sliding window (k, z, m)-clustering setting. The linear dependence on m is also unavoidable in general (Huang et al., 2023), albeit known offline coresets for (k, z, m)-clustering (Huang et al., 2026) can achieve $O(m)$ dependence, which is better than our $O(kdm)$. Achieving this in sliding window can be highly nontrivial and is an interesting future direction.

Our algorithm reacts window updates in polynomial time (with respect to main parameters $k, d, m, \varepsilon^{-1}, \log\Delta$). As can be implied from the definition of coreset, one can apply a black-box downstream α-approximate algorithm for (k, z, m)-clustering only on the coreset, and the returned result becomes $\alpha(1 + \epsilon)$-approximation.

Technical overview. Our overall strategy is to adapt a framework of Woodruff et al. (2023) which was devised to solve vanilla (k, z)-clustering in sliding window. The key step in this framework, is turning algorithm design for sliding-window, to a prefix-query insertion-only task: devise a *mergeable* summary that returns a coreset for *every prefix* of an input sequence. For vanilla clustering, such a summary was designed to simply maintain a vanilla coreset of the entire input, and then on query, return the coreset points intersecting the prefix.

However, this old plan of simply returning a prefix of a maintained coreset, is nontrivial to implement in our outlier setting. Specifically, we consider a common architecture for coresets with outliers (Huang et al., 2023; Jiang & Lou, 2025), where the input X is split into an outer part $\mathsf{Outer}(X)$, which contains points that may need to be treated as outliers and is kept exactly, and an inlier part $\mathsf{Inner}(X) \coloneqq X \setminus \mathsf{Outer}(X)$, which is compressed by an ordinary coreset algorithm $\mathsf{Vanilla}$. The resulting coreset has the format

$$\mathcal{R}(X) \coloneqq \mathsf{Outer}(X) \cup \mathsf{Vanilla}(\mathsf{Inner}(X)). \tag{3}$$

Now, two issues emerge: a) taking the prefix of the outlier set does not make sense, because then the returned set may not even contain enough number of outliers; and b) a more subtle issue is that, when we merge two summaries for X_1 and X_2, it is possible that after the merge some point from $\mathsf{Inner}(X_1)$ becomes part of $\mathsf{Outer}(X_1 \cup X_2)$, but what we store is only a summary $\mathsf{Vanilla}(\mathsf{Inner}(X))$ which may not contain this information.

Our idea. A key new feature of our mergeable summary algorithm, is that, on merges, it moves points only from the explicitly stored part toward the compressed part. This can resolve the mergeable issue. To implement this, we modify a recent coreset construction for (k, z, m)-clustering (Jiang & Lou, 2025, Reduction I): they pick $\mathsf{Outer}(X)$ by first drawing a small uniform sample S, then from $X \setminus S$, include all points from light buckets of a data-oblivious space partition, where a light bucket is a bucket that contains a small number of points. All other points constitute $\mathrm{Inner}(X)$ and a

[1] We account for this space in Section B.5.

vanilla coreset is built on it. We implement the sampling with shared random priorities and keep a bounded number of points from each remaining light buckets (with respect to the same data-oblivious partition) explicitly. Now, when merge happens, we can achieve the main feature as follows: if the merged uniform sample have more than required points, we move the excessive points to the buckets. Then this can only make buckets heavier, and potentially some buckets are no longer light and the points go to $\mathsf{Inner}(X)$.

Finally, to support all prefix queries, we store a compact history of changes to the explicitly retained points rather than a separate coreset for every prefix. These histories can themselves be merged, fulfilling the mergeable requirement.

Experimental evaluation. We evaluate robust k-means using a practical variant of our summary on five real data sets with injected outliers. We compute centers from its weighted output and measure their cost on the full active window, comparing against size-matched uniform samples, an offline robust coreset, and an online vanilla coreset (Woodruff et al., 2023). Our method has a lower mean empirical cost ratio than uniform sampling at all 45 combinations of data set and coreset size. At the smallest tested sizes, which retain 7–11% of each window, its mean cost ratios range from 1.22 to 1.44. The offline coreset is usually more accurate, but requires the full window at each query.

1.1 Related Work

Clustering in sliding windows. Babcock et al. (2003) initiated the study of sliding-window clustering with an algorithm for k-median whose space had a polynomial dependence on the window length. Subsequent work obtained a bicriteria approximation in polylogarithmic space (Braverman et al., 2015) and then the first proper constant-factor approximations for both k-median and k-means in $\mathrm{poly}(k, \log W)$ space (Braverman et al., 2016). Borassi et al. (2020) reduced the dependence on k to linear, up to polylogarithmic factors, and also gave practical implementations. Epasto et al. (2022) gave the first $(1+\varepsilon)$-approximation for general (k, z)-clustering, although with large polynomial dependencies on the dimension or $1/\varepsilon$. Finally, Woodruff et al. (2023) introduced online coresets and obtained nearly optimal $(1+\varepsilon)$-approximation and coreset bounds. Our work uses their reduction to sliding windows, but replaces its online-coreset input by a richer mergeable summary for a multi-stage coreset construction. The max-distance k-center objective with outliers has also been studied in sliding windows: de Berg et al. (2021) gave a $(1+\varepsilon)$-approximation in bounded-doubling metrics, while Pellizzoni et al. (2022) obtained constant-factor algorithms in general metrics with space linear in the number of centers and outliers, up to logarithmic factors. These results do not cover the finite-z sum objectives considered here.

2 Preliminaries

Notation. We assume that the input stream consists of distinct unweighted points from $[\Delta]^d$, where Δ is a positive integer and $[\Delta] \coloneqq \{1, \ldots, \Delta\}$. Cluster centers may be chosen from $\mathbb{R}^d$. Our algorithm can also be generalized to receive inputs from $\mathbb{R}^d$, we use $[\Delta]^d$ for the simplicity of representation here. A weighted point set X is equipped with a weight function $w_X : X \to \mathbb{R}_{\geq 0}$, and we write $w(X) \coloneqq \sum_{x \in X} w_X(x)$ for its total weight. For a point x and a center set $C \subseteq \mathbb{R}^d$, we write $\mathrm{dist}(x, C) \coloneqq \min_{c \in C} \mathrm{dist}(x, c)$.

For sequence $Z \coloneqq (z_1, \ldots, z_q)$, define $Z_{\leq t} \coloneqq (z_1, \ldots, z_t)$ for $1 \leq t \leq q$. Specifically, $Z_{\leq 0}$ is an empty sequence. For sequences $X \coloneqq (x_1, \ldots, x_a)$ and $Y \coloneqq (y_1, \ldots, y_b)$, define $X \circ Y \coloneqq (x_1, \ldots, x_a, y_1, \ldots, y_b)$. For subset $A \subseteq Z$, define $A_{\leq t} \coloneqq A \cap Z_{\leq t}$.

We use the following weighted extension of Equation (1) for clustering costs. For $h \geq 0$, let

$$\mathcal{L}_X(h) \coloneqq \left\{ \ell : X \to \mathbb{R}_{\geq 0} : 0 \leq \ell(x) \leq w_X(x) \text{ and } \sum_{x \in X} \ell(x) \leq h \right\}.$$

For a set $C \subseteq \mathbb{R}^d$ of centers, define

$$\mathrm{cost}_z^{(h)}(X, C) \coloneqq \min_{\ell \in \mathcal{L}_X(h)} \sum_{x \in X} \big(w_X(x) - \ell(x)\big)\, \mathrm{dist}^z(x, C), \tag{4}$$

and

$$\mathrm{OPT}_z^{(h)}(X) \coloneqq \min_{C\subseteq\mathbb{R}^d,\,|C|=k} \mathrm{cost}_z^{(h)}(X,C).$$

Definition 2.1 (Coreset with outliers)**.** *Let X be a weighted point set and $m \geq 0$. A weighted point set S supported on X is an ε-coreset for (k,z,m)-clustering on X if, for every $C \subseteq \mathbb{R}^d$ with $|C| = k$ and every $0 \leq h \leq m$,*

$$\mathrm{cost}_z^{(h)}(S,C) \in (1\pm\varepsilon)\,\mathrm{cost}_z^{(h)}(X,C).$$

2.1 Reduction I of Jiang & Lou (2025)

We review Jiang & Lou (2025, Reduction I). Fix an optimum guess $\widehat{\mathrm{OPT}}$ and an internal accuracy ε. Set

$$\lambda \coloneqq (z+1)^{-c}\left(\frac{\varepsilon\widehat{\mathrm{OPT}}}{m}\right)^{1/z}, \qquad M \coloneqq \left\lceil (1+\varepsilon^{-1})m \right\rceil, \tag{5}$$

where c is a sufficiently large absolute constant.

Definition 2.2 (Consistent hash (Czumaj et al., 2022))**.** *For a scale $\lambda > 0$, a (Γ,Λ)-consistent hash is a data-oblivious map $\varphi : [\Delta]^d \to \mathcal{U}$ such that every fiber $\varphi^{-1}(u)$ has diameter at most λ, and every set $A \subseteq [\Delta]^d$ with $\mathrm{diam}(A) \leq \lambda/\Gamma$ intersects at most Λ fibers, or equivalently, $|\varphi(A)| \leq \Lambda$. On a point set X, the nonempty fibers induce the bucket partition*

$$\mathcal{P}_\varphi(X) \coloneqq \{X \cap \varphi^{-1}(u) : u \in \varphi(X)\}.$$

In Euclidean space, such a deterministic hash exists with $\Gamma = O(d)$ and $\Lambda = O(d\log d)$, and can be stored and evaluated in $\mathrm{poly}(d, \log\Delta)$ space. We fix one such hash for each optimum guess and use it throughout the corresponding construction.

Reduction I applies three operations in sequence:

$$\begin{aligned} F_1 &\leftarrow \mathsf{Isolate}_{\mathrm{off}}(X), & F_2 &\leftarrow \mathsf{Sparse}_{\mathrm{off}}(X\setminus F_1),\\ \mathsf{Outer}_{\mathrm{off}}(X) &\leftarrow F_1\cup F_2, & \mathsf{Inner}_{\mathrm{off}}(X) &\leftarrow X\setminus \mathsf{Outer}_{\mathrm{off}}(X),\\ V_{\mathrm{in}} &\leftarrow \mathsf{Vanilla}(\mathsf{Inner}_{\mathrm{off}}(X)). \end{aligned} \tag{6}$$

When the guess $\widehat{\mathrm{OPT}}$ is "successful", the isolation stage leaves at most $N \coloneqq 2k\Lambda$ nonempty residual buckets. The sparse stage then stores every residual bucket of size below M exactly, so every bucket surviving into $\mathsf{Inner}_{\mathrm{off}}(X)$ has size at least M; it also has diameter at most λ by the consistent-hash property. The vanilla stage therefore only needs to handle large, low-diameter buckets: it represents $\mathsf{Inner}_{\mathrm{off}}(X)$ by an ε-coreset of vanilla (k,z)-clustering, which approximates $\mathrm{cost}_z^{(0)}(\mathsf{Inner}_{\mathrm{off}}(X), C)$ for every $|C| = k$.

The isolation stage uses two-level sampling without replacement. Given a nonempty set Z, one two-level sample first chooses a bucket $u \in \varphi(Z)$ uniformly and then chooses an occurrence uniformly from $Z_u \coloneqq Z \cap \varphi^{-1}(u)$. Starting from $G_0(Z) \coloneqq \varnothing$, in round i draw g_i from $Z \setminus G_{i-1}(Z)$ in this way and set

$$G_i(Z) \coloneqq G_{i-1}(Z) \cup \{g_i\}. \tag{7}$$

We define $\mathsf{Isolate}_{\mathrm{off}}(Z) \coloneqq G_r(Z)$, returning all of Z if fewer than r occurrences are available.

The other two operations are deterministic once the hash and the vanilla coreset algorithm are fixed. For a residual set R, define $R_u \coloneqq R \cap \varphi^{-1}(u)$ and

$$\mathsf{Sparse}_{\mathrm{off}}(R) \coloneqq \bigcup_{\substack{u\in\varphi(R)\\ |R_u|<M}} R_u. \tag{8}$$

Finally, for any input Y, define $\mathsf{Vanilla}(Y) \coloneqq$ the weighted ε-coreset returned by a full-input query to the vanilla online-coreset algorithm of Woodruff et al. (2023) after processing Y.

The following lemma states the reduction directly in terms of the three operations above.

Lemma 2.3 (Reduction I, informal (Jiang & Lou, 2025)). *For a fixed input X, call a guess $\widehat{\mathrm{OPT}}$ successful for X if $|\varphi(X \setminus \mathsf{Isolate}_{\mathrm{off}}(X))| \le N$. For every successful guess satisfying $\widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(X)$, the weighted set $\mathsf{Outer}_{\mathrm{off}}(X) \cup \mathsf{Vanilla}(\mathsf{Inner}_{\mathrm{off}}(X))$ is an $O(\varepsilon)$-coreset for (k,z,m)-clustering. Moreover, if $\frac{\mathrm{OPT}_z^{(m)}(X)}{z+1} \le \widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(X)$, then choosing*

$$r = \widetilde{O}\left(2^{O(z\log(z+1))}\frac{\Gamma^z m}{\varepsilon}\right) \tag{9}$$

makes the guess successful with high probability.

3 Mergeable Coreset Summaries

We first define the summary interface required by the sliding-window framework, state our core summary theorem, and derive Theorem 1.1. The remainder of the section proves the core theorem by making the reduction of Jiang & Lou (2025) mergeable.

3.1 Summary interface and main results

Definition 3.1 (Mergeable coreset summary). *Let $P \coloneqq (p_1,\dots,p_q)$ be a sequence of unweighted points. An ε-mergeable coreset summary for P is a data structure $\mathcal{D}_P$ supporting the following operations.*

1. $\mathsf{PrefixCoreset}(\mathcal{D}_P, t)$ *returns an ε-coreset for (k,z,m)-clustering on $P_{\le t}$.*
2. *For sequences X and Y, $\mathsf{Merge}(\mathcal{D}_X, \mathcal{D}_Y)$ returns a mergeable coreset summary $\mathcal{D}_{X\circ Y}$ of their concatenation.*

This definition generalizes the notion of online coreset as in (Woodruff et al., 2023), which specifically does not require explicitly maintaining a single coreset and directly taking its prefix to answer queries. Nevertheless, the main sliding-window framework of Woodruff et al. (2023) , stated in the following lemma, still applies to Definition 3.1, since its proof uses only the prefix-query and merge operations.

Lemma 3.2 (Woodruff et al. (2023), Theorem 4.1). *Let $W \le n$. Suppose there is an ε-mergeable coreset summary for sequences of length at most W such that, with probability at least $1-\delta$, every prefix query returns a coreset of size at most $N_{\mathrm{coreset}}(W,\varepsilon,\delta)$. Then the reduction yields a sliding-window algorithm for streams of length at most n that maintains an ε-coreset for (k,z,m)-clustering of size*

$$O\left(N_{\mathrm{coreset}}\left(W, \Theta\left(\frac{\varepsilon}{\log W}\right), \frac{\delta}{\mathrm{poly}(n)}\right)\log W\right) \tag{10}$$

and succeeds for all stream updates with probability at least $1-\delta$.

The preceding reduction makes the following summary lemma the core technical result of this paper.

Lemma 3.3. *There is a randomized construction that, for any integers $z,k,m,L,d,\Delta \ge 1$ and any $0<\varepsilon,\delta<1$, yields an $O(\varepsilon)$-mergeable coreset summary for sequences of length at most L. Set $\delta_V \coloneqq \delta/\mathrm{poly}(L,d,z,\log\Delta)$. For any fixed merge tree, with probability at least $1-\delta$, all prefix queries are correct and return coresets of size at most*

$$\widetilde{O}\left(2^{O(z\log(z+1))}\frac{(d^z+kd)m}{\varepsilon} + N_{\mathcal{V}}(L,\varepsilon,\delta_V)\right). \tag{11}$$

Proof of Theorem 1.1. Apply Lemma 3.3 with internal accuracy $\varepsilon_0 \coloneqq c_0\varepsilon/\log W$ for a sufficiently small absolute constant c_0, input-length bound W, and failure probability $\delta_s \coloneqq \delta/\mathrm{poly}(n)$ for each summary created over the entire stream. Within each summary, the vanilla coreset uses failure probability $\delta_V = \delta_s/\mathrm{poly}(W,d,z,\log\Delta)$. The resulting summary satisfies the hypothesis of Lemma 3.2, which therefore returns an ε-coreset for the current window simultaneously after every update with probability at least $1-\delta$.

The vanilla online coreset of Woodruff et al. (2023, Theorem 1.3) satisfies

$$N_{\mathcal{V}}(W,\varepsilon_0,\delta_V) = O\left(\frac{k}{\min\{\varepsilon_0^4,\varepsilon_0^{2+z}\}}\,\mathrm{polylog}\left(\frac{W\Delta}{\varepsilon_0\delta_V}\right)\right).$$

Substituting this bound and the chosen value of ε_0 into Equation (11), and accounting for the $O(\log W)$ active levels in Lemma 3.2, gives Equation (2). All additional logarithmic factors are absorbed by the $\widetilde{O}$ notation. □

The remainder of this section proves Lemma 3.3. Reduction I and its three operations were defined in Section 2.1; we now make their computation mergeable and establish the required invariants and correctness guarantees.

3.2 A MERGEABLE IMPLEMENTATION OF REDUCTION I

Lemma 3.4. *There is a randomized summary $\mathcal{D}_Z$ that supports prefix queries and merges of sequences. For every prefix $P = Z_{\le t}$ with a successful guess $\widehat{\mathrm{OPT}}$ (see Definition 3.5) satisfying $\widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(P)$, the query returns $\mathsf{Isolate}(P)$, $\mathsf{Sparse}(P)$, and a weighted set V_t such that $\mathsf{Isolate}(P) \cup \mathsf{Sparse}(P) \cup V_t$ is an $O(\varepsilon)$-coreset for (k, z, m)-clustering on P. All such prefix queries are correct with high probability, and every prefix output has size $\widetilde{O}(r + NM) + N_{\mathcal{V}}$, where r, N, M are the parameters of Reduction I and $N_{\mathcal{V}}$ is the size of the vanilla online coreset.*

Proof sketch of Lemma 3.4. The main challenge is to make the transition from Outer to Inner one-way, so that no merge needs to recover points from $\mathsf{Vanilla}(\mathsf{Inner}(Z))$. To that end, we partition Z into three sets U_Z, H_Z, and T_Z, and maintain the invariant that each point moves in only one direction during merges:

$$\underbrace{U_Z}_{\text{protected points}} \longrightarrow \underbrace{H_Z}_{\text{bucket heads}} \longrightarrow \underbrace{T_Z}_{\text{vanilla tail}}. \tag{12}$$

$\mathsf{Isolate}(Z), \mathsf{Sparse}(Z)$ and $\mathsf{Inner}(Z)$ can be obtained from U_Z, H_Z, T_Z.

Maintain $\mathsf{Isolate}(Z)$. We use Min-Hash to simulate the two-level sampling without replacement in Equation (7). For each round $i \in [r]$, fix independent global hashes $h_i : \mathcal{U} \to (0,1)$ and $\widetilde{h}_i : [\Delta]^d \to (0,1)$, and set

$$q_i(x) \coloneqq \big(h_i(\varphi(x)), \widetilde{h}_i(x), x\big).$$

Starting from $S_0(Z) \coloneqq \varnothing$, define

$$s_i(Z) \coloneqq \operatorname*{arg\,min}_{x \in Z \setminus S_{i-1}(Z)} q_i(x), \qquad S_i(Z) \coloneqq S_{i-1}(Z) \cup \{s_i(Z)\}. \tag{13}$$

Hence $s_i(Z)$ is sampled by first choosing a uniform residual bucket and then a uniform point in that bucket, so $S_r(Z)$ has the same distribution as $\mathsf{Isolate}_{\mathrm{off}}(Z)$. The Isolate summary is $\mathcal{IS}_Z \coloneqq (\mathcal{I}_Z, U_Z)$, where $\mathcal{I}_Z$ stores $S_r(Z_{\le t})$ for every prefix t and

$$\mathsf{Isolate}(Z) \coloneqq U_Z = \bigcup_t S_r(Z_{\le t}). \tag{14}$$

In words, $\mathcal{IS}_Z$ stores both the sampled states and every sampled point, together with the sequence length for indexing during merges.

Prefix Query. Sample heredity implies that an occurrence first enters U_Z at its own index. Hence $\mathsf{IsolateQuery}(\mathcal{IS}_Z, t) \coloneqq (U_Z)_{\le t} = U_{Z_{\le t}} = \mathsf{Isolate}(Z_{\le t})$. This may contain more points than the current sample $S_r(Z_{\le t})$; all of them are retained exactly, and removing them cannot increase the number of residual buckets.

Merging. The global priorities give the exact identity

$$s_i(X \circ Y) = \operatorname*{arg\,min}_{x \in C_i} q_i(x), \qquad C_i \coloneqq \big(S_i(X) \cup S_i(Y)\big) \setminus S_{i-1}(X \circ Y). \tag{15}$$

Applying it to the child histories constructs $\mathcal{IS}_{X \circ Y}$. $\mathsf{IsolateMerge}(\mathcal{IS}_X, \mathcal{IS}_Y)$ also returns the ordered release stream R_I of child-protected points no longer protected by the parent. These points remain explicit for the next component; the history, protected set, and release stream contain $\widetilde{O}(r)$ records and points with high probability.

The success condition checks whether the protected points leave at most N residual hash buckets.

Definition 3.5 (Successful guess)**.** *For a fixed guess and a sequence P, the guess is* successful for P *if $|\varphi(P \setminus U_P)| \le N$, where U_P is the protected set in Equation* (14)*.*

Maintain $\mathsf{Sparse}(Z)$**.** We process the residual points $R_Z \coloneqq Z \setminus U_Z$. For $A \subseteq Z$, let $\mathrm{first}_M(A)$ denote its first $\min\{M, |A|\}$ points in the order of Z. For each bucket u, define $R_{Z,u} \coloneqq R_Z \cap \varphi^{-1}(u)$, $H_{Z,u} \coloneqq \mathrm{first}_M(R_{Z,u})$, and $H_Z \coloneqq \bigcup_u H_{Z,u}$. The Sparse summary is $\mathcal{SS}_Z \coloneqq (H_Z, \tau_Z)$, where $\tau_Z \coloneqq \min\{t : |\varphi((R_Z)_{\le t})| > N\}$ and the minimum of the empty set is $+\infty$. A prefix t is successful exactly when $t < \tau_Z$. Define $\mathsf{Sparse}(Z) \coloneqq \mathsf{Sparse}_{\mathrm{off}}(R_Z)$.

Prefix Query. For a successful prefix t, let $H_t \coloneqq (H_Z)_{\le t}$ and $H_{t,u} \coloneqq H_t \cap \varphi^{-1}(u)$. Then

$$\mathsf{SparseQuery}(\mathcal{SS}_Z, t) \coloneqq \bigcup_{\substack{u \in \varphi(H_t) \\ |H_{t,u}| < M}} H_{t,u} = \mathsf{Sparse}(Z_{\le t}).$$

The equality follows from IsolateQuery and the first-M rule. Also define $\mathsf{Outer}(Z) \coloneqq \mathsf{Isolate}(Z) \cup \mathsf{Sparse}(Z)$ and $\mathsf{Inner}(Z) \coloneqq Z \setminus \mathsf{Outer}(Z)$.

Merging. The first M parent residual points in each bucket lie among $H_X \cup H_Y \cup R_I$. Thus $\mathsf{SparseMerge}(\mathcal{SS}_X, \mathcal{SS}_Y, R_I)$ keeps their first M points per bucket, determines the parent failure index, and returns $\mathcal{SS}_{X \circ Y}$ together with the remaining explicit points as an ordered release stream R_H. A point already beyond a child head cannot enter the parent head, so this step never recovers a compressed point.

Maintain $\mathsf{Vanilla}$**.** Let $T_Z \coloneqq R_Z \setminus H_Z$ be the tail. We use the online coreset construction of Woodruff et al. (2023) to maintain $\mathcal{V}_Z$ for T_Z. It supports the prefix query $\mathsf{VanillaQuery}(\mathcal{V}_Z, t)$ and the merge $\mathsf{VanillaMerge}(\mathcal{V}_X, \mathcal{V}_Y, R_H)$, where R_H is the release stream from SparseMerge. Details of these operations appear in Section B.3.

Altogether, the mergeable coreset summary is

$$\mathcal{D}_Z \coloneqq (\mathcal{IS}_Z, \mathcal{SS}_Z, \mathcal{V}_Z) \equiv (\mathcal{I}_Z, U_Z, H_Z, \mathcal{V}_Z, \tau_Z), \qquad Z = U_Z \cup H_Z \cup T_Z \tag{16}$$

at every prefix before τ_Z. For $P = Z_{\le t}$, let $I_t \coloneqq \mathsf{IsolateQuery}(\mathcal{IS}_Z, t)$, $S_t \coloneqq \mathsf{SparseQuery}(\mathcal{SS}_Z, t)$, and $V_t \coloneqq \big((H_Z)_{\le t} \setminus S_t\big) \cup \mathsf{VanillaQuery}(\mathcal{V}_Z, t)$. Then the prefix query returns $I_t \cup S_t \cup V_t$. The first two layers are exact, while only T_Z is compressed through $\mathcal{V}_Z$. During a merge, an occurrence may move from U_Z to H_Z through R_I, or from H_Z to T_Z through R_H, but never in the opposite direction. Thus no merge deletes an occurrence from, or recovers an occurrence from, an already compressed vanilla coreset.

The protected points number $\widetilde{O}(r)$, the bucket heads number $O(NM)$, and the vanilla tail coreset has size $N_{\mathcal{V}}$. These bounds apply to the parent after every merge. Further algorithmic and proof details appear in Sections B, B.2 and C. For a successful guess below the optimum, coreset correctness follows from Lemma 3.6. □

3.3 Correctness and optimum guesses

The following two lemmas complete the construction. Their proofs appear in Section D.

Lemma 3.6. *Consider a successful prefix $Z_{\le t}$ with $\widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(Z_{\le t})$. If $\mathsf{VanillaQuery}(\mathcal{V}_Z, t)$ is an α-coreset for $(T_Z)_{\le t}$, then $\mathsf{PrefixCoreset}(\mathcal{D}_Z, t)$ is an $O(\alpha+\varepsilon)$-coreset for (k, z, m)-clustering on $Z_{\le t}$.*

Lemma 3.7. *There is a deterministic procedure that, given L, d, Δ, z, generates a family of*

$$L_{\mathrm{opt}} = 1 + O\Big(\log_{z+1}\big(2^z L(\sqrt{d}\,\Delta)^z\big)\Big)$$

guesses, including the zero-cost case, such that every prefix of every input sequence from $[\Delta]^d$ of length at most L has a guess in $[\mathrm{OPT}_z^{(m)}/(z+1), \mathrm{OPT}_z^{(m)}]$. Running the fixed-guess construction for all guesses and returning the smallest successful guess gives an $O(\varepsilon)$-mergeable coreset summary. The returned coreset has the size bound of one fixed-guess output.

We now use this implementation to prove the mergeable coreset summary lemma.

Proof of Lemma 3.3. Apply Lemmas 3.4 and 3.7 with length bound L, using the same hash and sampling priorities at every merge node. A correct-scale guess succeeds for each prefix, and Lemma 3.6 shows that the smallest successful guess returns an $O(\varepsilon)$-coreset. The construction is mergeable because each fixed-guess summary is mergeable.

For one guess, a prefix output consists of $\widetilde{O}(r)$ protected points, $O(NM)$ bucket heads, and a vanilla prefix coreset of size at most $N_{\mathcal{V}}(L, \varepsilon, \delta_V)$. The query returns the output of one guess. Substituting Equations (5) and (9), $N = 2k\Lambda$, $\Gamma = O(d)$, and $\Lambda = O(d \log d)$ gives Equation (11). Allocating failure probability $\delta / \operatorname{poly}(L, d, z, \log \Delta)$ to the polynomially many relevant prefix and merge events gives simultaneous success with probability at least $1 - \delta$. □

4 EXPERIMENTS

Setup. We evaluate our sliding-window coreset on robust Euclidean k-means ($z = 2$) across a range of coreset sizes. We use five real data sets—Adult, Bank, Diabetes, Athlete, and Census—summarized in Table 1. Since these data sets were not designed for clustering with outliers, we augment each stream with a small number of synthetic points placed far from the original data. The stream has length $4W$ and proceeds in four consecutive phases of W arrivals each. In phase $s \in \{0, 1, 2, 3\}$, we inject the synthetic points at evenly spaced reserved positions, so that a complete active window in this phase contains approximately $t = k + m - s$ of them: after trimming, $k - s$ remain, nearly one per center, so a coreset that loses any injected point incurs a large additional cost.

Evaluation. For each of five independent stream seeds (2026–2030), we place one checkpoint in each of the four outlier-density phases. At each checkpoint, we solve weighted trimmed k-means on the extracted coreset using a weighted variant of k-means-- (Chawla & Gionis, 2013), and evaluate the resulting centers on every point of the original active window, removing total mass m. All methods solve the same trimmed objective on their weighted outputs. The reported empirical approximation ratio is the resulting cost divided by the minimum cost attained at the same coreset size and checkpoint, where the candidates consist of every solution produced by the methods in our comparison and an additional offline solver. This solver receives all points in the active window as input and returns the best solution among 12 restarts, so the denominator is a best-found empirical reference rather than a certified optimum.

Baselines. We compare Ours-SW with three methods. The Uniform baseline samples without replacement from the active window, using exactly as many points as Ours-SW returns; we repeat the sampling three times per seed and checkpoint and average the results. HJLW-Offline is a full-window Euclidean robust coreset based on the importance sampler of Huang et al. (2026). It rebuilds from the complete active window and matches the Ours-SW coreset size exactly; we use it as an empirical offline baseline. Vanilla-Online applies the vanilla online coreset of Woodruff et al. (2023) directly, without the robust isolation layer. It maintains 23 fixed-rate summaries during arrivals and selects the one with the closest coreset size at each checkpoint; its mean relative size mismatch is 4–5% across data sets. We reset Vanilla-Online at the boundary of each of the four disjoint tested windows, so it does not support arbitrary-time sliding-window queries.

Each curve contains ten coreset sizes. The first nine use geometrically increasing sampling rates $\gamma \in \{0.025, 0.042, 0.071, 0.119, 0.2, 0.336, 0.566, 0.951, 1.6\}$; the tenth is the exact full-window coreset. Consequently, all four methods attain a ratio of 1 at the right endpoint. We choose window sizes so that the mean coreset size at the two endpoints differs by approximately one order of magnitude on every data set.

Results. Ours-SW improves the mean empirical ratio over Uniform by 28–82% and over Vanilla-Online by 15–64% across the five data sets (Table 2), and it outperforms Vanilla-Online at 37 of the 45 data-set/size combinations below the exact endpoint. This gap is as expected: through its isolation layer, Ours-SW protects points in sparse regions and retains them exactly, so the injected outliers survive compression and can be removed by the down-stream solver; ordinary k-means coresets have no such mechanism and lose them. Compared with HJLW-Offline, the offline coreset of Huang et al. (2026) with the best known theoretical size bound, Ours-SW incurs a 5–12% higher mean ratio on each data set; in return, it answers sliding-window queries online, while HJLW-Offline must rebuild a

Table 1: Summary of the five real data sets. d: number of retained numerical coordinates; N: number of source records; W: window size; k: number of centers; m: number of removed outliers.

Data set	Domain	d	N	W	k	m
Adult	Income demographics	6	48,842	7,500	10	30
Bank	Telemarketing	10	41,188	9,000	8	35
Diabetes	Hospital encounters	8	101,766	10,000	12	50
Athlete	Wearable accelerometry	3	1,140,000	30,000	15	70
Census	Demographics	13	2,458,285	20,000	20	100

Table 2: Robust k-means mean quality across ten sizes, averaged over five stream seeds and four checkpoints per seed. Lower is better.

Data set	Ours-SW	HJLW-Offline	Vanilla-Online	Uniform
Adult	1.204	1.063	2.392	2.431
Bank	1.220	1.127	1.683	1.696
Diabetes	1.141	1.080	1.942	2.206
Athlete	1.160	1.049	3.257	6.311
Census	1.145	1.087	1.349	1.610

coreset from the complete active window at every query. We believe this modest loss is well justified by the sliding-window support it enables.

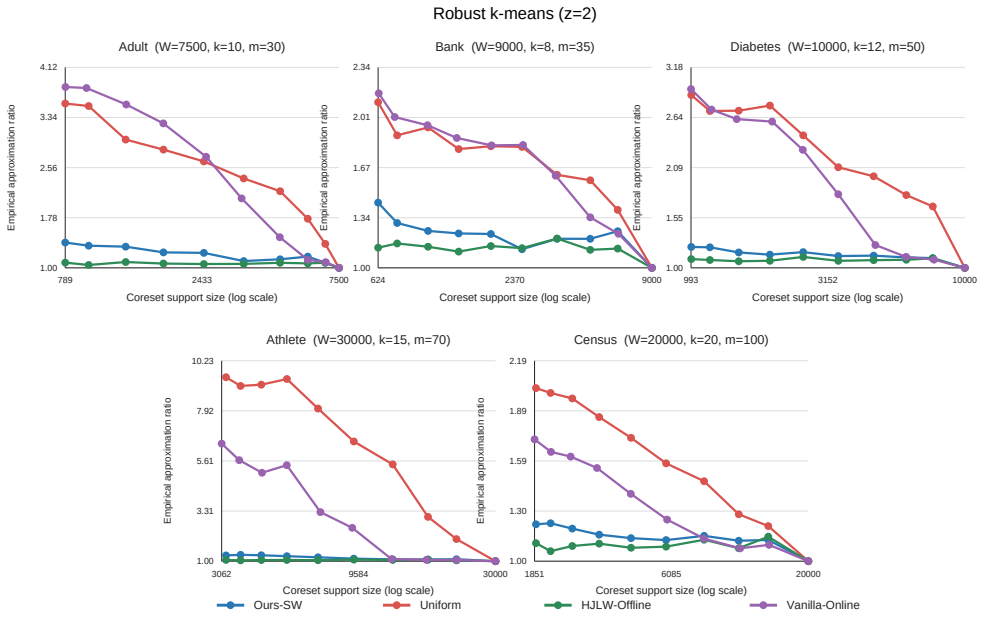

Figure 1: Robust k-means empirical approximation ratio versus realized coreset size. Each point averages five stream seeds, each with four active-window checkpoints (and, for Uniform, three samples per checkpoint). The horizontal axis is logarithmic. The curves need not decrease monotonically: each size uses an independently sampled coreset, and the trimmed local-search solver may converge to different minima.

5 AI Disclosure

In this work, we used generative AI tools for refining writing, finding related works, implementing algorithms for experiments and drawing experiment figures.

We have not used generative AI tools for developing theoretical models, formulating mathematical claims, providing critical ingredients for proving mathematical claims, designing experiments, data analysis, generating/cleaning/reformating data sets,

We have reviewed all AI-assisted work, including text, claims or codes produced with the aid of generative AI and take responsibility for the final content of this work. LLM-generated code was reviewed and tested for correctness by all authors.

References

Akanksha Agrawal, Tanmay Inamdar, Saket Saurabh, and Jie Xue. Clustering what matters: Optimal approximation for clustering with outliers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pp. 6666–6674, 2023. doi: 10.1609/aaai.v37i6.25818.

Brian Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pp. 234–243, 2003.

Michele Borassi, Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Sliding window algorithms for k-clustering problems. In *Advances in Neural Information Processing Systems 33*, 2020.

Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering on sliding windows in polylogarithmic space. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pp. 350–364, 2015.

Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1374–1390, 2016.

Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 642–651, 2001.

Sanjay Chawla and Aristides Gionis. k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*, pp. 189–197, 2013. doi: 10.1137/1.9781611972832.21.

Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 826–835, 2008.

Vincent Cohen-Addad, Shaofeng H.-C. Jiang, Qiaoyuan Yang, Yubo Zhang, and Samson Zhou. Fair clustering in the sliding window model. In *International Conference on Learning Representations (ICLR)*, 2025.

Artur Czumaj, Arnold Filtser, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. Streaming facility location in high dimension via geometric hashing, 2022. URL `https://arxiv.org/abs/2204.02095`. Full version of the FOCS 2022 paper.

Mark de Berg, Morteza Monemizadeh, and Yu Zhong. k-center clustering with outliers in the sliding-window model. In *29th Annual European Symposium on Algorithms (ESA)*, pp. 13:1–13:13, 2021. doi: 10.4230/LIPIcs.ESA.2021.13.

Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Improved sliding window algorithms for clustering and coverage via bucketing-based sketches. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 3005–3042, 2022.

Dan Feldman and Leonard J. Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1343–1354, 2012. doi: 10.1137/1.9781611973099.106.

Anupam Gupta, Benjamin Moseley, and Rudy Zhou. Structural iterative rounding for generalized k-median problems. *Mathematical Programming*, 212:581–634, 2025. doi: 10.1007/s10107-024-02119-7.

Lingxiao Huang, Shaofeng H.-C. Jiang, Jian Li, and Xuan Wu. ε-coresets for clustering (with outliers) in doubling metrics. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 814–825, 2018.

Lingxiao Huang, Shaofeng H.-C. Jiang, Jianing Lou, and Xuan Wu. Near-optimal coresets for robust clustering. In *International Conference on Learning Representations (ICLR)*, 2023.

Lingxiao Huang, Jian Li, and Xuan Wu. On optimal coreset construction for euclidean (k, z)-clustering. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1594–1604, 2024. doi: 10.1145/3618260.3649707.

Lingxiao Huang, Zhenyu Jiang, Yi Li, and Xuan Wu. Towards tight robust coresets for k-medians clustering. In *53rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 112:1–112:21, 2026. doi: 10.4230/LIPIcs.ICALP.2026.112.

Shaofeng H.-C. Jiang and Jianing Lou. Coresets for robust clustering via black-box reductions to vanilla case. In *52nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 101:1–101:18, 2025. doi: 10.4230/LIPIcs.ICALP.2025.101.

Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 339–352, 2018. doi: 10.1145/3188745.3188882.

Paolo Pellizzoni, Andrea Pietracaprina, and Geppino Pucci. k-center clustering with outliers in sliding windows. *Algorithms*, 15(2):52, 2022. doi: 10.3390/a15020052.

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*, 2023.

Zhen Zhang, Junyu Huang, and Qilong Feng. Faster approximation schemes for (constrained) k-means with outliers. In *49th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 306, pp. 84:1–84:17, 2024. doi: 10.4230/LIPIcs.MFCS.2024.84.

A Adaptations of Existing Results

We adapt two existing results to our setting: the reduction of Jiang & Lou (2025) and the sliding-window framework of Woodruff et al. (2023). The latter adaptation uses only the summary interface, not its implementation.

A.1 Adapting Reduction I

We use the first reduction of Jiang & Lou (2025) without changing its analysis, but our mergeable representation has a different form. Their streaming reduction returns

$$G \cup X_S \cup S_D,$$

where G is the isolated sample, X_S contains the sparse residual buckets exactly, and S_D is a vanilla coreset for the union X_D of the dense residual buckets. Our prefix query instead returns

$$U \cup H \cup V.$$

There are two differences in the representation. First, U may strictly contain the current isolated sample because it also protects occurrences that appeared in an earlier prefix sample. Second, H contains not only every sparse bucket but also the first M occurrences of every dense bucket, while V represents only the remaining tail. Thus, with $H_{\text{in}} := H \cap \mathsf{Inner}(X)$ denoting the heads of the dense buckets and T their tails, the analogue of S_D is $H_{\text{in}} \cup V$, not V alone.

These differences concern how the three stages are made mergeable; they do not require a new analysis of Reduction I. The protected points and small residual buckets form $\mathsf{Outer}(X)$, and $H_{\text{in}} \cup V$ is a vanilla coreset for the dense set $\mathsf{Inner}(X)$, which is partitioned into H_{in} and T. We therefore only need the known composability and dense-part guarantees of Jiang & Lou (2025), followed by the short adaptation below. The isolated-point extraction analysis appears in Section D, and the merge-specific arguments appear in Sections B.2 and C.

It is convenient to use an additive-error version of Definition 2.1.

Definition A.1 (Additive-error coreset with outliers)**.** *A weighted set S supported on X is a (β, η)-coreset with outliers for X up to outlier budget m if, for every $|C| = k$ and $0 \le h \le m$,*

$$\left|\mathrm{cost}_z^{(h)}(S, C) - \mathrm{cost}_z^{(h)}(X, C)\right| \le \beta\, \mathrm{cost}_z^{(h)}(X, C) + \eta. \tag{17}$$

A.1.1 Known ingredients

The first ingredient is the outlier-cost convolution underlying the composability result of Jiang & Lou (2025, Fact 2.3).

Lemma A.2 (Jiang & Lou, 2025)**.** *Let $X_1, \dots, X_s$ be pairwise disjoint weighted sets. For every center set C and $h \ge 0$,*

$$\mathrm{cost}_z^{(h)}\left(\bigcup_{i=1}^{s} X_i, C\right) = \min_{\substack{h_1,\dots,h_s \ge 0 \\ h_1+\cdots+h_s \le h}} \sum_{i=1}^{s} \mathrm{cost}_z^{(h_i)}(X_i, C). \tag{18}$$

Proof. Any feasible removal from the union restricts to removals of masses $h_1, \dots, h_s$ from the individual sets with $\sum_i h_i \le h$. Conversely, feasible removals from the individual sets with total mass at most h form a feasible removal from the union. Minimizing in the two directions proves the identity. □

Corollary A.3 (Jiang & Lou, 2025). *For every $i \in [s]$, let S_i be a (β, η_i)-coreset with outliers for X_i up to budget m. Then $\bigcup_i S_i$ is a $(\beta, \sum_i \eta_i)$-coreset with outliers for $\bigcup_i X_i$ up to budget m.*

Proof. The formulation with an "at most" outlier budget is important here. By Lemma A.2, both union costs minimize over the same set of nonnegative budget allocations $h_1 + \cdots + h_s \le h$, regardless of the total weights of the coresets. Using an optimal allocation for the input gives

$$\mathrm{cost}_z^{(h)}\left(\bigcup_i S_i, C\right) \le (1+\beta)\,\mathrm{cost}_z^{(h)}\left(\bigcup_i X_i, C\right) + \sum_i \eta_i.$$

Using an optimal allocation for the coreset and applying the lower guarantee term by term similarly gives

$$\mathrm{cost}_z^{(h)}\left(\bigcup_i S_i, C\right) \ge (1-\beta)\,\mathrm{cost}_z^{(h)}\left(\bigcup_i X_i, C\right) - \sum_i \eta_i.$$

Together these inequalities prove the claim. □

We say that a partition $\mathcal{P}$ of a weighted set X is λ-bounded if $\mathrm{diam}(P) \le \lambda$ for every $P \in \mathcal{P}$. It is (m, ε)-dense if

$$w(P) \ge (1+\varepsilon^{-1})m \qquad \text{for every } P \in \mathcal{P}. \tag{19}$$

Lemma A.4 (Jiang & Lou, 2025, Lemma 3.3). *Let $0 < \alpha, \varepsilon \le 1/2$, and suppose X admits an (m, ε)-dense, λ-bounded partition. Let S be a weighted set supported on X with $w(S) \ge m$ such that, for every $|C| = k$,*

$$\left|\mathrm{cost}_z^{(0)}(S, C) - \mathrm{cost}_z^{(0)}(X, C)\right| \le \alpha\,\mathrm{cost}_z^{(0)}(X, C). \tag{20}$$

Then S is an

$$\left(O(\alpha+\varepsilon),\ 2^{O(z\log(z+1))} m\lambda^z\right) \tag{21}$$

coreset with outliers for X up to budget m.

The accuracy parameter is separated into the vanilla-coreset error α and the density slack ε. In particular, the lemma does not assume that the vanilla coreset preserves the total input weight. The weaker condition $w(S) \ge m$ only ensures that all relevant outlier budgets are feasible in the exact-budget formulation of that lemma.

A.1.2 The adapted analytic guarantee

Write $c_z := (z+1)^{-c}$, where c is the sufficiently large absolute constant used in Equation (5). We choose c large enough that the factor $2^{O(z\log(z+1))} c_z^z$ below is sufficiently small.

Lemma A.5. *Let $X = \mathsf{Outer}(X) \cup \mathsf{Inner}(X)$, where the two sets are disjoint, $\mathsf{Outer}(X)$ is stored exactly, and the nonempty hash buckets of $\mathsf{Inner}(X)$ form an (m, ε)-dense, λ-bounded partition. Let V_{in} be a vanilla α-coreset for $\mathsf{Inner}(X)$ supported on $\mathsf{Inner}(X)$, and suppose $w(V_{\mathrm{in}}) \ge m$ whenever $\mathsf{Inner}(X)$ is nonempty. Then $\mathsf{Outer}(X) \cup V_{\mathrm{in}}$ is an*

$$\left(O(\alpha+\varepsilon),\ 2^{O(z\log(z+1))} m\lambda^z\right)$$

coreset with outliers for X up to budget m.

In particular, suppose

$$\lambda := c_z \left(\frac{\varepsilon \widehat{\mathrm{OPT}}}{m}\right)^{1/z}, \qquad \widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(X),$$

Then $\mathsf{Outer}(X) \cup V_{\mathrm{in}}$ is an $O(\alpha+\varepsilon)$-coreset in the sense of Definition 2.1.

Proof. If $\mathsf{Inner}(X)$ is empty, then $\mathsf{Outer}(X) = X$ and the claim is immediate. Otherwise $w(V_{\mathrm{in}}) \geq m$, so all budgets used below are feasible in both $\mathsf{Inner}(X)$ and V_{in}. By Lemma A.4, V_{in} has the claimed outlier guarantee for $\mathsf{Inner}(X)$. The exact set $\mathsf{Outer}(X)$ is a $(0, 0)$-coreset with outliers for itself. Therefore, Corollary A.3 proves the first claim.

Define the additive error by $\eta := 2^{O(z \log(z+1))} m\lambda^z$. With a sufficiently small choice of c_z,

$$\eta \leq \varepsilon \widehat{\mathrm{OPT}} \leq \varepsilon \, \mathrm{OPT}_z^{(m)}(X).$$

For every C and every $h \leq m$,

$$\mathrm{OPT}_z^{(m)}(X) \leq \mathrm{cost}_z^{(m)}(X, C) \leq \mathrm{cost}_z^{(h)}(X, C).$$

Hence the additive error is at most $\varepsilon \, \mathrm{cost}_z^{(h)}(X, C)$ and can be absorbed into the multiplicative term. This proves the second claim. □

The original form of Reduction I follows directly from Jiang & Lou (2025, Lemma 3.3). For our summary, $\mathsf{Outer}(X)$ is U together with the small residual buckets; take $V_{\mathrm{in}} := H_{\mathrm{in}} \cup V$, where H_{in} consists of the exact dense-bucket heads. The proof of Lemma 3.6 verifies that $H_{\mathrm{in}} \cup V$ is a vanilla coreset for the complete dense set. If this set is nonempty, its exact head has mass at least $M > m$; every remaining bucket also has diameter at most λ. Hence Lemma A.5 applies without any requirement that V preserve the tail mass. The separate claim that a correct-scale guess succeeds follows from the structural exceptional-set lemma and the isolated-sampling analysis in Section D.

A.2 From Mergeable Coreset Summaries to Sliding Windows

Proof of Lemma 3.2. Apply Algorithm 4 and the proof of Theorem 4.1 of Woodruff et al. (2023), replacing each online-coreset block by a mergeable coreset summary. When their framework recompresses blocks, we invoke Merge on their summaries in reverse input order. This order is correct because

$$\mathrm{rev}(X \circ Y) = \mathrm{rev}(Y) \circ \mathrm{rev}(X),$$

and the merge guarantee in Definition 3.1 yields a summary of the same type for this ordered concatenation. Thus the block invariant in their proof is unchanged.

Their query uses the online-coreset prefix corresponding to the suffix of the oldest block intersecting the window. We replace this operation by PrefixCoreset; all newer blocks are queried in full. The union of these outputs is a coreset with outliers for the window by Corollary A.3. In particular, the convolution identity in Lemma A.2 handles the distribution of the outlier budget among blocks.

The remaining analysis of Woodruff et al. (2023) is unchanged. Truncating their hierarchy at the window scale leaves $O(\log W)$ active levels, so internal accuracy $\Theta(\varepsilon/\log W)$ gives total error ε, and assigning failure probability $\delta/\operatorname{poly}(n)$ to the $\operatorname{poly}(n)$ summary instances gives simultaneous success probability at least $1 - \delta$. The window answer is the union of $O(\log W)$ block queries, each of size at most $N_{\mathrm{coreset}}(W, \Theta(\varepsilon/\log W), \delta/\operatorname{poly}(n))$, which is Equation (10).

□

B Summary Algorithms and Correctness

We give the algorithms underlying Lemma 3.4 and prove that their composition preserves the summary invariant. The summary $\mathcal{D}_Z = (\mathcal{IS}_Z, \mathcal{SS}_Z, \mathcal{V}_Z)$ has three components, treated in turn in the next three subsections. The sampling and history merge guarantees are stated here and proved in Section C. All set operations are on input points.

B.1 The Isolate summary

Algorithm 1 Merging isolated samples

procedure IsolateSampleMerge(S_X, S_Y, r)
  $S \leftarrow$ empty ordered list
  for $i = 1, \dots, r$ **do**
    $C \leftarrow (S_X[1:i] \cup S_Y[1:i]) \setminus S$
    if $C = \varnothing$ **then**
      break
    append $\arg\min_{x \in C} q_i(x)$ to S
  return S

Lemma B.1. *For sequences X and Y, Algorithm 1 returns $S_r(X \circ Y)$. In particular, $S_r(X \circ Y) \subseteq S_r(X) \cup S_r(Y)$. It stores $O(r)$ points and takes $O(r^2)$ time.*

The proof is deferred to Section C.1.

Algorithm 2 lifts this operation from the final sample to all prefixes. Define $\mathsf{State}(\mathcal{I}, t) \coloneqq$ the most recent isolated state recorded by $\mathcal{I}$ at or before index t, and $\mathsf{Last}(\mathcal{I}) \coloneqq \mathsf{State}(\mathcal{I}, +\infty)$. The stream R_I contains protected points of the later child that the parent no longer needs to protect. The loop processes, in increasing order, every index at which $\mathcal{I}_Y$ changes or a point first enters U_Y.

Algorithm 2 Merging isolated histories

procedure IsolateHistoryMerge($\mathcal{I}_X, U_X, \mathcal{I}_Y, U_Y, |X|, r$)
  $\mathcal{I} \leftarrow \mathcal{I}_X$; $U \leftarrow U_X$; $R_I \leftarrow \varnothing$
  $A \leftarrow \mathsf{Last}(\mathcal{I}_X)$; $S^{\mathrm{old}} \leftarrow A$
  for all these indices j in Y **do**
    $t \leftarrow |X| + j$; $B \leftarrow \mathsf{State}(\mathcal{I}_Y, j)$
    $S \leftarrow$ IsolateSampleMerge(A, B, r)
    if $S \neq S^{\mathrm{old}}$ **then**
      append the change from S^{old} to S at index t to $\mathcal{I}$
    for all points p that first enter U_Y at index j **do**
      if $p \in S$ **then**
        insert p into U with first-appearance index t
      else
        append (p, t) to R_I
    $S^{\mathrm{old}} \leftarrow S$
  return $(\mathcal{I}, U, R_I)$

Lemma B.2. *Algorithm 2 records the exact isolated state of $X \circ Y$ at every prefix. Its protected set is $U_{X \circ Y} = \bigcup_t S_r((X \circ Y)_{\le t})$, and R_I contains exactly the occurrences protected by the later child but not by the parent, each at its index in $X \circ Y$. With probability at least $1 - \delta_I$, the history, $U_{X \circ Y}$, and R_I contain $\widetilde{O}(r)$ records and points in total.*

The proof is deferred to Section C.2.

B.2 The Sparse summary

The history merge determines which protected points enter the residual input. We now show how to update the bucket heads and pass the remaining points to the vanilla tail. For a sequence Z, let $R_Z \coloneqq Z \setminus U_Z$ be the post-protection residual set, and write $R_{Z,y} \coloneqq R_Z \cap \varphi^{-1}(y)$ for its restriction to bucket y. Define $H_{Z,y} \coloneqq \mathrm{first}_M(R_{Z,y})$ and write $H_Z \coloneqq \bigcup_y H_{Z,y}$ and $T_Z \coloneqq R_Z \setminus H_Z$. In the partition identities below, the sets on each side are disjoint. These identities hold at prefixes before the relevant failure index.

Algorithm 3 merges the residual bucket heads. For a bucket y, the list $R_I[y]$ contains isolated-release events mapped to y. The output R_H consists exactly of explicit head events moved to the parent tail.

Algorithm 3 Merging bucket heads

procedure HEADMERGE($H_X, H_Y, R_I, \tau_X, \tau_Y, |X|$)
  $H \leftarrow$ empty bucket dictionary; $R_H \leftarrow \varnothing$
  for all $y \in \mathsf{keys}(H_X) \cup \mathsf{keys}(H_Y) \cup \mathsf{keys}(R_I)$ **do**
    $K \leftarrow$ merge $H_X[y]$, $H_Y[y]$, and $R_I[y]$ in sequence order
    $H[y] \leftarrow$ the first $\min\{M, |K|\}$ events of K
    append $K \setminus H[y]$ to R_H
  $\tau \leftarrow$ earliest of τ_X, $|X| + \tau_Y$, and the first index with $N + 1$ distinct keys
  discard from H and R_H all events at or after τ
  return (H, R_H, τ)

Lemma B.3. *At every prefix before failure, Algorithm 3 stores exactly the first M residual events of every nonempty bucket and records the first index at which more than N buckets occur. If R_H is its release stream, then*

$$T_{X\circ Y} = T_X \cup T_Y \cup R_H. \tag{22}$$

The head contains $O(NM)$ points.

By Equation (28), the isolated-history merge immediately gives

$$R_{X\circ Y} = R_X \cup R_Y \cup R_I. \tag{23}$$

Released points are ordered by their indices in the parent sequence. Thus the prefix residual sets form an insertion-only sequence in that order.

Proof of Lemma B.3. Let $K \coloneqq H_X \cup H_Y \cup R_I$. If $p \in T_X$, then at least M occurrences of its bucket precede p in R_X. By Equation (23), they remain parent residual occurrences preceding p in the parent sequence, so p cannot enter the parent head. The same holds for $p \in T_Y$. Hence $H_{X\circ Y} \subseteq K$, and retaining the first M candidates of each bucket constructs the parent head exactly.

The first occurrence of every parent residual bucket also belongs to K: it is either a child-head occurrence or is newly exposed through R_I. Therefore the ordered candidates detect exactly when the $(N + 1)$-st residual bucket first appears. Moreover, a child failure cannot disappear after a merge because Equation (23) preserves its residual occurrences in the parent. Taking the first new or inherited failure index thus gives the exact $\tau_{X\circ Y}$.

Finally, using $R_X = H_X \cup T_X$ and $R_Y = H_Y \cup T_Y$ in Equation (23) gives

$$R_{X\circ Y} = T_X \cup T_Y \cup K.$$

The head merge partitions K into $H_{X\circ Y}$ and its release stream R_H; removing the parent head proves

$$T_{X\circ Y} = T_X \cup T_Y \cup R_H,$$

which is Equation (22). Before failure there are at most N residual buckets, each storing at most M head occurrences, so the head contains $O(NM)$ points. □

B.3 THE VANILLA SUMMARY

The remaining points passed to the vanilla tail are summarized by the online coreset of Woodruff et al. (2023). An ε-online coreset $\mathcal{V}_Z$ is a weighted subsequence such that $(\mathcal{V}_Z)_{\le t}$ is an ε-coreset for (k, z)-clustering on $Z_{\le t}$ for every t; thus $\mathsf{VanillaQuery}(\mathcal{V}_Z, t) \coloneqq (\mathcal{V}_Z)_{\le t}$. Their Theorem 1.3 gives a size bound

$$N_{\mathcal{V}}(L, \varepsilon, \delta) = O\left(\frac{k}{\min\{\varepsilon^4, \varepsilon^{2+z}\}} \operatorname{polylog}\left(\frac{L\Delta}{\varepsilon\delta}\right)\right)$$

for sequences of length at most L, with failure probability at most δ.

Lemma B.4. *There is a randomized algorithm* VanillaMerge *that, given online α-coresets $\mathcal{V}_X, \mathcal{V}_Y$ for two parts of a sequence of length at most L and its remaining points R, returns an online $(\alpha + \beta + \alpha\beta)$-coreset of size at most $N_{\mathcal{V}}(L, \beta, \delta)$, with probability at least $1 - \delta$ conditioned on the child guarantees.*

Proof. Apply the online-coreset algorithm to the weighted supports of $\mathcal{V}_X, \mathcal{V}_Y$ and the explicit points of R in their original order. The claim follows as in the proof of Theorem 4.1 of Woodruff et al. (2023). □

B.4 SUMMARY CONSTRUCTION, MERGING, AND QUERIES

The IsolateMerge and SparseMerge interfaces of Section 3.2 are implemented by Algorithms 2 and 3, respectively. They return the release streams R_I and R_H. The VanillaMerge interface is guaranteed by Lemma B.4.

Algorithm 4 The mergeable coreset summary

procedure SINGLETONSUMMARY(x)
  let $\mathcal{I}$ record the isolated state $\{x\}$ at index 1
  return $((\mathcal{I}, \{(x, 1)\}), (\varnothing, +\infty), \mathsf{EmptyVanilla})$

procedure SUMMARYMERGE($\mathcal{D}_X, \mathcal{D}_Y$)
  $(\mathcal{IS}, R_I) \leftarrow$ ISOLATEMERGE$(\mathcal{IS}_X, \mathcal{IS}_Y)$
  $(\mathcal{SS}, R_H) \leftarrow$ SPARSEMERGE$(\mathcal{SS}_X, \mathcal{SS}_Y, R_I)$
  $\mathcal{V} \leftarrow$ VANILLAMERGE$(\mathcal{V}_X, \mathcal{V}_Y, R_H)$ ▷ sequence order
  return $(\mathcal{IS}, \mathcal{SS}, \mathcal{V})$

procedure PREFIXCORESET($\mathcal{D}_Z, t$)
  if $t \geq \tau_Z$ **then**
    return FAIL
  $U \leftarrow$ ISOLATEQUERY$(\mathcal{IS}_Z, t)$
  $S \leftarrow$ SPARSEQUERY$(\mathcal{SS}_Z, t)$
  $H \leftarrow$ head events whose indices are at most t
  $V \leftarrow (H \setminus S) \cup$ VANILLAQUERY$(\mathcal{V}_Z, t)$
  return $U \cup S \cup V$

Lemma B.5. *Suppose the two inputs to* SUMMARYMERGE *are valid summaries and their concatenation has length at most L. Its output is a summary of the same type and, at every prefix before τ_Z,*

$$Z_{\leq t} = (U_Z)_{\leq t} \cup (H_Z)_{\leq t} \cup (T_Z)_{\leq t}.$$

The three sets partition the input points of the prefix. Moreover, $\mathcal{V}_Z$ is a vanilla online coreset for T_Z, and every prefix coreset output has size at most $\widetilde{O}(r + NM) + N_{\mathcal{V}}$.

Proof of Lemma B.5. We induct on the merge tree. The claim is immediate for a singleton. For child sequences X, Y, Lemma C.4 partitions $U_X \cup U_Y$ into $U_{X\circ Y}$ and R_I. The head merge then partitions $H_X \cup H_Y \cup R_I$ into $H_{X\circ Y}$ and R_H, while Equation (22) gives $T_{X\circ Y} = T_X \cup T_Y \cup R_H$. Hence, at every valid prefix,

$$(X \circ Y)_{\leq t} = (U_{X\circ Y})_{\leq t} \cup (H_{X\circ Y})_{\leq t} \cup (T_{X\circ Y})_{\leq t}.$$

The release stream is ordered by indices in $X \circ Y$. Therefore $\mathsf{VanillaMerge}(\mathcal{V}_X, \mathcal{V}_Y, R_H)$ receives, in sequence order, exactly the two child-tail prefixes and $(R_H)_{\leq t}$, whose union is $(T_{X\circ Y})_{\leq t}$. The ordered merge interface of Lemma B.4 thus produces a vanilla coreset for every parent tail prefix.

By Lemma C.3, the protected set contains $\widetilde{O}(r)$ points with the allocated high probability. The head contains $O(NM)$ points by Lemma B.3, and every vanilla prefix coreset has at most $N_{\mathcal{V}}$ points, where L bounds the parent sequence length. Every prefix output contains only these three parts, proving its $\widetilde{O}(r + NM) + N_{\mathcal{V}}$ size bound. □

Proof of Lemma 3.4. The IsolateMerge operation gives the exact parent history and protected set. The SparseMerge operation constructs the parent heads and releases the remaining explicit points to the tail. Applying VanillaMerge then gives the summary invariant of Lemma B.5, including all prefix partitions and the stated coreset-size bound. Only explicit protected points and heads are released; no operation recovers or deletes an individual point from a child vanilla coreset. □

B.5 AUXILIARY SPACE

We count the space used by our summary apart from the vanilla online-coreset black box; the latter may store a bicriteria solution in addition to its output coreset. For one guess and one block, the isolated history and protected set use $\widetilde{O}(r)$ records by Lemma C.3, while the bucket heads use $O(NM)$ records by Lemma B.3. Head events after the failure index are discarded. During a merge, the release streams consist of explicit points from the children, so they do not increase this bound asymptotically.

There are L_{opt} guesses and $O(\log W)$ active blocks. Thus, excluding the vanilla black box, the algorithm stores

$$\widetilde{O}(r + NM) = \widetilde{O}\left(2^{O(z\log(z+1))}\,\frac{(d^z + kd)m}{\varepsilon}\right)$$

point and history records, each carrying its point and prefix index. Hash descriptions use another $L_{\mathrm{opt}}\,\mathrm{poly}(d, \log\Delta)$ words.

C CORRECTNESS OF THE ISOLATE SUMMARY

This section proves the two deferred lemmas of Section B: the exact merge of isolated samples and the exact merge of isolated histories. Throughout, input points are distinct, and $q_1, \ldots, q_r$ are the global priorities fixed in Section 3.2.

C.1 PROOF OF LEMMA B.1

We show that the sequential without-replacement sample has the heredity property needed by the summary. This argument is deterministic once the global priorities are fixed.

Lemma C.1. *Let X and Y be sequences. For every $i \geq 1$ for which the sample of $X \circ Y$ has not terminated, define*

$$C_i := \big(S_i(X) \cup S_i(Y)\big) \setminus S_{i-1}(X \circ Y).$$

Then

$$s_i(X \circ Y) = \arg\min_{x \in C_i} q_i(x). \tag{24}$$

Consequently, Algorithm 1 returns $S_r(X \circ Y)$ while storing $O(r)$ points and taking $O(r^2)$ time.

Proof. Let $u := s_i(X \circ Y)$ and suppose $u \in X$. We claim that $u \in S_i(X)$. Otherwise, let $x_j := s_j(X)$ for every child round j. Then $q_i(x_i) < q_i(u)$, so x_i must have been selected by the union in some round $t_1 < i$. At child round t_1, the point x_{t_1} precedes x_i under q_{t_1}; hence x_{t_1} must have been selected by the union in an even earlier round $t_2 < t_1$. Iterating produces an impossible strictly decreasing sequence of positive round indices. Thus $u \in S_i(X)$; the case $u \in Y$ is symmetric.

Therefore $u \in C_i$. Every point of C_i is still available to the union in round i, so its q_i-minimum is exactly u, proving Equation (24). Applying the identity inductively gives Algorithm 1. Since $|C_i| \leq 2i$, the algorithm stores $O(r)$ points and takes $O(r^2)$ time. □

Corollary C.2. *For sequences X, Y and every $j \leq r$,*

$$S_j(X \circ Y) \subseteq S_j(X) \cup S_j(Y).$$

In particular, if a new occurrence p is appended to a sequence Z, then

$$S_j(Z \circ (p)) \subseteq S_j(Z) \cup \{p\}.$$

Proof. By Lemma C.1, every union occurrence selected in one of the first j rounds belongs to the first j samples of one of the two children. This proves the first inclusion. The second follows by taking $Y := (p)$. □

Proof of Lemma B.1. Exactness is Lemma C.1: applying Equation (24) inductively over the rounds shows that Algorithm 1 returns $S_r(X \circ Y)$. Taking $j = r$ in Corollary C.2 gives $S_r(X \circ Y) \subseteq S_r(X) \cup S_r(Y)$. The algorithm stores one point per round, hence $O(r)$ points, and round i minimizes over a candidate set of size at most $2i$, so the total running time is $O(r^2)$. □

The prefix consequence in Corollary C.2 is used below to bound the isolated history and to justify when a protected occurrence can be released.

C.2 Proof of Lemma B.2

For a sequence Z, we first show that the states $S_r(Z_{\le t})$ over all prefixes admit a compact change history, and then prove the correctness and size bound for the outputs of Algorithm 2.

The only probabilistic fact needed here is the standard record-minimum bound: among at most N items arriving independently of i.i.d. continuous priorities, the prefix minimum changes $O(\log(N/\eta))$ times with probability at least $1-\eta$.

Lemma C.3. *Let Z have q occurrences. With probability at least $1-\delta_I$, the total number of changes among the sample positions $s_1(Z_{\le t}), \dots, s_r(Z_{\le t})$ over all prefixes is*

$$O\left(r \log^2 \frac{qr}{\delta_I}\right). \tag{25}$$

Consequently, both the change history $\mathcal{I}_Z$ and the protected set

$$U_Z := \bigcup_t S_r(Z_{\le t})$$

have $\widetilde{O}(r)$ entries. Moreover, every occurrence in U_Z first enters an isolated state at its own index in Z.

Proof. Fix $j \in [r]$ and set $R_{j,t} := Z_{\le t} \setminus S_{j-1}(Z_{\le t})$. Sample heredity gives $R_{j,t} \subseteq R_{j,t+1}$: a new occurrence either enters this residual sequence directly or displaces an old sample into it. This sequence depends only on $q_1, \dots, q_{j-1}$ and is therefore independent of q_j. Hence $s_j(Z_{\le t})$ is the q_j-minimum of an insertion-only sequence independent of its priorities.

Apply the record-minimum bound first to the h_j-priorities of newly appearing buckets and then to the $\widetilde{h}_j$-priorities of occurrences within each bucket. With the failure probability divided among all positions and buckets, only $O(\log(qr/\delta_I))$ buckets become the minimum bucket, and the minimum occurrence within each such bucket changes $O(\log(qr/\delta_I))$ times. Thus position j has $O(\log^2(qr/\delta_I))$ segments. A union bound over $j \in [r]$ proves Equation (25); the same bound applies to U_Z, since every protected occurrence appears in an initial state or at a recorded change.

Finally, Corollary C.2 says that, after an occurrence has arrived, it cannot newly enter a later isolated state. Therefore any occurrence that ever belongs to U_Z enters at its own index in Z, proving the last claim. □

Let X and Y be sequences, and set $A := S_r(X)$. For every prefix ending in Y, exact sample merging gives

$$S_r((X \circ Y)_{\le |X|+j}) = \mathsf{IsolateSampleMerge}\big(A, S_r(Y_{\le j}), r\big), \qquad 1 \le j \le |Y|. \tag{26}$$

Lemma C.4. *Let $(\mathcal{I}, U, R_I)$ be returned by Algorithm 2. The history $\mathcal{I}$ records $S_r((X \circ Y)_{\le t})$ exactly for every prefix index t. Moreover,*

$$U = U_{X \circ Y} = \bigcup_t S_r((X \circ Y)_{\le t}), \tag{27}$$

and the following identity holds, with disjoint sets on each side:

$$U_X \cup U_Y = U_{X \circ Y} \cup R_I. \tag{28}$$

Every occurrence of R_I is emitted exactly once at its index in $X \circ Y$. With probability at least $1-\delta_I$, all three outputs contain $\widetilde{O}(r)$ records and points in total.

Proof. Prefixes inside X are copied from $\mathcal{I}_X$. For a prefix ending in Y, Equation (26) is exact by Lemma C.1; its right-hand side changes only when the state recorded by $\mathcal{I}_Y$ changes. Thus the algorithm records every parent state exactly.

Every point of U_X remains protected because every prefix of X is also a parent prefix. Now consider $p \in U_Y$ when it first enters a child state. By Lemma C.3, this is the prefix ending at p. If p belongs to the exact parent state at that prefix, the algorithm protects it. Otherwise, sample heredity implies that this already-arrived occurrence can never enter a later parent state, so the algorithm releases it once, at the index of p in $X \circ Y$. Conversely, sample-merge heredity places every parent sample in the union of the two child samples. This proves Equations (27) and (28) and disjointness.

Finally, apply Lemma C.3 to X, Y, and $X \circ Y$, with failure probability $\delta_I/3$ for each. The parent history and protected set have $\widetilde{O}(r)$ size, and $R_I \subseteq U_Y$ gives the same bound for the release stream. □

Proof of Lemma B.2. We verify the four claims against Lemma C.4. The exact recorded state at every prefix is its first claim. The protected-set identity is Equation (27). The release-stream description is Equation (28) together with its single-emission property: every occurrence of R_I is emitted exactly once, at its index in $X \circ Y$. Finally, with probability at least $1 - \delta_I$, the history, the protected set, and the release stream contain $\widetilde{O}(r)$ records and points in total by the last claim of Lemma C.4. □

D Fixed-Guess Correctness and Optimum Guesses

By Definition 3.5, a guess is successful for a prefix X if $|\varphi(X \setminus U_X)| \le N$. By Lemma B.3, this holds exactly when the prefix precedes the guess's failure index. Success is prefix-specific: a later failure does not invalidate an earlier query.

D.1 The known isolation guarantee

The success of a correct-scale guess rests on the isolation analysis of Jiang & Lou (2025). Throughout, input points are distinct, and $q_1, \dots, q_r$ are the global priorities fixed in Section 3.2.

We first state the geometric lemma used by the extraction argument.

Lemma D.1 (Jiang & Lou, 2025, Lemma 5.3)**.** *Let X be an unweighted point set with $|X| \ge m$, let $\lambda > 0$, and let φ be a (Γ, Λ)-consistent hash at scale λ. There is a set $F \subseteq X$ such that*

$$|F| \le m + \left(\frac{2\Gamma}{\lambda}\right)^z \mathrm{OPT}_z^{(m)}(X) \tag{29}$$

and

$$|\varphi(X \setminus F)| \le k\Lambda. \tag{30}$$

Our scale convention differs from theirs by an inessential constant.

For a positive guess satisfying

$$\frac{\mathrm{OPT}_z^{(m)}(X)}{z+1} \le \widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(X), \tag{31}$$

substitution of Equation (5) into Equation (29) gives

$$\begin{aligned} |F| &\le m + (z+1)\left(\frac{2\Gamma}{c_z}\right)^z \frac{m}{\varepsilon} \\ &\le T := 2^{O(z\log(z+1))}\frac{\Gamma^z m}{\varepsilon}. \end{aligned} \tag{32}$$

Here we used $c_z = (z+1)^{-c}$ and absorbed the additive m into the last expression. The zero-cost case is handled separately by the zero guess in Lemma 3.7.

D.2 Priority realization

The lexicographic priorities in Equation (13) implement exactly the sampling process used in the extraction analysis.

Lemma D.2. *Condition on the first $i-1$ sampling rounds and let $R \coloneqq X \setminus S_{i-1}(X)$ be nonempty. The occurrence $s_i(X)$ is distributed as follows: first choose a bucket uniformly from the nonempty buckets $\varphi(R)$, and then choose an occurrence uniformly from that bucket. Consequently, $s_1(X), s_2(X), \ldots$ are two-level samples drawn sequentially without replacement.*

Proof. Conditioned on R, the fresh values $h_i(u)$ are i.i.d. and continuous, so their minimum selects a uniform bucket of $\varphi(R)$. Within that bucket, the fresh values $\widetilde{h}_i$ similarly select a uniform residual occurrence. Since the minimization in round i excludes $S_{i-1}(X)$, the resulting sequence is sampled without replacement. □

Lemma D.3. *Fix $F \subseteq X$ with $|F| \le T$, where $T \ge 1$, and let*

$$r \coloneqq cT \left\lceil \log \frac{T}{\delta_I} \right\rceil$$

for a sufficiently large constant c. Then, with probability at least $1 - \delta_I$,

$$|\varphi(X \setminus S_r(X))| \le 2|\varphi(X \setminus F)|. \tag{33}$$

Proof. By Lemma D.2, $S_r(X)$ has exactly the two-level without-replacement distribution used in Jiang & Lou (2025, Lemma C.1). Applying that lemma with failure probability δ_I gives Equation (33). □

Combining Lemmas D.1 and D.3 with Equation (32) and setting $N \coloneqq 2k\Lambda$ proves the remaining probabilistic claim of Lemma 2.3: for a correct-scale guess,

$$|\varphi(X \setminus S_r(X))| \le 2|\varphi(X \setminus F)| \le 2k\Lambda = N$$

with probability at least $1 - \delta_I$. Since the protected set used by the summary contains $S_r(X)$, removing the protected set can only decrease the number of residual buckets.

D.3 Correctness for one guess

Lemma D.4. *Fix a positive guess and a prefix X satisfying*

$$\frac{\mathrm{OPT}_z^{(m)}(X)}{z+1} \le \widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(X).$$

With probability at least $1 - \delta_I$, the residual set $X \setminus U_X$ intersects at most $N = 2k\Lambda$ hash buckets. Consequently, the guess is successful at this prefix.

Proof. By Lemma D.1, there is an exceptional set F of the size allowed by Lemma D.3 whose removal leaves at most $k\Lambda$ buckets. Our choice of r therefore gives, with probability at least $1 - \delta_I$,

$$|\varphi(X \setminus S_r(X))| \le 2|\varphi(X \setminus F)| \le 2k\Lambda = N.$$

Since $S_r(X) \subseteq U_X$, the residual $X \setminus U_X$ intersects no more buckets. By Lemma B.3, the guess has not failed at this prefix. □

Proof of Lemma 3.6. Fix a successful prefix $X \coloneqq Z_{\le t}$ and restrict all sets below to this prefix. By Lemma B.5, $X = U \cup H \cup T$, and $V \coloneqq \mathsf{VanillaQuery}(\mathcal{V}_Z, t)$ is a vanilla α-coreset for T.

Partition $R \coloneqq X \setminus U$ into buckets of size below M and buckets of size at least M. The former buckets are stored exactly in H. Define

$$R_u \coloneqq R \cap \varphi^{-1}(u),$$

$$\mathsf{Outer}(X) = U \cup \bigcup_{\substack{u \in \varphi(R) \\ |R_u| < M}} R_u,$$

$$\mathsf{Inner}(X) = \bigcup_{\substack{u \in \varphi(R) \\ |R_u| \ge M}} R_u,$$

$$H_{\mathrm{in}} \coloneqq H \cap \mathsf{Inner}(X).$$

Then $\mathsf{Inner}(X) = H_{\mathrm{in}} \cup T$, so exactness of H_{in} and the guarantee for V imply that $H_{\mathrm{in}} \cup V$ is a vanilla α-coreset for $\mathsf{Inner}(X)$.

No preservation of the tail mass is needed. If $\mathsf{Inner}(X)$ is nonempty, the exact head of each dense bucket already has mass $M > m$, so every outlier budget $h \le m$ is feasible in $H_{\mathrm{in}} \cup V$ independently of $w(V)$. If $\mathsf{Inner}(X)$ is empty, the prefix is represented exactly by $\mathsf{Outer}(X)$.

Every bucket in $\mathsf{Inner}(X)$ has diameter at most λ and mass at least $M \ge (1+\varepsilon^{-1})m$. Since $\widehat{\mathrm{OPT}} \le \mathrm{OPT}_z^{(m)}(X)$, Lemma A.5 applied to the exact part $\mathsf{Outer}(X)$ and dense part $\mathsf{Inner}(X)$ shows that $\mathsf{Outer}(X) \cup H_{\mathrm{in}} \cup V = U \cup H \cup V$ is an $O(\alpha+\varepsilon)$-coreset for X. This is exactly the output of $\mathsf{PrefixCoreset}$, with $\mathsf{Outer}(X) = \mathsf{Isolate}(X) \cup \mathsf{Sparse}(X)$ and $V_t = H_{\mathrm{in}} \cup V$. □

The preceding proof only requires that the successful guess does not exceed the optimum. A constant-factor lower bound on the guess is used solely to guarantee success of some guess.

D.4 Removing the optimum guess

Proof of Lemma 3.7. Consider a prefix X of length at most L. For a positive optimum, some center in an optimal solution serves two distinct grid locations. One of them is at distance at least $1/2$ from the center, while assigning all points to any input location gives cost at most $|X|(\sqrt{d}\,\Delta)^z \le L(\sqrt{d}\,\Delta)^z$. Hence

$$2^{-z} \le \mathrm{OPT}_z^{(m)}(X) \le L(\sqrt{d}\,\Delta)^z.$$

We therefore run the positive guesses $g_j := 2^{-z}(z+1)^j$ over this range, together with a zero guess. The number of guesses is $L_{\mathrm{opt}} = 1 + O(\log_{z+1}(2^z L(\sqrt{d}\,\Delta)^z))$. For every positive optimum, the largest g_j not exceeding it lies in $[\mathrm{OPT}_z^{(m)}(X)/(z+1), \mathrm{OPT}_z^{(m)}(X)]$ and is successful by Lemma D.4.

For the zero guess, set $\lambda := 0$ and use the exact-location hash. If the optimum is zero, removing at most m occurrences leaves at most k locations; by Lemma D.3, the residual occupies at most $2k \le N$ buckets with high probability, so this guess succeeds. At any prefix where it succeeds, its output is also correct: all buckets have diameter zero, so the additive term in Lemma A.5 vanishes and the proof of Lemma 3.6 applies with $\widehat{\mathrm{OPT}} = 0$.

At a query prefix, return the output of the smallest guess that is successful at that prefix. For a positive optimum, the selected positive guess is no larger than the successful correct-scale guess and is therefore correct by Lemma 3.6; if zero is selected, the preceding zero-diameter argument applies. For a zero optimum, the zero guess succeeds and is selected.

Allocate the failure probability over all guesses, merge nodes, prefixes, and vanilla instances. A union bound gives all claims simultaneously with probability at least $1-\delta$; independence is unnecessary. The query returns one fixed-guess output, so the number of guesses does not multiply the coreset-size bound. □