Differentially Private Clustering in Data Streams

Alessandro Epasto
Google

Tamalika Mukherjee
Max Planck Institute for Security and Privacy

Peilin Zhong
Google

Abstract

Clustering tasks such as k-means and k-median are central in unsupervised learning, and streaming algorithms for these tasks are widely used to handle large or evolving datasets. When applied in sensitive domains, however, such algorithms must also provide rigorous privacy guarantees.

In this work, we provide the first differentially private (DP) algorithms for k-means and k-median clustering of d-dimensional Euclidean data points over a stream of length at most T, using space that is sublinear in T, in the continual release setting where the algorithm is required to output a clustering at every timestep.

We achieve (1) an $O(1)$-multiplicative approximation with $O(k^{1.5} poly(d, \log T))$ space and $poly(k, d, \log T)$ additive error, or (2) a $(1+\gamma)$-multiplicative approximation with $O_\gamma(poly(k, 2^{O_\gamma(d)}, \log T))$ space for any $\gamma > 0$, with additive error $poly(k, 2^{O_\gamma(d)}, \log T)$. Our main technical contribution is a DP clustering framework for data streams that only requires an offline DP coreset or clustering algorithm as a blackbox.

1 INTRODUCTION

In real-world applications, a major challenge in dealing with large-scale data is that the entire datasets are too large to be stored in the computing system. The need to address this challenge and the success of large-scale systems (such as Spark Streaming Zaharia et al. [2012]) that process data in streams have driven the study of the *streaming model*, introduced by the seminal work Alon et al. [1999]. In this model, there is a stream of data points — a data point arrives at each timestamp, and a streaming algorithm can only access these data points by a single pass. The goal is to output an (approximate) solution to a problem with respect to the set of data points that are arrived while using as small space as possible. If an algorithm is required to output at every timestamp when a new data point arrives, then it is called the *continual release* setting. Otherwise, the algorithm only needs to output at the end of the stream, which is called the *one-shot* setting. The streaming model has attracted a lot of attention from different areas in the past decades. In particular, streaming clustering algorithms have been extensively studied by the clustering literature.

Clustering is an essential primitive in unsupervised machine learning, and its geometric formulations, such as k-means and k-median, have been studied extensively, e.g., Arya et al. [2001], Charikar et al. [2002], Har-Peled and Mazumdar [2004], Chen [2006, 2008], Awasthi et al. [2010], Ostrovsky et al. [2012], Li and Svensson [2016], Ahmadian et al. [2020]. In the streaming k-clustering problem, there is a stream of points in $\mathbb{R}^d$, and the goal is to output a set of k centers at each timestamp t, and minimize the k-clustering cost with respect to data points arrived before t. There is a long list of work (e.g., a subset includes Har-Peled and Mazumdar [2004], Har-Peled and Kushal [2005], Chen [2009], Feldman and Langberg [2011], Braverman et al. [2019], Cohen-Addad et al. [2022c, 2023]) studying k-means and k-median problem in the streaming setting. The state-of-the-art result is achieved by Cohen-Addad et al. [2023] which uses $\tilde{O}\left(\frac{kd}{\gamma^2}\right) \cdot \min\left(\frac{1}{\gamma^z}, k\right) \cdot \mathrm{poly}(\log\log T)$ space to obtain a $(1+\gamma)$-approximation with probability 0.9 at the end of the stream. However, none of these algorithms are private, which means they are not applicable when the dataset involves personal information and privacy is considered to be a major concern in real-world applications.

Differential privacy (DP) Dwork et al. [2016] has become the de facto standard for preserving data privacy due to its compelling privacy guarantees and mathematically rigorous definition. DP k-clustering algorithms in

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

the offline setting have been studied for years Nissim et al. [2007], Feldman et al. [2009, 2017], Gupta et al. [2010], Balcan et al. [2017], Huang and Liu [2018], Nissim and Stemmer [2018], Stemmer and Kaplan [2018], Ghazi et al. [2020], Cohen et al. [2021], where the main focus is to improve the approximation ratio and achieve fast sequential running time. DP k-clustering problem has also been studied in other computational models which are more relevant to large-scale computations such as sublinear-time Blocki et al. [2021] and massively parallel computing (MPC) Cohen-Addad et al. [2022a,b], and distributed computing setting Xia et al. [2020], Chang et al. [2021]. However, the landscape of the DP k-clustering problem in the streaming model is still mysterious. In fact, to the best of our knowledge, there is no previously known DP streaming algorithm achieving $O(1)$-multiplicative error using sublinear space even in the one-shot setting.

In this work, we present the *first* DP streaming algorithms for Euclidean k-means and k-median clustering using $\mathrm{poly}(k, d, \log(T))$ space to achieve an $O(1)$-multiplicative error and a $\mathrm{poly}(k, d, \log(T))$-additive error. Our algorithms are DP under the *continual release* setting. Note that any DP algorithm under the continual release setting is always DP under the one-shot setting.

1.1 Differential Privacy Model and Clustering Problem

We first formally define differential privacy and the streaming model under continual release setting and then define the clustering problem. The input is a stream of points $x_1, x_2, \cdots, x_T \in \mathbb{R}^d$, where each $x_i \in \mathbb{R}^d$ satisfies $\|x_i\|_2 \leq \Lambda$, i.e., we assume all input points are within a ball of radius Λ. We work in the *insertion-only* setting, where points arrive sequentially but never leave the stream. That is, at time t the dataset consists of all points $x_1, \ldots, x_t$ seen so far, and future outputs must remain consistent with this growing prefix.

In this paper, we study streaming algorithms $\mathcal{A}$ under the continual release setting, i.e., the entire output of $\mathcal{A}$ is $(s_1, s_2, \cdots, s_T)$ where s_t is the output of $\mathcal{A}$ at timestamp t with respect to the data arrived no later than t. We work under *event-level privacy* where the streams $\mathcal{S} = (x_1, \ldots, x_T)$ and $\mathcal{S}' = (x'_1, \ldots, x'_T)$ are neighboring if there exists at most one timestamp $t^* \in [T]$ for which $x_{t^*} \neq x'_{t^*}$ and $x_t = x'_t$ for all $t \neq t^*$.

If we do not specify the timestamp, then the output of $\mathcal{A}(\mathcal{S})$ indicates the entire output of $\mathcal{A}$ over the stream $\mathcal{S}$ at every timestamp.

Definition 1 (Differential privacy Dwork et al. [2016])**.** *A randomized algorithm $\mathcal{A}$ is (ε, δ)-DP if for every pair of neighboring streams $\mathcal{S} \sim \mathcal{S}'$, and for all sets $\mathcal{O}$ of possible outputs, we have that $\Pr[\mathcal{A}(\mathcal{S}) \in \mathcal{O}] \leq e^{\varepsilon} \Pr[\mathcal{A}(\mathcal{S}') \in \mathcal{O}] + \delta$. When $\delta = 0$ we simply say that the algorithm is ε-DP.*

For points $x, y \in \mathbb{R}^d$, we let $d(x, y) = \|x - y\|_2$ be the Euclidean distance between x and y. Given a set $\mathcal{C}$, we define $d(x, C) := \min_{c \in C} d(x, c)$.

For a set of centers C, we define the cost of clustering for the set $\mathcal{S}$ w.r.t. C as

$$\mathsf{cost}(C, \mathcal{S}) = \sum_{x \in \mathcal{S}} d^z(x, C)$$

where $z = 1$ for k-median, and $z = 2$ for k-means.

Our goal in DP clustering is to produce a set of k centers $C_{\mathcal{S}}$ for input stream $\mathcal{S}$ such that (1) $C_{\mathcal{S}}$ is (ε, δ)-DP wrt $\mathcal{S}$, and (2) $\mathsf{cost}(C_{\mathcal{S}}, \mathcal{S}) \leq \alpha \cdot \mathsf{cost}(C_{\mathcal{S}}^{opt}, \mathcal{S}) + \beta$.

1.2 Our Results

Before presenting our main results, we first define some useful terminology. The notation $O_x(\cdot)$ ignores factors involving x. We use the term *semicoreset* to describe a relaxed version of a coreset that allows an additive error proportional to the optimal cost. This notion is not intended as a canonical definition, but rather as a convenient abstraction that simplifies the presentation of our results. In particular, it enables a unified statement of Theorem 1 that can work using both static DP coreset algorithms and static DP clustering algorithms as a blackbox.

Definition 2 ((κ, η_1, η_2)-semicoreset)**.** *Given a point set P in $\mathbb{R}^d$, a (multi)set Q is a (κ, η_1, η_2)-semicoreset of P for k-clustering (k-means or k-median) if*

$$\begin{aligned} &\frac{1}{\kappa} \cdot \mathsf{cost}(C, P) - \eta_1 \cdot \mathsf{cost}(C_P^{opt}, P) - \eta_2 \\ &\leq \mathsf{cost}(C, Q) \\ &\leq \kappa \cdot \mathsf{cost}(C, P) + \eta_2 \end{aligned}$$

for any set of k-centers $C \subseteq \mathbb{R}^d$.

In particular, a coreset that has additive error that is not proportional to the optimal cost is simply a $(\kappa, 0, \eta_2)$-semicoreset. We also abuse notation and refer to a coreset with *no* additive error, i.e., $(\kappa, 0, 0)$-semicoreset as a *κ-coreset*.

Given an offline DP k-clustering algorithm one can obtain an offline DP semicoreset by using a transformation (stated in Theorem 9) by Cohen-Addad et al. [2022b] — we use this transformation as a blackbox in the sequel. We state our results for k-means, but note that our results easily generalize to k-median. As is standard in DP

clustering literature (e.g.,Stemmer and Kaplan [2018]), we assume Λ is an upper bound on the diameter of the space of input points. For ease of presentation, we assume $\Lambda = 1$ in this part.

Our main result is a general framework for DP k-clustering in the streaming setting which utilizes an offline DP semicoreset algorithm as a black-box (see Theorem 1). Using existing results from the DP clustering literature, the cost of the resulting clustering output by our framework achieves (1) an $O(1)$-multiplicative error with space complexity having a $k^{1.5}$ dependency — using the DP clustering algorithm from Stemmer and Kaplan [2018] (see Theorem 2), or (2) a $(1+\gamma)$-multiplicative error with space complexity having a $\mathrm{poly}(k)$ dependency — using the DP coreset algorithm from Ghazi et al. [2020] (see Theorem 3).

We emphasize that our work establishes the theoretical feasibility of sublinear-space DP clustering in streams—a fundamental question that was previously open. As offline DP clustering algorithms achieve better additive error bounds, our black-box approach immediately inherits these improvements, potentially leading to more practical streaming variants.

We assume we are given a non-DP algorithm in the offline setting that can compute a $(1+\gamma)$-approximation to k-means (e.g. Saulpic et al. [2019]) in the following statements.

Theorem 1 (Main)**.** *Given dimension d, clustering parameter k, arbitrary parameter C_M, a non-DP $(1+\gamma)$-coreset algorithm, an (ε,δ)-DP (κ,η_1,η_2)-semicoreset algorithm $\mathcal{A}$ that outputs a semicoreset of size $SZ_{\mathcal{A}}(\cdot)$ and using space $S_{\mathcal{A}}(\cdot)$. Then, there exists a streaming algorithm $\mathcal{A}'$ for k-means that outputs a set of centers $\mathcal{C}_{\hat{\mathcal{Y}}}$ at every timestep $t \in [T]$ such that*

1. *(Privacy) $\mathcal{A}'$ is $(3\varepsilon,\delta)$-DP under the continual release setting.*
2. *(Accuracy) With probability $1-\frac{1}{T^2}-\frac{1}{k^{O(\mathrm{poly}(k)}}$,*

$$\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}},\mathcal{S}) \le \kappa\cdot\left((1+\gamma)C_M+\eta_1+\frac{(1+\gamma)^4}{(1-\gamma)^3}(\kappa+C_M)\right)\cdot \mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S})+V(d,k,\varepsilon,\delta,T,\gamma)$$

   where $\mathcal{S}$ denotes the set of all points given by the stream before timestamp t, $\mathcal{C}_{\mathcal{S}}^{opt}$ is the optimal solution, and $V(d,k,\varepsilon,\delta,T,\gamma) = \tilde{O}_\gamma(\kappa(\frac{C_M k^2}{d}+kM)\frac{1}{\varepsilon}\,\mathrm{poly}\log(T))$.
3. *(Space) $\mathcal{A}'$ consumes $\tilde{O}_\gamma(S_{\mathcal{A}}(M)+SZ_{\mathcal{A}}(M)+M+k\,\mathrm{poly}\log(\frac{T}{M}))$ space.*

where $M = O(\frac{d^3\eta_2}{C_M})$.

Remark. The space complexity achieved by our framework depends on the additive error (i.e., the term η_2) of the offline DP (κ,η_1,η_2)-semicoreset algorithm used as a black-box. In particular, if there exists a DP semicoreset algorithm which achieves a better additive error in the offline setting, then our framework immediately gives a DP clustering algorithm with better space bounds in the continual release setting.

Remark. The parameter C_M in Theorem 1 is assigned an appropriate value depending on the error of the offline DP semicoreset algorithm used as a black-box in order to obtain the desired multiplicative approximation of the resulting algorithm $\mathcal{A}'$. For e.g., we can set $C_M = O(1)$ to obtain an $O(1)$-multiplicative approximation in Theorem 2, and we can set $C_M = \gamma/2$ to obtain a $(1+O(\gamma))$-multiplicative approximation in Theorem 3.

We first present Theorem 2 which is the result of applying our framework to the DP k-means algorithm given by Stemmer and Kaplan [2018] in the offline setting that achieves an $O(1)$ multiplicative error. We note that Stemmer and Kaplan [2018] gave a DP coreset algorithm which has an additive error with $\mathrm{poly}(k)$ dependency whereas their DP k-means algorithm achieves an additive error of $k^{1.5}$. This is why we choose to use their DP k-means algorithm and compute a DP semicoreset albeit with a slight loss in privacy to save on space (in terms of k) instead of directly using their DP coreset algorithm.

Theorem 2. *Given dimension d, clustering parameter k, privacy parameters ε,δ, approximation parameter $\gamma > 0$. There exists a streaming algorithm $\mathcal{A}'$ for k-means that outputs a set of k centers $\mathcal{C}_{\hat{\mathcal{Y}}}$ at every timestep $t \in [T]$ such that*

1. *(Privacy) $\mathcal{A}'$ is $(5\varepsilon,\delta)$-DP under the continual release setting.*
2. *(Accuracy) With probability $1-\frac{1}{T^2}-\frac{1}{k^{O(\mathrm{poly}(k))}}$,*

$$\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}},\mathcal{S}) \le O_\gamma(1)\cdot\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S})+V(d,k,\varepsilon,\delta,T)$$

   where $\mathcal{S}$ denotes the set of all points given by the stream before timestamp t, $\mathcal{C}_{\mathcal{S}}^{opt}$ is the optimal solution, and $V(d,k,\varepsilon,\delta,T) = \tilde{O}_\gamma((k^{2.5}d^{3.51})\cdot\mathrm{poly}(\log(T),\log(\frac{1}{\delta}),\frac{1}{\varepsilon}))$.
3. *(Space) $\mathcal{A}'$ consumes $\tilde{O}((k^{1.5}\cdot d^{4.51})\cdot\mathrm{poly}(\log(T),\log(\frac{1}{\delta}),\frac{1}{\varepsilon}))$ space.*

Next, we present the result of applying our framework to the DP coreset for k-means of Ghazi et al. [2020] which achieves a multiplicative error of $1+\gamma$ and an additive error of $O_\gamma(\frac{k^2 2^{O_\gamma(d)}}{\varepsilon}\,\mathrm{poly}\log(T))$ in Theorem 3.

Theorem 3. *Given dimension d, clustering parameter k, privacy parameter ε, approximation parameter $\gamma > 0$. There exists a streaming algorithm $\mathcal{A}'$ for k-means that outputs a set of k centers $\mathcal{C}_{\hat{\mathcal{Y}}}$ at every timestep $t \in [T]$ such that*

1. *(Privacy) $\mathcal{A}'$ is 3ε-DP under the continual release setting.*
2. *(Accuracy) With probability $1 - \frac{1}{T^2} - \frac{1}{k^{O(\mathrm{poly}(k))}}$,*

   $$\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \mathcal{S}) \leq (1+\gamma)\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt}, \mathcal{S}) + V(d, k, \varepsilon, T)$$

   where $\mathcal{S}$ denotes the set of all points given by the stream before timestamp t, $\mathcal{C}_{\mathcal{S}}^{opt}$ is the optimal solution, $V(d,k,\varepsilon,T) = \tilde{O}_\gamma(\frac{k^3 d^3 2^{O_\gamma(d)}}{\varepsilon} \cdot \mathrm{poly}\log(T))$.
3. *(Space) $\mathcal{A}'$ consumes $\tilde{O}\left(\mathrm{poly}\left(\frac{2^{O_\gamma(d)}}{\varepsilon}\log(T), k, d\right)\right)$ space.*

Remark. We note that the exponential dependency in d in Theorem 3 is a *direct consequence* of the additive error of the DP coreset having the same dependency in Ghazi et al. [2020].

1.3 Related Work

In the offline setting, private clustering was first studied by Gupta et al. [2010], and Feldman et al. [2009]. In particular, Gupta et al. [2010] used the exponential mechanism to produce a pure DP polynomial-time algorithm that achieves $(O(1), \tilde{O}(k^2\Lambda))$-approximation in discrete spaces. However, their algorithm is highly inefficient in Euclidean space (see Stemmer and Kaplan [2018] for a detailed exposition). Balcan et al. [2017], Stemmer and Kaplan [2018] focused on designing an efficient polynomial time algorithm for clustering that achieves a constant (multiplicative) factor approximation in high-dimensional Euclidean space by adopting the techniques of Gupta et al. [2010] while maintaining efficiency. Feldman et al. [2009] introduced the notion of private coresets and a recent line of work has adopted their techniques to give clustering algorithms with better approximation guarantees and efficiency Feldman et al. [2017], Nissim and Stemmer [2018]. Stemmer and Kaplan [2018] gave a pure DP k-means algorithm with $O(1)$-multiplicative error, Chaturvedi et al. [2021] achieved the same multiplicative error but improved the additive error to $\tilde{O}(k\sqrt{d}\Lambda^2/\varepsilon)$. Nguyen [2020] and Ghazi et al. [2020] gave pure DP clustering algorithms that achieve optimal multiplicative error of $(1+\gamma)w^*$, where w^* is the best approximation ratio for non-private k-means clustering.

Concurrent Works. la Tour et al. [2023] concurrently released results on differentially private k-means clustering in the continual release setting. Their work is mostly incomparable with ours — their main focus is to optimize the approximation ratio under the more general setting of continual observation with the insertion *and* deletion of points, but their algorithm does not optimize for space and requires storing all the input points in the stream. More recently la Tour et al. [2024] showed that a twenty year old greedy algorithm for clustering can be made DP and gives DP clustering algorithms for both the static and continual release setting that achieves the same multiplicative error as Ghazi et al. [2020] and additive error that has a $\log^{1.5}(T)$ dependency. This algorithm also does not optimize for space and inherently stores all the points in the stream.

In contrast, we work in the insertion-only streaming setting and our main focus is to optimize the approximation ratio using low space, i.e., $\mathrm{poly}\log(T)$, in the continual release setting. Note that we achieve the same multiplicative error as la Tour et al. [2023] in Theorem 3.

1.4 Our Techniques

Our techniques apply to both k-means and k-median clustering, but for simplicity, we explain our techniques for the problem of k-means. We first outline the challenges to designing a DP k-means clustering algorithm in the continual release setting and then discuss the main ideas behind our DP clustering framework.

Naive Merge and Reduce approaches fail. A standard streaming technique for clustering is the Merge-and-Reduce framework Har-Peled and Mazumdar [2004], Agarwal et al. [2004], Feldman et al. [2020]. The idea is to maintain coresets $C_1, \ldots, C_L$ for $L = \log T$ levels (using a static coreset algorithm as a black-box) such that their union forms a coreset of all points seen so far. Cohen-Addad et al. [2022c] showed that any point set has a $(1+\gamma)$-coreset with size $k \cdot \mathrm{poly}(1/\gamma)$, thus each level has capacity threshold $M = k \cdot \mathrm{poly}(\log T)$. When a new point x arrives, it is inserted into C_1. For $i \in [L]$, if some C_i exceeds size M, we compress it into a $(1 + 1/\mathrm{poly}(\log T))$-approximate coreset C', reset $C_i \leftarrow \emptyset$, and insert C' into C_{i+1}. This process repeats until all levels respect the size bound. By induction, each C_i is a $(1+1/\mathrm{poly}(\log T))^i$-coreset of its underlying input points, and hence $C_1 \cup C_2 \cup \cdots \cup C_L$ is a $(1 + 1/\mathrm{poly}(\log(T)))^L = (1 + O(\log(T))/\mathrm{poly}(\log(T)))$-coreset of the entire input stream.

We can adapt the classical merge-and-reduce method to the DP setting as follows. At level 1, whenever the buffer C_1 reaches size M, we run a DP k-means coreset routine to compress it. The resulting summary is inserted

into C_2, and higher levels merge summaries using a non-private coreset algorithm. By the post-processing property, this preserves differential privacy while avoiding repeated privacy loss at higher levels. Note that the size checks that trigger compression at each level can be implemented privately with the Sparse Vector Technique (SVT) Hardt and Rothblum [2010], Dwork et al. [2009] (see Appendix D for details).

The key point is that any DP coreset must introduce some additive error $\eta = \Omega(\Lambda^2)$ Gupta et al. [2010], Feldman et al. [2009]. If we apply this framework to the entire stream, the additive error introduced at the base still accumulates across levels: each merge adds the error from its children. After $\log T$ levels, this leads to a total additive error of $T^{\Omega(1)} \cdot \Lambda^2$, which is too large to yield meaningful guarantees.

Our Approach. Our main technical innovation is to design methods that prevent the additive error from growing exponentially, as in the naive implementation of the DP merge-and-reduce framework described above.

Rather than applying a single instance of the DP merge-and-reduce framework to the entire input stream, we first partition the space $\mathbb{R}^d$ so that nearby input points are grouped together, and apply multiple instances of the DP merge-and-reduce framework to disjoint groups in parallel. To do this, we compute a set of candidate centers forming a *bicriteria approximation*[1] for k-means, and then assign input points to groups based on these centers. Within each group, we run a DP merge-and-reduce framework: the first level uses a DP clustering or DP coreset algorithm, while higher levels are computed using a non-DP coreset algorithm. This produces a DP *semicoreset* (see Definition 2). The semicoreset abstraction makes our framework flexible, since it can incorporate either a DP clustering algorithm or a DP coreset algorithm as the base primitive in the merge-and-reduce process.

For every timestamp $t \in [T]$, on a high-level, our algorithm (Algorithm 1) does the following —

1. Compute a set of centers $\mathcal{F}$ in an online DP fashion that satisfies a bicriteria approximation (Algorithm 3) to k-means.
2. Maintain DP semicoresets of the points assigned to centers in $\mathcal{F}$ in parallel via multiple DP Merge-and-Reduce (Algorithm 4) instances
3. Output the union of these semicoresets called $\hat{\mathcal{Y}}$.

[1]Here, a bicriteria approximation means that the algorithm may output more than k centers while still approximating the k-means cost.

In a post-processing step — Compute a non-DP k-means $(1+\gamma)$-approximation algorithm on the output $\hat{\mathcal{Y}}$.

In the remainder of this section, we briefly discuss the different components of our approach.

Bicriteria Approximation. We first design a sublinear space online ε-DP algorithm that outputs a set of $\tilde{O}(k \log(T))$ candidate centers satisfying a $d^{O(1)}$-multiplicative approximation and $\tilde{O}(\frac{d^2\Lambda^2 k}{\varepsilon} \cdot \mathrm{poly}\log(T))$ additive error. The bicriteria approximation algorithm uses two main ingredients — *quadtrees* and *heavy hitters*. A quadtree creates a nested series of grids that partition $\mathbb{R}^d$ and can be used to embed input points into a Hierarchically Separated Tree (HST) metric, which often simplifies the analysis of k-means cost. We initialize our quadtree with $\log(\Lambda)$ levels, and use this embedding to map every input point to the center of a grid cell at each level.

For a fixed level, our goal is to approximately choose $\tilde{O}(k)$ cells that have the most points and store them as candidate centers in set $\mathcal{F}$ while using low space. To do this, we first hash cells of the fixed quadtree level to $w = O(k)$ buckets, and at every timestep, we track the DP size of each bucket through the standard Binary Mechanism Dwork et al. [2010], Chan et al. [2011], as well as privately compute the heavy hitters of each bucket using the DP heavy hitter algorithm from Epasto et al. [2023] . Note that we need to track the DP size of each bucket using the Binary Mechanism separately to ensure that we can prune false positives among candidate centers. Without this pruning step, cells with very small counts could be mistakenly retained as heavy hitters. To the best of our knowledge this is the first use of DP Heavy hitters for DP clustering *in the continual release setting*.

Finally, the algorithm outputs the cumulative set of candidate centers $\mathcal{F}$ across each of the $\log \Lambda$ levels of the quadtree, yielding a bicriteria approximation. See Theorem 4 for a formal statement.

Grouping points and applying Merge and Reduce. Instead of running a single instance of the DP Merge and Reduce framework over the entire stream (as described in the naive approach), our strategy is to partition the space $\mathbb{R}^d$ into groups such that input points close to a candidate center (of the bicriteria solution $\mathcal{F}$) are in the same group and then run $\log(\Lambda)$ parallel instances of the DP Merge and Reduce framework *per group*. We use a partitioning technique introduced by Chen [2009] and define *rings* R_r (where $1 \le r \le \log \Lambda$) based on their distance from the current centers in $\mathcal{F}$ and map input points to these rings (see Definition 3 for a formal definition).

The intuition is that any DP coreset necessarily incurs additive error that scales with the square of the diameter of the set it is applied to. If we ran a single DP Merge and Reduce on all points assigned to a center, the diameter could be as large as Λ, leading to $\Omega(\Lambda^2)$ additive error. By splitting the group into $\log(\Lambda)$ concentric rings, each ring has diameter only $O(2^r)$, so the additive error of the DP routine on ring R_r is $O(2^{2r})$. This error is now proportional to the cost contributed by points in that ring, which allows us to keep the total error under control.

Now we can run the DP Merge and Reduce framework and obtain DP semicoresets for points in each ring, and by taking the union of these semicoresets over all rings, we obtain a DP semicoreset for the stream of points seen so far. Since the DP clustering algorithms we use as a black-box in our Merge and Reduce framework from Ghazi et al. [2020], Stemmer and Kaplan [2018] achieve constant multiplicative error, this technique will result in constant multiplicative error of the resulting DP semicoreset.

Charging Additive Error to Multiplicative Error. Because we are taking the union of DP semicoresets over all rings, the additive error resulting from this union depends on $\sum_r N_r/M$, where N_r is the number of points in ring r and M controls when the DP Merge and Reduce framework is applied. At first glance, this sum could be as large as T/M, which seems too large to be useful.

The crucial insight is that we use a structural property of the bicriteria solution[2] (Lemma 5): the total sum of squared ring radii over all points is bounded, up to constants, by the k-means cost of the bicriteria solution F (i.e., $\text{cost}(F, S) = \sum_{x \in S} d(x, F)^2$), which in turn is within a constant factor of the optimal k-means cost plus a small additive term. This property ensures that the additive error accumulated by the DP Merge and Reduce instances is controlled by the clustering cost. In turn, this allows us to charge the additive error to the multiplicative error, so that with an appropriate choice of M the overall error is just a small constant-factor blow-up in the approximation plus a polylogarithmic additive term (see Appendix B.2 for details).

Finally, observe that the bicriteria solution $\mathcal{F}$ can change over time, i.e., new centers can be added to $\mathcal{F}$. Thus whenever a new center (or centers) are added to $\mathcal{F}$, we redefine the rings (based on distance to the set $\mathcal{F}$) and initiate new DP Merge and Reduce instances for each ring.

[2]Intuitively, this property holds because the bicriteria solution already "pays" for the squared distances that define the rings: every point is assigned to a nearby candidate center, and the radius of its ring is proportional to this distance.

Comparison of our techniques to Cohen-Addad et al. [2022b]. Cohen-Addad et al. [2022b] gave DP k-means and k-median algorithms in the Massively Parallel Computation (MPC) model which achieved $O(1)$-multiplicative and $\text{poly}(k, d)$ additive error. Although their techniques are similar to ours, in the sense that they reduce the problem of solving DP clustering on the entire dataset to solving multiple instances of the DP clustering problem on partitioned input data, there are several differences in our work and new challenges that we have to deal with as a result.

First, their work does not tackle sublinear space in the continual release setting. Second, our bicriteria approximation algorithm uses a black-box continual release *DP heavy hitters* algorithm from Epasto et al. [2023] to find the candidate centers and thus requires a completely different analysis compared to the greedy approach used in Cohen-Addad et al. [2022b]. More importantly, we have to implement a novel DP Merge and Reduce framework that works with both an offline DP clustering or DP coreset algorithm, which was not required in the MPC setting.

2 DIFFERENTIALLY PRIVATE CLUSTERING FRAMEWORK

In this section, we first present our differentially private clustering framework in the continual release setting which is given by Algorithm 1 and discuss the different components of the algorithm. Notably, our framework allows us to plug in any existing offline DP coreset or clustering algorithm to obtain a corresponding DP clustering algorithm in the continual release setting. The proofs of Theorem 1, Theorem 2 and Theorem 3 are stated in Appendix B.

Before presenting the main algorithm, we formally define a ring centered at a set as we will use this to partition the space $\mathbb{R}^d$ in our algorithm.

Definition 3 (Ring centered at a Set)**.** *Let $r \in \mathbb{R}$. Ring R_r for set $\mathcal{F}$ contains the set of points $\{x_i\}_{i \in [T]}$ such that $2^{r-1} \leq d(x_i, \mathcal{F}) < 2^r$.*

Main Algorithm (Algorithm 1). When a new point x_t arrives, our algorithm does the following

1. Update the bicriteria solution $\mathcal{F}$ (see Algorithm 2)
2. If new centers have been added to $\mathcal{F}$ then create rings according to Definition 3 for the set $\mathcal{F}$ and add x_t to a ring[3]. For each ring $1 \leq r \leq \log(\Lambda)$

[3]In the pseudo code Algorithm 1, the symbol $\perp$ represents an empty update that is effectively ignored. This is needed for

Algorithm 1: Main Algorithm

Data: Stream $\mathcal{S}$ of points $x_1, \ldots, x_T \in \mathbb{R}^d$, Privacy parameters $\varepsilon, \varepsilon_1, \delta_1$

Initialize bicriteria solution $\mathcal{F} = \emptyset$;
Initialize flag for new centers being added $flag_{new} = 0$, and DP semicoreset $\hat{\mathcal{Y}} = \emptyset$;
Call **Initialize**(ε) of **BicriteriaDPCenters**; ▷ See Algorithm 2

for *new point* x_t **do**
  $\mathcal{F}_t \leftarrow$**Update**(x_t) of **BicriteriaDPCenters**;
  if $\mathcal{F}_t \neq \emptyset$ *and* $|\mathcal{F} \cap \mathcal{F}_t| < |\mathcal{F}_t|$ **then**
    ▷ New centers added to $\mathcal{F}$ — need to redefine rings
    $flag_{new} = 1$;
  $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{F}_t$;
  for $1 \leq r \leq \log \Lambda$, *run the following in parallel* **do**
    Let R_r represent the ring centered at $\mathcal{F}$ (see Definition 3);
    if $flag_{new} = 1$ **then**
      Create new instance **DP-Merge-Reduce**$_{R_r}$ by calling **Initialize**$(\varepsilon, \varepsilon_1, \delta_1)$ of **DP-Merge-Reduce**; ▷ See Algorithm 4
    if $x_t \in R_r$ **then**
      $\hat{\mathcal{Y}}_r \leftarrow$ **Update**(x_t) of **DP-Merge-Reduce**$_{R_r}$;
    else
      $\hat{\mathcal{Y}}_r \leftarrow$ **Update**$(\perp)$ of **DP-Merge-Reduce**$_{R_r}$;
  $flag_{new} = 0$;
  Set $\hat{\mathcal{Y}} \leftarrow \hat{\mathcal{Y}} \cup (\cup_r \hat{\mathcal{Y}}_r)$;
  Run non-DP $(1+\gamma)$-coreset algorithm on $\hat{\mathcal{Y}}$ and store this semicoreset as the new $\hat{\mathcal{Y}}$;
  Output the semicoreset $\hat{\mathcal{Y}}$;
  Delete existing input points (if any) in memory;

create an instance of DP-Merge-Reduce$_r$ (see Algorithm 4) which outputs a DP semicoreset per ring. If no new centers have been added to $\mathcal{F}$ in this timestep, then instead of creating new rings, the algorithm adds x_t to an existing ring (and corresponding DP-Merge-Reduce instance).

3. Release the union of the DP semicoresets as $\hat{\mathcal{Y}}$.

In order to keep our space usage small, we apply a $(1+\gamma)$-approximation non-DP coreset algorithm to the union of semicoresets of these rings as $\hat{\mathcal{Y}}$

Finally, in an offline postprocessing step, we apply a ρ-approximate non-DP clustering algorithm to $\hat{\mathcal{Y}}$.

technical reasons to ensure DP by avoiding the value of the input affecting the number of events in the sub-streams.

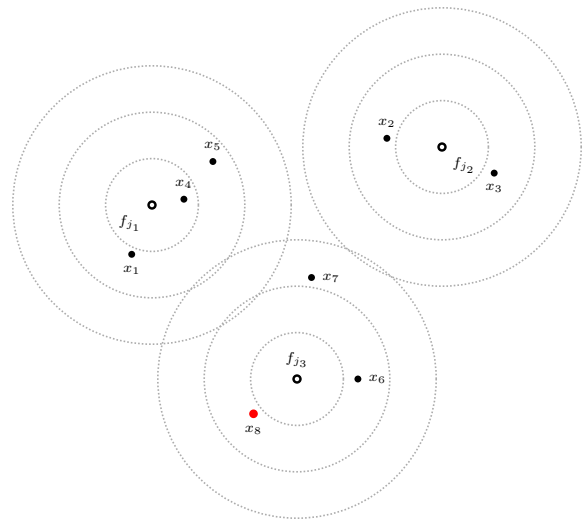

Figure 1: Illustration of Algorithm 1 at time $t = 8$. When the new point x_8 arrives, the algorithm: (1) updates the bicriteria solution $\mathcal{F} = \{f_{j_1}, f_{j_2}, f_{j_3}\}$; (2) creates concentric rings around each candidate center in $\mathcal{F}$ (or reuses existing rings) and assigns x_8 to the appropriate ring of its nearest center, maintaining a DP-Merge-Reduce$_r$ instance per ring; and (3) releases the union of the resulting DP semicoresets as $\hat{\mathcal{Y}}$.

Analysis. For the sake of analysis, we split the entire stream of points into *epochs* dictated by the addition of *new* centers to the bicriteria solution $\mathcal{F}$. Let $T_1, \ldots, T_e$ be the epochs such that for a fixed i, the set of bicriteria centers $\mathcal{F}$ over the timesteps $t \in T_i$ is fixed. Clearly $T_1 \cup \ldots \cup T_e = [T]$ and $T_1 \cap \ldots \cap T_e = \emptyset$.

We first state the theoretical guarantee of BicriteriaDPCenters (additional details in Appendix C) and DP-Merge-Reduce (additional details in Appendix D). We need these results to state the guarantees of our Main Algorithm (see Algorithm 1).

Theorem 4. *[BicriteriaDPCenters] Let $\mathcal{S} := \{x_1, \ldots, x_T\}$ be the stream of input points in Euclidean space. For $t \in [T]$, let $\mathcal{F}_t$ be the set of centers until time step t. Let $cost(\mathcal{F}, \mathcal{S}) := \sum_{t=1}^{T} cost(\mathcal{F}_t)$ where $cost(\mathcal{F}_t) := \min_{f \in \mathcal{F}_t} dist^2(x_t, f)$. There exists an algorithm BicriteriaDPCenters (see Algorithm 2) that outputs a set of centers $\mathcal{F}$ at every timestep $t \in [T]$ such that BicriteriaDPCenters*

1. *(Privacy) is ε-DP.*
2. *(Accuracy) With probability at least*

$1 - k^{-\operatorname{poly}(k, \log \Lambda)}$,

$$cost(\mathcal{F}, \mathcal{S}) \leq O(d^3) \cdot cost(C_{\mathcal{S}}^{opt}, \mathcal{S}) + \tilde{O}\big(\frac{d^2 \Lambda^2 k}{\varepsilon} \operatorname{poly}\big(\log\big(T \cdot k \cdot \Lambda\big)\big)\big)$$

where $cost(C_{\mathcal{S}}^{opt}, \mathcal{S})$ is the optimal k-means cost for $\mathcal{S}$.

3. *(Space) uses $\tilde{O}(k \operatorname{poly}(\log(T\Lambda k)))$space.*

4. *(Size) $\mathcal{F}$ has at most $\tilde{O}(k \log T)$centers.*

Theorem 5 (DP-Merge-Reduce)**.** *Let $0 < \xi < 1$, T be the length of the entire stream, ε, δ be privacy parameters, M be an arbitrary parameter such that $M > \frac{12}{\varepsilon} \log(\frac{2T}{\xi})$, and P be a sub-stream of non-empty points with length N.*

Suppose we are given black-box access to an offline $(\varepsilon_1, \delta_1)$-DP algorithm $\mathcal{A}$ that computes a (κ, η_1, η_2)-semicoreset of $X \subseteq \mathbb{R}^d$ of size $SZ_{\mathcal{A}}(N, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_A)$ using space $S_{\mathcal{A}}(N, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_A)$ with failure probability ξ_A. And we are given black-box access to an offline non-DP algorithm $\mathcal{B}$ that computes a $(1+\gamma)$-coreset of $X \subseteq \mathbb{R}^d$ of size $SZ_{\mathcal{B}}(N, k, d, \gamma, \xi_B)$ using space $S_{\mathcal{B}}(N, k, d, \gamma, \xi_B)$ with failure probability ξ_B. Then there exists an algorithm DP-Merge-Reduce (see Algorithm 4) in the streaming model such that

- *(Privacy) DP-Merge-Reduce is $(\varepsilon + \varepsilon_1, \delta_1)$-DP.*
- *(Accuracy) With probability $1 - \xi_A - \xi_B - \xi$, the semicoreset released by DP-Merge-Reduce is a $((1+\gamma)\kappa, (\frac{4N}{M} - 1)(1+\gamma)\eta + \tilde{M})$-semicoreset of P. Where $\tilde{M} := M + \frac{6}{\varepsilon} \log(\frac{2T}{\xi})$.*
- *(Space) DP-Merge-Reduce requires $S_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_A) + S_{\mathcal{B}}(SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_A), k, d, \gamma, \xi_B) + \lceil \log(2N/M) \rceil \cdot S_{\mathcal{B}}(SZ_{\mathcal{B}}(M, k, d, \gamma, \xi_B), k, d, \gamma, \xi_B) + 3M/2$ space.*
- *(Size) The resulting coreset has size at most $O(k \log k \cdot \gamma^{-4})$.*

Remark. The parameter M denotes the block size of the base level in DP-Merge-Reduce (see Algorithm 4). We treat M as an arbitrary parameter in Theorem 5, Lemma 2 and Lemma 4. However we set M to be a function of α (the multiplicative approximation error of the bicriteria solution), η_2 (the additive approximation error of the DP semicoreset algorithm) and a parameter C_M in the proof of Theorem 13, i.e., $M := \frac{\alpha \eta_2}{C_M}$. To obtain an $O(1)$-multiplicative approximation in Theorem 2, we assign $C_M = O(1)$. To obtain a $(1+\gamma')$-multiplicative approximation in Theorem 3, we assign $C_M = \gamma/2$. See proof of Theorem 11 and Theorem 12 for details.

Privacy. We first show that the output of our main algorithm is indeed differentially private.

Lemma 1. *Let the underlying DP semicoreset algorithm used as a blackbox be $(\varepsilon_1, \delta_1)$-DP, then Algorithm 1 is $(\varepsilon_1 + 2\varepsilon, \delta_1)$-DP under the continual release setting.*

Proof. We first observe that for a fixed input point x_t, we can view Algorithm 1 as two main steps (1) x_t is first used to update the subroutine BicriteriaDPCenters which produces a set of candidate centers $\mathcal{F}$ (2) x_t is assigned to an appropriate ring R_r (defined according to $\mathcal{F}$) and added to the corresponding DP-Merge-Reduce$_{R_r}$ instance of that ring which produces a DP semicoreset (3) Finally, the algorithm releases the union of these DP semicoresets over disjoint rings at timestep t.

Step (1) involves x_t being processed by BicriteriaDP-Centers which is ε-DP under the continual release setting (by Theorem 4). In Step (2), note that although the rings themselves may overlap (as illlustrated in Figure 1), an input point is assigned to only one ring. Thus, since the rings partition the input space by definition, the corresponding DP-Merge-Reduce$_{R_r}$ instances are disjoint. Since the semicoreset output by each DP-Merge-Reduce$_{R_r}$ instance is $(\varepsilon_1 + \varepsilon, \delta_1)$-DP under the continual release setting (by Theorem 5), then by parallel composition (Item 2 of Theorem 6), the total privacy over all semicoresets output by DP-Merge-Reduce$_{R_r}$ instances (where $1 \leq r \leq \Lambda$) is $(\varepsilon_1 + \varepsilon, \delta_1)$-DP. In Step (3), the union of DP semicoresets is DP by postprocessing.

Thus, each step is private under the continual release setting and note that each step is carried out in the same fixed timestep t. By sequential composition (Item 1 of Theorem 6) of the privacy for Steps (1) and (2), and postprocessing of Step (3), our claim follows. □

3 CONCLUSION AND FUTURE WORK

To the best of our knowledge, in this paper we designed the *first sublinear space differentially private* algorithms for k-clustering in the continual release setting (sublinear in stream length T). Our paper gives a general DP clustering framework that uses any existing offline DP clustering or coreset algorithm as a blackbox and the space guarantee of the final algorithm depends on the additive error of the offline DP algorithm. In particular, we achieve a $(1+\gamma)$-multiplicative approximation using the DP coreset algorithm from Ghazi et al. [2020] with $\tilde{O}_{\gamma}(\operatorname{poly}(k, 2^{O_{\gamma}(d)}, \log(T)))$ space

and an $O(1)$-multiplicative approximation with $\tilde{O}(k^{1.5} \cdot \mathrm{poly}(d, \log(T)))$ space using the DP k-means algorithm from Stemmer and Kaplan [2018]. Further improving the dependence on dimension d is a natural and exciting next step, building on the framework we develop here.

Finally, while our work achieves sublinear space in the insertion-only streaming setting, recently la Tour et al. [2023] studied the DP clustering problem under insertions and deletions but their results do not optimize for space — an exciting open direction is to design sublinear space differentially private algorithms for k-clustering in this model.

References

Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004. doi: 10.1145/1008731.1008736. URL https://doi.org/10.1145/1008731.1008736. 4, 23

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, 49, 2020. 1

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999. 1

Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *STOC*, 2001. 1

Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *FOCS*, 2010. 1

Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional euclidean spaces. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 322–331. PMLR, 2017. 2, 4

Jeremiah Blocki, Elena Grigorescu, and Tamalika Mukherjee. Differentially-private sublinear-time clustering. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 332–337. IEEE, 2021. 2

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 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. 1

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011. doi: 10.1145/2043621.2043626. URL https://doi.org/10.1145/2043621.2043626. 5, 13

Alisa Chang, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Locally private k-means in one round. In *International Conference on Machine Learning*, pages 1441–1451. PMLR, 2021. 2

Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65, 2002. 1

Anamay Chaturvedi, Huy L. Nguyen, and Eric Z. Xu. Differentially private k-means via exponential mechanism and max cover. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021*, 2021. 4

Ke Chen. On k-median clustering in high dimensions. In *SODA*, 2006. 1

Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *SODA*, 2008. 1

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. doi: 10.1137/070699007. URL https://doi.org/10.1137/070699007. 1, 5

Edith Cohen, Haim Kaplan, Yishay Mansour, Uri Stemmer, and Eliad Tsfadia. Differentially-private clustering of easy instances. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2049–2059. PMLR, 18–24 Jul 2021. 2

Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 169–182. ACM, 2021. 13, 14, 23

Vincent Cohen-Addad, Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Andres Munoz Medina, David Saulpic, Chris Schwiegelshohn, and Sergei Vassilvitskii. Scalable differentially private clustering via hierarchically separated trees. In *KDD*, page 221–230,

New York, NY, USA, 2022a. Association for Computing Machinery. ISBN 9781450393850. 2

Vincent Cohen-Addad, Alessandro Epasto, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. Near-optimal private and scalable k-clustering. In *Advances in Neural Information Processing Systems*, 2022b. 2, 6, 13, 20

Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1038–1051, 2022c. 1, 4

Vincent Cohen-Addad, David P. Woodruff, and Samson Zhou. Streaming euclidean k-median and k-means to a (1 + eps)-approximation with $o_{k,eps}$(log n) memory words. In *FOCS*, 2023. 1

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. doi: 10.1561/0400000042. URL `https://doi.org/10.1561/0400000042`. 13

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 381–390. ACM, 2009. 5

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 715–724. ACM, 2010. doi: 10.1145/1806689.1806787. URL `https://doi.org/10.1145/1806689.1806787`. 5, 13, 19

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7, 2016. 1, 2

Alessandro Epasto, Jieming Mao, Andres Muñoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. *CoRR*, abs/2301.05605, 2023. doi: 10.48550/arXiv.2301.05605. URL `https://doi.org/10.48550/arXiv.2301.05605`. 5, 6, 13, 19, 24

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578, 2011. 1

Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 361–370. ACM, 2009. 2, 4, 5

Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2017, Pittsburgh, PA, USA, April 18-21, 2017*, 2017. 2, 4

Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal on Computing*, 49(3):601–657, 2020. 4, 23

Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2, 3, 4, 6, 8, 14, 23

Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1106–1125. SIAM, 2010. 2, 4, 5

Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 126–134, 2005. 1

Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, 2004. 1, 4, 23

Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, Las Vegas, Nevada, USA, October 23-26, 2010*, pages 61–70. IEEE Computer Society, 2010. 5

Zhiyi Huang and Jinyan Liu. Optimal differentially private algorithms for k-means clustering. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 395–408. ACM, 2018. 2

Max Dupré la Tour, Monika Henzinger, and David Saulpic. Making old things new: A unified algo-

rithm for differentially private clustering. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. 4

Max Dupré la Tour, Monika Henzinger, and David Saulpic. Differential privacy for clustering under continual observation, 2023. 4, 9

Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45, 2016. 1

Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009. 13

Huy L. Nguyen. A note on differentially private clustering with large additive error. *CoRR*, abs/2009.13317, 2020. URL `https://arxiv.org/abs/2009.13317`. 4

Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *Algorithmic Learning Theory, ALT 2018, 7-9 April 2018, Lanzarote, Canary Islands, Spain*, volume 83 of *Proceedings of Machine Learning Research*, pages 619–653. PMLR, 2018. 2, 4

Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84. ACM, 2007. 2

Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59, 2012. 1

David Saulpic, Vincent Cohen-Addad, and Andreas Emil Feldmann. Near-linear time approximations schemes for clustering in doubling metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 540–559. IEEE Computer Society, 2019. doi: 10.1109/FOCS.2019.00041. URL `https://doi.org/10.1109/FOCS.2019.00041`. 3

Uri Stemmer and Haim Kaplan. Differentially private k-means with constant multiplicative error. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018. 2, 3, 4, 6, 9, 14, 23

Chang Xia, Jingyu Hua, Wei Tong, and Sheng Zhong. Distributed k-means clustering guaranteeing local differential privacy. *Computers & Security*, 90:101699, 2020. 2

Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In *4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 12)*, 2012. 1

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Not Applicable**

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes**
   (b) Complete proofs of all theoretical results. **Yes**
   (c) Clear explanations of any assumptions. **Yes**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Not Applicable**
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Not Applicable**
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Not Applicable**
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Not Applicable**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. **Not Applicable**

(b) The license information of the assets, if applicable. **Not Applicable**

(c) New assets either in the supplemental material or as a URL, if applicable. **Not Applicable**

(d) Information about consent from data providers/curators. **Not Applicable**

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. **Not Applicable**

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

A PRELIMINARIES

Norms and heavy hitters. Let $p \geq 1$, the ℓ_p-norm of a vector $\mathbf{x} = (x_1, \ldots, x_t)$ is defined as $\|\mathbf{x}\|_p = (\sum_{i=1}^{t} |x_i|^p)^{1/p}$. Given a multiset $\mathcal{S}$, denote the frequency of an item x appearing in $\mathcal{S}$ as $f(x)$. We say that an item x is an θ-heavy hitter (θ-HH for short) if $f(x) \geq \theta \|\mathcal{S}\|_1$.

Differential Privacy background. We state some DP theorems that we frequently utilize below. For a more detailed background on DP refer to Dwork and Roth [2014].

Theorem 6 (Composition Theorems McSherry [2009])**.**

1. *(Sequential) Let M_i each provide $(\varepsilon_i, \delta_i)$-DP. The sequence of $M_i(X)$ provides $(\sum_i \varepsilon_i, \sum_i \delta_i)$-DP.*
2. *(Parallel) Let M_i each provide (ε, δ)-DP. Let D_i be arbitrary disjoint subsets of input domain D. The sequence of $M_i(X \cap D_i)$ provides (ε, δ)-DP.*

Theorem 7 (Binary Mechanism BM Chan et al. [2011], Dwork et al. [2010])**.** *Let $\varepsilon \geq 0, \gamma \in (0, 0.5)$, there is an ε-DP algorithm for the sum of the stream in the continual release model. With probability $1 - \xi$, the additive error of the output for every timestamp $t \in [T]$ is always at most $O(\frac{1}{\varepsilon} \log^{2.5}(T) \log(\frac{1}{\xi}))$ and uses $O(\log T)$ space.*

Theorem 8 (DP-HH algorithm Epasto et al. [2023])**.** *Let $\varepsilon > 0$, $\gamma_h \in (0, 0.5)$, $0 < \theta < 1$, $\xi \in (0, 0.5)$. There is an ε-DP algorithm in the streaming continual release model such that with probability at least $1 - \xi$, it always outputs a set $H \subseteq \mathcal{U}$ and a function $\hat{f} : H \to \mathbb{R}$ for every timestamp $t \in [T]$ such that*

1. *$\forall a \in H$, $\hat{f}(a) \in (1 \pm \gamma_h) \cdot f_a$ where f_a is the frequency of a in the stream $\mathcal{S} = (a_1, a_2, \ldots, a_t)$*
2. *$\forall a \in \mathcal{U}$, if $f_a \geq \frac{1}{\varepsilon \gamma_h} \operatorname{poly}\big(\log\big(\frac{T \cdot |\mathcal{U}|}{\theta \xi \gamma_h}\big)\big)$ and $f_a^1 \geq \theta \|\mathcal{S}\|_1$ then $a \in H$*
3. *The size of H is at most $O((\log(T/\xi) + \log|\mathcal{U}|) \cdot (\frac{1+\gamma_h}{1-\gamma_h}) \cdot \frac{1}{\theta})$*

The algorithm uses $\frac{1}{\gamma_h^2 \theta^3} \operatorname{poly}\left(\log\left(\frac{T \cdot |\mathcal{U}|}{\xi \theta}\right)\right)$ space.

Remark. In Algorithm 3 we use the notation **Initialize** and **Update** for the Binary Mechanism (BM) and DP-Heavy Hitters (DP-HH) primitives. For clarity:

- **Initialize** sets up the respective data structure with its privacy budget and parameters (e.g., stream length, thresholds).
- **Update** processes a single incoming item (or the empty symbol $\perp$) and returns the current private estimate. For BM this is a differentially private count, while for DP-HH it is a set of candidate heavy hitters together with approximate frequencies.

We do not re-write these algorithms explicitly, since our use is a direct adaptation of the continual release algorithms already stated in Theorem 7, Theorem 8 and in the cited references.

Useful Clustering Theorems. We present some useful clustering theorems that we frequently utilize below.

We clarify that the statement below concerns sequential algorithms in the MPC model. However, when the entire input resides on a single machine, a sequential algorithm is trivially static. Since our use case involves static algorithms in such a setting, we restate the lemma accordingly to reflect this context more accurately.

Theorem 9 (Lemma C.1 in Cohen-Addad et al. [2022b])**.** *Let $\mathcal{A}$ be an (ε, δ)-DP (static) algorithm that takes as input a dataset P of size at least $\Omega(k \cdot \varepsilon^{-1} \log(n))$ contained in a ball $\mathcal{B}(0, \Lambda)$ (with known center) and outputs a set of k centers $\mathcal{C}$ contained in $\mathcal{B}$ such that $cost(\mathcal{C}, \mathcal{X}) \leq O(1) cost(\mathcal{C}^{opt}, \mathcal{X}) + V(n, d, k, \varepsilon, \delta) \cdot \Lambda^p$. Then for any fixed $\eta > 0$ there exists a $(3\varepsilon, \delta)$-DP (sequential) algorithm $\mathcal{B}$ on the dataset $\mathcal{Y}$ that outputs a set $\hat{P} \subset B$ such that for any set $\mathcal{C}$ of size at most k.*

$$
\begin{aligned}
cost(\mathcal{C}, \hat{P}) \leq\ & O(1) \cdot (cost(\mathcal{C}_P^{opt}, P) \\
& + cost(\mathcal{C}, P) + (V(n, d, k, \varepsilon, \delta) \\
& + k\varepsilon^{-1} \log(n)) \cdot \Lambda^p) \qquad (1) \\
cost(\mathcal{C}, P) \leq\ & O(1) \cdot (cost(\mathcal{C}_P^{opt}, P) \\
& + cost(\mathcal{C}, \hat{P}) + (V(n, d, k, \varepsilon, \delta) \\
& + k\varepsilon^{-1} \log(n)) \cdot \Lambda^p) \qquad (2)
\end{aligned}
$$

In addition constructing the set requires only $O(k^{1+\eta} \cdot d \cdot \log(n))$ space and at most $O(k^{\eta} \cdot d \cdot \log(n))$ time per point in P for some arbitrarily chosen constant $\eta > 0$.

Theorem 10 (Cohen-Addad et al. [2021])**.** *There exists a non-DP $(1 + \gamma)$-coreset of size $O(k \log k \cdot (\gamma^{-2-\max(2,z)}) \cdot 2^{O(z \log z)} \cdot \operatorname{poly} \log(\gamma^{-1}))$ for (k, z)-clustering in Euclidean spaces.*

B PROOFS OF DP CLUSTERING FRAMEWORK

We now present the proof of the guarantees of our DP clustering framework (Algorithm 1) stated in Theorem 1 below.

Proof. The privacy guarantee follows from Lemma 1, and the accuracy guarantee follows from Theorem 14. The space guarantee follows from Lemma 2, where we

simplify the second term $S_{\mathcal{B}}(SZ_{\mathcal{A}}(\cdot))$ in the statement of Lemma 2 by noting that the space used by the non-DP coreset algorithm $\mathcal{B}$ is linear in the semicoreset size of the algorithm $\mathcal{A}$. □

Next we present the proofs of Theorem 2 and Theorem 3. We first restate the existing offline DP clustering and coreset results that we use as a blackbox from Stemmer and Kaplan [2018] and Ghazi et al. [2020] in Theorem 11 and Theorem 12. Note that in Theorem 2, we need to convert the set of k centers obtained from applying the DP clustering algorithm in Theorem 11 into a semicoreset via Theorem 9.

Theorem 11 (Theorem 3.10 in Stemmer and Kaplan [2018])**.** *There is an (ε, δ)-DP algorithm that takes a database S containing n points in the d-dimensional ball $\mathcal{B}(0, \Lambda)$, and outputs with probability $1 - \beta$, a (γ, η)-approximation for the k-means of S where $\gamma = O(1)$ and $\eta = O((k^{1.01} \cdot d^{0.51} + k^{1.5}) \cdot \mathrm{poly}(\log(n), \log(\frac{1}{\beta}), \log(\frac{1}{\delta}), \frac{1}{\varepsilon}))$*

The proof of Theorem 2 is stated below.

Proof. We first note that using Theorem 9 together with Theorem 11 gives us an algorithm $\mathcal{A}$ that produces a (κ, η_1, η_2)-semicoreset where $\kappa = O(1), \eta_1 = O(1)$ and $\eta_2 = O((k^{1.01} \cdot d^{0.51} + k^{1.5} + k) \cdot \mathrm{poly}(\log(T) \log(\frac{1}{\delta}), \frac{1}{\varepsilon}))$ and that we can use as a black-box in our DP clustering framework. The privacy guarantee follows by observing that the resulting algorithm from Theorem 9 is $(3\varepsilon, \delta)$-DP and applying Item 1 of Theorem 1. We choose $C_M = 100$ and the accuracy guarantee follows from Item 2 of Theorem 1.

Finally, for the space usage, we first determine the value of M in Theorem 1. Because $\eta_2 = O((k^{1.01} \cdot d^{0.51} + k^{1.5}) \cdot \mathrm{poly}(\log(T) \log(\frac{1}{\delta}), \frac{1}{\varepsilon}))$ and $C_M = 100$, we have that $M = O((k^{1.01} \cdot d^{3.51} + d^3 \cdot k^{1.5}) \cdot \mathrm{poly}(\log(T), \log(\frac{1}{\delta}), \frac{1}{\varepsilon}))$.

Note that $S_{\mathcal{A}}(M, k, d) = O((M + k)d) + O(k^{1+\eta} \cdot d \cdot \log(M))$ for some arbitrarily chosen constant $\eta > 0$ where the second term comes from the space usage of Theorem 9. Also $SZ_{\mathcal{A}}(\cdot) = k$. The final space complexity is obtained by plugging in the value of M and setting $\eta = 0.01$. □

Theorem 12 (Lemma 16 in Ghazi et al. [2020])**.** *There is a $2^{O_\gamma(d)} \mathrm{poly}(n)$-time ε-DP algorithm that with probability 0.99 produces an $\left(1 + \gamma, O_\gamma\left(\frac{k^2 2^{O_\gamma(d)}}{\varepsilon} \mathrm{poly} \log n\right)\right)$-coreset for k-means (and k-median). The size of the coreset is $2^{O_\gamma(d)} \cdot \mathrm{poly}(k, \log n)$.*

The proof of Theorem 3 is stated below.

Proof. First note that the ε-DP coreset from Theorem 12 Ghazi et al. [2020] is a $(\kappa, 0, \eta)$-semicoreset where $\kappa = 1 + \gamma$ and $\eta = O_\gamma\left(\frac{k^2 2^{O_\gamma(d)}}{\varepsilon} \mathrm{poly} \log(T)\right)$ and thus we can use it directly as a blackbox in our DP clustering framework. The privacy guarantee follows from Item 1 of Theorem 1. The accuracy guarantee follows from Item 2 of Theorem 1 where we choose $C_M = \gamma/2$.

Finally, for the space usage, we first note that in Theorem 1, by plugging in the values of C_M and η, we have that $M = O_\gamma\left(\frac{d^3 k^2 2^{O_\gamma(d)}}{\varepsilon} \mathrm{poly} \log T\right)$. By observing that $S_{\mathcal{A}}(M, d, k) = \mathrm{poly}(M, d, k)$ and total coreset size $SZ_{\mathcal{A}}(\cdot) = 2^{O_\gamma(d)} \cdot \mathrm{poly}(k, \log T)$ and plugging in the value of M in Item 3 of Theorem 1, the space guarantee follows. □

B.1 Space and Size

We present the space (see Lemma 2) and coreset size (see Lemma 3) of Algorithm 1 in terms of the space and coreset size of the underlying DP semicoreset algorithm $\mathcal{A}$ and non-DP coreset algorithm $\mathcal{B}$ used in DP-Merge-Reduce. We use the algorithm from Cohen-Addad et al. [2021] as our non-DP coreset algorithm $\mathcal{B}$ whose guarantees are given by Theorem 10.

Lemma 2. *Algorithm 1 consumes space*

$$\begin{aligned}
&\log(\Lambda) \cdot (S_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_{\mathcal{A}}) \\
&+ S_{\mathcal{B}}(SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_{\mathcal{A}}), k, d, \gamma, \xi_B) \\
&+ 3M/2) + \tilde{O}(k \log(T/M) \log(\Lambda) \log k \cdot \gamma^{-4}) \\
&+ O(k \log(k) \log^2(\Lambda) \log T) \\
&+ O(k \log(\Lambda) \log^2(k) \mathrm{poly}\left(\log\left(T\Lambda k\right)\right))
\end{aligned}$$

Proof. The last term in the above claim is the total space used by BicriteriaDPCenters and the second last term is the space used to store the bicriteria solution $\mathcal{F}$ (see Theorem 8). We focus on proving that the DP-Merge-Reduce instances consume the space specified by the first term of the above claim.

We sometimes abuse notation and omit the input for the non-DP coreset size $SZ_{\mathcal{B}}$ as by Theorem 10 we know that $SZ_{\mathcal{B}}(\cdot) = \tilde{O}(k \log(k) \cdot \gamma^{-4})$. For a fixed epoch T_i, recall that the set of centers $\mathcal{F}_t$ is fixed for $t \in T_i$. According to Algorithm 1, at timestep $t \in T_i$, we compute $\log(\Lambda)$ instances of DP-Merge-Reduce in parallel. By Theorem 5, since the space required to compute semicoreset $\hat{\mathcal{Y}}_r^{(t)}$ using DP-Merge-Reduce is $S_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_{\mathcal{A}}) + S_{\mathcal{B}}(SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_{\mathcal{A}}), k, d, \gamma, \xi_{\mathcal{B}}) + \lceil \log(2N/M) \rceil \cdot S_{\mathcal{B}}(SZ_{\mathcal{B}}(\cdot), k, d, \gamma, \xi_{\mathcal{B}}) + 3M/2$, the

total space at timestep $t \in T_i$ is

$$\begin{aligned}
&\sum_{r=1}^{\log(\Lambda)} (S_{\mathcal{A}}(M,k,d,\varepsilon,\delta,\kappa,\eta_1,\eta_2,\xi_{\mathcal{A}}) \\
&+ S_{\mathcal{B}}(SZ_{\mathcal{A}}(M,k,d,\varepsilon,\delta,\kappa,\eta_1,\eta_2,\xi_{\mathcal{A}}),k,d,\gamma,\xi_{\mathcal{B}}) \\
&+ \lceil \log(2N_r^{(t)}/M) \rceil \cdot S_{\mathcal{B}}(SZ_{\mathcal{B}}(\cdot),k,d,\gamma,\xi_{\mathcal{B}}) + 3M/2) \\
&= \log(\Lambda) \cdot (S_{\mathcal{A}}(M,k,d,\varepsilon,\delta,\kappa,\eta_1,\eta_2,\xi_{\mathcal{A}}) \\
&+ S_{\mathcal{B}}(SZ_{\mathcal{A}}(M,k,d,\varepsilon,\delta,\kappa,\eta_1,\eta_2,\xi_{\mathcal{A}}),k,d,\gamma,\xi_{\mathcal{B}}) \\
&+ 3M/2) \\
&+ S_{\mathcal{B}}(SZ_{\mathcal{B}}(\cdot),k,d,\gamma,\xi_{\mathcal{B}}) \sum_{r=1}^{\log(\Lambda)} \lceil \log(2N_r^{(t)}/M) \rceil \qquad (3)
\end{aligned}$$

We focus on the last term in eq. (3). In particular, since the space used by $\mathcal{B}$ is linear in the coreset size, we have that $S_{\mathcal{B}}(SZ_{\mathcal{B}}(\cdot),k,d,\gamma) = \tilde{O}(k \log k \cdot \gamma^{-4})$. Next we simplify the adjoining sum

$$\begin{aligned}
&\sum_{r=1}^{\log(\Lambda)} \lceil \log(\frac{2N_r^{(t)}}{M}) \rceil \leq \sum_{r=1}^{\log(\Lambda)} (\log(\frac{2N_r^{(t)}}{M}) + 1) \\
&= \log\left(\frac{2^{\log(\Lambda)}}{M^{\log(\Lambda)}} \cdot \prod_{r=1}^{\log(\Lambda)} N_r^{(t)}\right) + \log(\Lambda) \\
&< \log\left(\frac{\Lambda \cdot 2^{\log(\Lambda)}}{M^{\log(\Lambda)}} \cdot T^{\log(\Lambda)}\right) \\
&= \log(\Lambda)(2 + \log(T) - \log(M)) \qquad (4)
\end{aligned}$$

where we use the fact that the number of points in any ring N_r cannot be larger than T in the second last step.

□

Lemma 3. *Algorithm 1 releases a coreset of size at most* $\tilde{O}(k \log(k) \cdot \gamma^{-4})$.

Proof. Consider a fixed epoch T_i. By Theorem 5, the coreset $\hat{\mathcal{Y}}_r^{(t)}$ has size $\tilde{O}(k \log(k) \cdot \gamma^{-4})$ for $t \in T_i$. Since we run a non-DP coreset algorithm at the end of each epoch (whose theoretical guarantees are given by Theorem 10), the size of the coreset at the end of the epoch is also $\tilde{O}(k \log(k) \cdot \gamma^{-4})$. Recall that an epoch is created every time a new center is added to $\mathcal{F}$, therefore the total size of the coreset is given by $|\mathcal{F}| \cdot \tilde{O}(k \log(k) \cdot \gamma^{-4})$. But since we apply another non-DP coreset algorithm to $\hat{\mathcal{Y}}$ before releasing it, the coreset size is $\tilde{O}(k \log(k) \cdot \gamma^{-4})$. □

B.2 Accuracy

We analyze the accuracy of Algorithm 1 in the sequel. We first state the accuracy guarantee of the DP semi-coreset $\hat{\mathcal{Y}}_r^{(t)}$ released by DP-Merge-Reduce$_r$ for each ring $R_r^{(t)}$ at timestep $t \in [T]$ in Lemma 4. We defer the detailed proofs to Appendix B.

Lemma 4. *Let* $\hat{\mathcal{Y}}_r^{(t)}$ *be the output of DP-Merge-and-Reduce*$_r$ *(see Algorithm 1) and* $N_r^{(t)}$ *be the number of (non-empty) points in* $R_r^{(t)}$ *at timestep* $t \in [T]$. *Then* $\hat{\mathcal{Y}}_r^{(t)}$ *is a DP* $((1+\gamma)\kappa, (1+\gamma)\eta_1, (\frac{4N_r^{(t)}}{M} - 1)(1+\gamma)\eta_2 + \tilde{M})$*-semicoreset of* $R_r^{(t)}$ *at timestep* t. *Where* $\tilde{M} := M + \frac{6}{\varepsilon}\log(\frac{2T}{\xi})$. *In other words, for any set of* k *centers* $\mathcal{C}$*, with probability* $1 - 3\xi$,

$$\begin{aligned}
&\frac{(1-\gamma)}{\kappa} \cdot \mathsf{cost}(\mathcal{C}, R_r^{(t)}) - (1-\gamma)\eta_1 \cdot \mathit{cost}(\mathcal{C}^{opt}_{R_r^{(t)}}, R_r^{(t)}) \cdot (2^r)^2 \\
&- ((\frac{4N_r^{(t)}}{M} - 1)(1+\gamma)\eta_2 + \tilde{M}) \cdot (2^r)^2 \\
&\leq \mathsf{cost}(\mathcal{C}, \hat{\mathcal{Y}}_r^{(t)}) \\
&\leq (1+\gamma)\kappa \cdot \mathsf{cost}(\mathcal{C}, R_r^{(t)}) \\
&+ ((\frac{4N_r^{(t)}}{M} - 1)(1+\gamma)\eta_2 + \tilde{M}) \cdot (2^r)^2 \qquad (5)
\end{aligned}$$

Proof. The lemma follows from the accuracy guarantees of DP-Merge-Reduce (see Theorem 5) and the fact that the additive cost incurred is proportional to the radius of the ring R_r which is given by 2^r. □

Next, we show that the union of the DP semicoresets over all rings $\hat{\mathcal{Y}}$ is a semicoreset for the stream $\mathcal{S}$.

Theorem 13. *Given dimension* d*, clustering parameter* k*, arbitrary parameter* C_M*, non-DP* $(1+\gamma)$*-coreset,* $O(d^3)$*-approximate bicriteria solution from Theorem 4, DP* (κ, η_1, η_2)*-semicoreset, and privacy parameter* ε. *Let* $\hat{\mathcal{Y}}$ *be the output of Algorithm 1 for stream* $\mathcal{S} = \{x_1, \ldots, x_T\}$. *Then for any set of* k *centers* $\mathcal{C}$*, we have that with probability* $1 - \frac{1}{T^2} - \frac{1}{k^{O(\mathrm{poly}(k, \log(\Lambda))}}$*, the following holds*

$$\begin{aligned}
&\frac{(1-\gamma)^2}{\kappa}\mathsf{cost}(\mathcal{C}, \mathcal{S}) - ((1-\gamma)^2\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)) \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \qquad (6) \\
&- C_M(1-\gamma^2) \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&- (1-\gamma)V''(d,k,\varepsilon,T,\Lambda) \\
&\leq \mathsf{cost}(\mathcal{C}, \hat{\mathcal{Y}}) \\
&\leq (1+\gamma)^2\kappa\mathsf{cost}(\mathcal{C}, \mathcal{S}) + 16C_M(1+\gamma)^2\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1+\gamma)^2 \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1+\gamma)V''(d,k,\varepsilon,T,\Lambda) \qquad (7)
\end{aligned}$$

where $V'(d,k,\varepsilon,T,\Lambda) = O\left(\frac{\Lambda^2 k^2}{d\varepsilon}\log^2(\Lambda)\log^4(k)\log(T)\,\mathrm{poly}\log(T \cdot k \cdot \Lambda)\right)$,

$V''(d,k,\varepsilon,T,\Lambda) = O(k\Lambda^2\log^2(k)\log(\Lambda)\log T) \cdot (\frac{\alpha\eta_2}{C_M} + \tilde{O}(\frac{1}{\varepsilon}\cdot \mathrm{poly}(\log(T\cdot k\cdot\Lambda))$, *and* $\alpha := O(d^3)$.

Proof. Fix an epoch T_i. Let $\hat{\mathcal{Y}}^{(t)}$ be the output of Algorithm 1 at timestep $t \in T_i$. Since the centers in $\mathcal{F}|_{T_i}$ and consequently the rings R_r are fixed, we can compute the cost of $\hat{\mathcal{Y}}^{(t)}$ by summing eq. (5) from Lemma 4 over all rings R_r where $1 \le r \le \log(\Lambda)$ as follows:

$$\begin{aligned}
&\frac{(1-\gamma)}{\kappa}\sum_r \mathsf{cost}(\mathcal{C}, R_r^{(t)}) \\
&-(1-\gamma)\eta_1\sum_r \mathsf{cost}(\mathcal{C}^{opt}_{R_r^{(t)}}, R_r^{(t)})\cdot(2^r)^2 \\
&-(1+\gamma)\eta_2\sum_r(\frac{4N_r^{(t)}}{M}-1)\cdot(2^r)^2 - \sum_r \tilde{M}\cdot(2^r)^2 \\
&\le \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}^{(t)}) \\
&\le (1+\gamma)\kappa\sum_r \mathsf{cost}(\mathcal{C}, R_r^{(t)}) \\
&+(1+\gamma)\eta_2\sum_r(\frac{4N_r^{(t)}}{M}-1) + \sum_r \tilde{M}\cdot(2^r)^2 \qquad (8)
\end{aligned}$$

Next, we take the union of $\hat{\mathcal{Y}}^{(t)}$ over all timesteps in epoch T_i. Note that for a fixed epoch and fixed ring R_r, the additive error $\tilde{M}$ is incurred at most once since this error stems from privately testing if the number of points at the base level is larger than the block size M (see Algorithm 5) in $\mathsf{DP\text{-}Merge\text{-}Reduce}_r$. Thus summing eq. (8) over all timesteps in T_i gives us

$$\begin{aligned}
&\frac{(1-\gamma)}{\kappa}\mathsf{cost}(\mathcal{C},\mathcal{S}|_{T_i}) - (1-\gamma)\eta_1\Lambda^2\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) \\
&-(1+\gamma)\eta_2\sum_{t\in T_i}\sum_r(\frac{4N_r^{(t)}}{M}-1)\cdot(2^r)^2 - \tilde{M}\cdot\frac{4}{3}(\Lambda^2-1) \\
&\le \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}^{(t)}|_{t\in T_i}) \\
&\le (1+\gamma)\kappa\mathsf{cost}(\mathcal{C},\mathcal{S}|_{T_i}) \\
&+(1+\gamma)\eta_2\sum_{t\in T_i}\sum_r(\frac{4N_r^{(t)}}{M}-1) + \tilde{M}\cdot\frac{4}{3}(\Lambda^2-1) \qquad (9)
\end{aligned}$$

where $\mathcal{S}|_{T_i}$ denotes the input points in stream $\mathcal{S}$ restricted to epoch T_i and $\hat{\mathcal{Y}}^{(t)}|_{t\in T_i}$ is defined analogously.

Lemma 5. *If x_i is assigned to the ring R_r, let $r(x_i) := 2^r$. Then*

$$\begin{aligned}
\sum_{x_i\in\mathcal{S}} r(x_i)^2 \le \sum_{x_i\in\mathcal{S}} 4d(x_i,\mathcal{F})^2 &= 4\mathsf{cost}(\mathcal{F},\mathcal{S}) \\
&\le 4\alpha\cdot\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}},\mathcal{S}) \\
&+ V(d,k,\varepsilon,T,\Lambda) \quad (10)
\end{aligned}$$

where $\alpha := O(d^3)$ and $V(d,k,\varepsilon,T,\Lambda) = O\left(\frac{d^2\Lambda^2 k\log(\Lambda)\log^2(k)}{\varepsilon}\mathrm{poly}\big(\log(T\cdot k\cdot\Lambda)\big)\right)$ are the multiplicative/additive factors from the bicriteria approximation (see Theorem 4).

Proof. The statement follows immediately from the definition of the ring R_r (Definition 3) and the accuracy guarantees of the bicriteria solution (Theorem 4). □

Observe that $\sum_{t\in T_i}\sum_r N_r^{(t)}\cdot(2^r)^2 = \sum_{x\in\mathcal{S}|_{T_i}} r(x)^2$, thus we can use Lemma 5 to simplify the additive error in eq. (9) to obtain

$$\begin{aligned}
&\frac{(1-\gamma)}{\kappa}\mathsf{cost}(\mathcal{C},\mathcal{S}|_{T_i}) - (1-\gamma)\eta_1\Lambda^2\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) \\
&-\frac{4}{M}(1+\gamma)\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) + V(d,k,\varepsilon,T,\Lambda)) \\
&-\tilde{M}\cdot\frac{4\Lambda^2}{3} \\
&\le \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}^{(t)}|_{t\in T_i}) \\
&\le (1+\gamma)\kappa\mathsf{cost}(\mathcal{C},\mathcal{S}|_{T_i}) \\
&+\frac{4}{M}(1+\gamma)\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) \\
&+V(d,k,\varepsilon,T,\Lambda)) + \tilde{M}\cdot\frac{4\Lambda^2}{3} \qquad (11)
\end{aligned}$$

Since a new epoch starts whenever a new center is added to the bicriteria solution $\mathcal{F}$, and we run a $(1+\gamma)$-approx non-DP coreset algorithm on the coreset $\hat{\mathcal{Y}}$ at the beginning of a new epoch, denote this new coreset as $\hat{\mathcal{Y}}_{T_i}$. We state the guarantees of $\hat{\mathcal{Y}}_{T_i}$ below.

$$\begin{aligned}
&\frac{(1-\gamma)^2}{\kappa}\mathsf{cost}(\mathcal{C},\mathcal{S}|_{T_i}) - (1-\gamma)^2\eta_1\Lambda^2\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) \\
&-\frac{4}{M}(1-\gamma^2)\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) + V(d,k,\varepsilon,T,\Lambda)) \\
&-\frac{4\Lambda^2(1-\gamma)\tilde{M}}{3} \\
&\le \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}_{T_i}) \\
&\le (1+\gamma)^2\kappa\mathsf{cost}(\mathcal{C},\mathcal{S}|_{T_i}) \\
&+\frac{4}{M}(1+\gamma)^2\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}|_{T_i}},\mathcal{S}|_{T_i}) \\
&+V(d,k,\varepsilon,T,\Lambda)) + \frac{4\Lambda^2(1+\gamma)\tilde{M}}{3} \qquad (12)
\end{aligned}$$

Observe that the total number of epochs is bounded by the total number of centers in $\mathcal{F}$ at the end of the stream. By Theorem 4 we know that $|\mathcal{F}| = O(k\log^2(k)\log(\Lambda)\log T)$. Therefore, we can sum eq. (12) over all epochs to have that for any set of k centers $\mathcal{C}$:

$$\begin{aligned}
&\frac{(1-\gamma)^2}{\kappa}\mathsf{cost}(\mathcal{C},\mathcal{S}) - (1-\gamma)^2\eta_1\Lambda^2\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&- \frac{4}{M}(1-\gamma^2)\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&+ V^*(d,k,\varepsilon,T,\Lambda)) - |\mathcal{F}|\cdot\frac{4\Lambda^2(1-\gamma)\tilde{M}}{3} \\
&\leq \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}) \\
&\leq (1+\gamma)^2\kappa\mathsf{cost}(\mathcal{C},\mathcal{S}) + \frac{4}{M}(1+\gamma)^2\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&+ V^*(d,k,\varepsilon,T,\Lambda)) + |\mathcal{F}|\cdot\frac{4\Lambda^2(1+\gamma)\tilde{M}}{3} \qquad (13)
\end{aligned}$$

where $V^*(d,k,\varepsilon,T,\Lambda) = O\left(\frac{d^2\Lambda^2k^2}{\varepsilon}\log^2(\Lambda)\log^4(k)\log(T)\,\mathrm{poly}\left(\log\left(T\cdot k\cdot\Lambda\right)\right)\right)$.

Grouping like terms together, we have

$$\begin{aligned}
&\frac{(1-\gamma)^2}{\kappa}\mathsf{cost}(\mathcal{C},\mathcal{S}) - ((1-\gamma)^2\eta_1\Lambda^2 \\
&+ \frac{16\alpha}{M}(1-\gamma^2)\eta_2)\cdot\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&- \frac{4}{M}(1-\gamma^2)\eta_2\cdot V^*(d,k,\varepsilon,T,\Lambda)) - |\mathcal{F}|\cdot\frac{4\Lambda^2(1-\gamma)\tilde{M}}{3} \\
&\leq \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}) \\
&\leq (1+\gamma)^2\kappa\mathsf{cost}(\mathcal{C},\mathcal{S}) + \frac{4}{M}(1+\gamma)^2\eta_2(4\alpha\cdot\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&+ V^*(d,k,\varepsilon,T,\Lambda)) + |\mathcal{F}|\cdot\frac{4\Lambda^2(1+\gamma)\tilde{M}}{3} \qquad (14)
\end{aligned}$$

We set

$$M := \frac{\alpha\eta_2}{C_M} \qquad (15)$$

where C_M is a parameter chosen in the sequel (see Section 2). Simplifying eq. (14) and taking a union bound over all rings and epochs, the desired claim in the theorem statement holds with probability $1-\log(\Lambda)|\mathcal{F}|3\xi - \frac{1}{k^{O(\mathrm{poly}(k,\log(\Lambda))}}$. Thus we set $\xi := \frac{1}{3|\mathcal{F}|\log(\Lambda)T^2}$. □

Finally, Theorem 14 gives the cost of clustering result for the output $\hat{\mathcal{Y}}$ after the offline postprocessing step is executed. Recall that the postprocessing step involves running a ρ-approximation non-DP clustering algorithm on $\hat{\mathcal{Y}}$. We present the clustering guarantee in terms of the ρ-approx non-DP clustering algorithm, the non-DP $(1+\gamma)$-coreset algorithm, the DP (κ,η_1,η_2)-coreset algorithm and parameter C_M.

Theorem 14 (Main Accuracy)**.** *Given dimension d, clustering parameter k, arbitrary parameter C_M, non-DP $(1+\gamma)$-coreset, $O(d^3)$-approximate bicriteria solution from Theorem 4, DP (κ,η_1,η_2)-semicoreset, and privacy parameter ε. Let $\mathcal{C}_{\hat{\mathcal{Y}}}$ be the set of k centers obtained from running the offline ρ-approx non-DP k-means algorithm on $\hat{\mathcal{Y}}$. Then,*

$$\begin{aligned}
&\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}},\mathcal{S}) \\
&\leq \frac{\kappa}{(1-\gamma)^3}\cdot((1+\gamma)^3\rho(\kappa+16C_M) + ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)))\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&+ \frac{\kappa}{(1-\gamma)^3}\cdot(\rho C_M(1+\gamma)^3 \\
&+ C_M(1-\gamma^2)(1-\gamma))V'(d,k,\varepsilon,T,\Lambda) \\
&+ \frac{\kappa}{(1-\gamma)^3}\cdot((1+\gamma)^2\rho + (1-\gamma)^2)V''(d,k,\varepsilon,T,\Lambda) \qquad (16)
\end{aligned}$$

where $V'(d,k,\varepsilon,T,\Lambda) = O\left(\frac{\Lambda^2k^2}{d\varepsilon}\log^2(\Lambda)\log^4(k)\log(T)\,\mathrm{poly}\log\left(T\cdot k\cdot\Lambda\right)\right)$, $V''(d,k,\varepsilon,T,\Lambda) = O(k\Lambda^2\log^2(k)\log(\Lambda)\log T)\cdot\left(\frac{\alpha\eta_2}{C_M} + \tilde{O}(\frac{1}{\varepsilon}\cdot\mathrm{poly}(\log(T\cdot k\cdot\Lambda)\right)$, and $\alpha := O(d^3)$.

Proof. We first recall that our algorithm releases the semicoreset $\hat{\mathcal{Y}}$ after running a non-DP $(1+\gamma)$-coreset algorithm on it. Thus, the guarantee of the resulting semicoreset which we call $\hat{\mathcal{Y}}_{off}$ is

$$\begin{aligned}
&\frac{(1-\gamma)^3}{\kappa}\mathsf{cost}(\mathcal{C},\mathcal{S}) - ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma))\cdot\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&- C_M(1-\gamma^2)(1-\gamma)\cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&- (1-\gamma)^2V''(d,k,\varepsilon,T,\Lambda) \\
&\leq \mathsf{cost}(\mathcal{C},\hat{\mathcal{Y}}_{off}) \\
&\leq (1+\gamma)^3\kappa\mathsf{cost}(\mathcal{C},\mathcal{S}) + 16C_M(1+\gamma)^3\mathsf{cost}(\mathcal{C}_{\mathcal{S}}^{opt},\mathcal{S}) \\
&+ C_M(1+\gamma)^3\cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1+\gamma)^2V''(d,k,\varepsilon,T,\Lambda) \qquad (17)
\end{aligned}$$

where $V'(d,k,\varepsilon,T,\Lambda) = O\left(\frac{\Lambda^2k^2}{d\varepsilon}\log^2(\Lambda)\log^4(k)\log(T)\,\mathrm{poly}\log\left(T\cdot k\cdot\Lambda\right)\right)$, $V''(d,k,\varepsilon,T,\Lambda) = O(k\Lambda^2\log^2(k)\log(\Lambda)\log T)\cdot(\frac{\alpha\eta_2}{C_M} + \tilde{O}(\frac{1}{\varepsilon}\cdot\mathrm{poly}(\log(T\cdot k\cdot\Lambda))$, and $\alpha := O(d^3)$.

Recall from the theorem statement that $\mathcal{C}_{\hat{\mathcal{Y}}}$ is the set of k centers obtained from running the offline ρ-approx non-DP k-means algorithm on $\hat{\mathcal{Y}}_{off}$ which gives us the fol-

lowing guarantee,

$$\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \hat{\mathcal{Y}}_{off}) \leq \rho \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\hat{\mathcal{Y}}}, \hat{\mathcal{Y}}_{off}) \tag{18}$$

Since eq. (17) is true for any set of k centers, in particular it holds for $\mathcal{C}_{\hat{\mathcal{Y}}}$, which gives us the following

$$\begin{aligned}
&\frac{(1-\gamma)^3}{\kappa}\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \mathcal{S}) - ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)) \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&- C_M(1-\gamma^2)(1-\gamma) \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&- (1-\gamma)^2 V''(d,k,\varepsilon,T,\Lambda) \\
&\leq \mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \hat{\mathcal{Y}}_{off}) \\
&\leq (1+\gamma)^3\kappa\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \mathcal{S}) + 16C_M(1+\gamma)^3\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1+\gamma)^3 \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1+\gamma)^2 V''(d,k,\varepsilon,T,\Lambda)
\end{aligned} \tag{19}$$

We derive the desired expression in the theorem statement by starting with the LHS of eq. (19) and then applying eq. (18) to obtain eq. (22). By observing that the cost of $\hat{\mathcal{Y}}_{off}$ wrt its optimal set of centers $\mathcal{C}^{opt}_{\hat{\mathcal{Y}}}$ is less than its cost wrt to $\mathcal{C}^{opt}_{\mathcal{S}}$, we obtain eq. (23). eq. (24) is obtained by simply applying the RHS of the coreset relation from eq. (19) wrt $\mathcal{C}^{opt}_{\mathcal{S}}$. The final statement is derived by grouping like terms and simplifying eq. (25).

$$\begin{aligned}
&\frac{(1-\gamma)^3}{\kappa}\mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \mathcal{S}) && (20)\\
&\leq \mathsf{cost}(\mathcal{C}_{\hat{\mathcal{Y}}}, \hat{\mathcal{Y}}_{off}) + ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)) \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1-\gamma^2)(1-\gamma) \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1-\gamma)^2 V''(d,k,\varepsilon,T,\Lambda) && (21)\\
&\leq \rho\mathsf{cost}(\mathcal{C}^{opt}_{\hat{\mathcal{Y}}}, \hat{\mathcal{Y}}_{off}) + ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)) \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1-\gamma^2)(1-\gamma) \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1-\gamma)^2 V''(d,k,\varepsilon,T,\Lambda) && (22)\\
&\leq \rho\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \hat{\mathcal{Y}}_{off}) + ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)) \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1-\gamma^2)(1-\gamma) \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1-\gamma)^2 V''(d,k,\varepsilon,T,\Lambda) && (23)\\
&\leq \rho((1+\gamma)^3\kappa\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ 16C_M(1+\gamma)^3\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1+\gamma)^3 \cdot V'(d,k,\varepsilon,T,\Lambda) \\
&+ (1+\gamma)^2 V''(d,k,\varepsilon,T,\Lambda)) + ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)) \cdot \mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ C_M(1-\gamma^2)(1-\gamma) \cdot V'(d,k,\varepsilon,T,\Lambda)) \\
&+ (1-\gamma)^2 V''(d,k,\varepsilon,T,\Lambda) && (24)\\
&\leq ((1+\gamma)^3\rho(\kappa + 16C_M) \\
&+ ((1-\gamma)^3\eta_1\Lambda^2 \\
&+ 16C_M(1-\gamma^2)(1-\gamma)))\mathsf{cost}(\mathcal{C}^{opt}_{\mathcal{S}}, \mathcal{S}) \\
&+ (\rho C_M(1+\gamma)^3 + C_M(1-\gamma^2)(1-\gamma))V'(d,k,\varepsilon,T,\Lambda) \\
&+ ((1+\gamma)^2\rho + (1-\gamma)^2)V''(d,k,\varepsilon,T,\Lambda) && (25)
\end{aligned}$$

□

C BICRITERIA APPROXIMATION IN CONTINUAL RELEASE SETTING

We describe our bicriteria approximation algorithm and analysis in more detail here.

Algorithm. Our bicriteria approximation algorithm is given by Algorithm 2 which initializes $\log(k)$ parallel instances of randomly shifted quadtrees. Each input point x_t is assigned to a cell in every level of every quadtree. For a fixed quadtree $1 \leq q \leq \log(k)$ and fixed level $0 \leq \ell \leq \Lambda$, the subroutine DPFindCenters (see Algorithm 3) returns a candidate set of centers $\hat{\mathcal{F}}_t$ which is

Algorithm 2: BicriteriaDPCenters

Data: Privacy parameter ε, Stream $\mathcal{S}$ of points $x_1, \ldots, x_T \in \mathbb{R}^d$
Initialize(ε):
$\varepsilon' := \frac{\varepsilon}{4\log(\Lambda)\log^2(k)}$;
Parallel quadtrees $Q_1, \ldots, Q_{\log(k)}$ such that:
each quadtree Q_q has $\log(\Lambda)$ levels with the bottom level having grid size $\Theta(1)$;
for $0 \le \ell \le \log(\Lambda)$ *and* $1 \le q \le \log(k)$ **do**
  Initialize $\mathbf{DPFindCenters}_{\ell,q}$ of $\mathbf{DPFindCenters}(\varepsilon')$;
Set of candidate centers $\mathcal{F} := \emptyset$;
Update(x_t):
for $0 \le \ell \le \log(\Lambda)$ *and* $1 \le q \le \log(k)$ **do**
  $\hat{\mathcal{F}}_t \leftarrow \mathbf{Update}(x_t)$ of $\mathbf{DPFindCenters}_{\ell,q}$; ▷ See Algorithm 3
  $\mathcal{F} \leftarrow \mathcal{F} \cup \hat{\mathcal{F}}_t$;
Output $\mathcal{F}$;

added to the current set of candidate set of centers $\mathcal{F}$.

The DPFindCenters subroutine (see Algorithm 3) finds the approximate heaviest $O(k)$ cells in a fixed level of a fixed quadtree. It achieves this by first hashing the cell containing the current point to a bucket, note that there are $w := O(k)$ many buckets. For each hash bucket $j \in [w]$, the algorithm maintains a continual release θ-heavy hitter instance DP-HH$_j$. We use the ℓ_1-heavy hitter algorithm from Epasto et al. [2023] as DP-HH — it returns a set H of θ-heavy hitters and their approximate counts $\hat{f}(\mathbf{c})$ for all $\mathbf{c} \in H$. Since we are storing the centerpoints of all the cells marked as heavy hitters as candidate centers, we need to ensure that we do not store too many false positives, i.e., cells whose counts are much smaller than θ times the length of the hashed substream. To address this challenge, we have an additional pruning step that eliminates any cell $\mathbf{c}$ whose approximate count is less than $\Theta(\theta)\hat{T}_{h(\mathbf{c})}$ where $\hat{T}_{h(\mathbf{c})}$ denotes the DP count of each hash bucket $j \in [w]$ at timestep $t \in [T]$. We keep track of $\hat{T}_j$ via an instance of the Binary Mechanism Dwork et al. [2010] denoted as $\mathbf{BM}_j$ for each $j \in [w]$. Finally, only the centerpoints of cells that pass this pruning step are added as candidate centers to the set $\hat{\mathcal{F}}_t$.

Algorithm 3: DPFindCenters

Data: Privacy parameter ε', Stream of points $x_1, \ldots, x_T$, level of quadtree ℓ
Initialize(ε'):
$\varepsilon' \leftarrow \varepsilon'$; ▷ where $\varepsilon' := \frac{\varepsilon}{4\log(\Lambda)\log^2(k)}$
$w = O(k)$;
Hash function $h : [2^\ell] \to [w]$ s.t. $\forall$ cells $\mathbf{c}, \forall j \in [w], \Pr[h(\mathbf{c}) = j] = \frac{1}{w}$;
$\hat{T}_1 = 0, \ldots, \hat{T}_w = 0$; ▷ DP Count for the size of hash bucket
Initialize $\mathbf{BM}_1, \ldots, \mathbf{BM}_w$ of **BinaryMechanism**(T, ε'); ▷ See Theorem 7 Dwork et al. [2010]
Initialize **DP-HH**$_1, \ldots,$ **DP-HH**$_w$ of **DP-HH**(T, ε'); ▷ See Theorem 8 Epasto et al. [2023]
Update(x_t):
Initialize $\hat{\mathcal{F}}_t = \emptyset$;
Let x_t be mapped to cell $\mathbf{c}^*$; ▷ **DPFindCenters** is initialized per level ℓ of quadtree instance q
for $p = 1, \ldots, L$, *where* $L := \log(k^2)$ *run in parallel* **do**
  for $j \in [w]$ **do** ▷ Update the DP count of each hash bucket
    if $j = h(\mathbf{c}^*)$ **then**
      $\hat{T}_j \leftarrow$ **Update**(1) of $\mathbf{BM}_j$;
    else
      $\hat{T}_j \leftarrow$ **Update**(0) of $\mathbf{BM}_j$
  for $j \in [w]$ **do** ▷ Update the DP HHs of each hash bucket
    if $h(\mathbf{c}^*) = j$ **then**
      $\hat{f}, H \leftarrow$ **Update**$(\mathbf{c}^*)$ of **DP-HH**$_j$;
    else
      $\hat{f}, H \leftarrow$ **Update**$(\perp)$ of **DP-HH**$_j$;
    for *cell* $\mathbf{c} \in H$ **do**
      if $\hat{f}(\mathbf{c}) \ge \frac{\theta}{1000} \cdot \hat{T}_{h(\mathbf{c})}$ **then**
        Add centerpoint of $\mathbf{c}$ to $\hat{\mathcal{F}}_t$ as a center; ▷ θ is the HH threshold parameter and set to be an appropriate constant
Return $\hat{\mathcal{F}}_t$;

C.1 Proof of Theorem 4

Lemma 6 (Privacy of Theorem 4)**.** *BicriteriaDPCenters is ε-DP under the continual release setting.*

Proof. For a fixed timestep t, an input point x_t is passed to an instance of DPFindCenters (Algorithm 3) where it is assigned to a specific cell for a specific level of the quadtree, and cells at the same level are disjoint. The cell containing x_t is then hashed to a bucket which maintains a DP-HH instance.

There are $\log(\Lambda)$ levels per quadtree, thus point x_t is a

member of $\log(\Lambda)$ cells in total. As there are $2\log^2(k)$ parallel processes — considering $\log(k)$ quadtrees and $\log(k^2)$ parallel processes per quadtree — a single point participates in $2\log\Lambda\log^2(k)$ total calls to DP-HH. Since DP-HH is DP under the continual release setting (by Theorem 8), assigning each DP-HH instance a privacy budget of $\frac{\varepsilon}{4\log(\Lambda)\log^2(k)}$ preserves $(\varepsilon/2)$-DP by sequential composition (Item 1 of Theorem 6).

Next, the algorithm uses the Binary Mechanism (which we know is DP by Theorem 7) with a privacy budget of $\frac{\varepsilon}{4\log(\Lambda)\log^2(k)}$ to keep track of the size of each hash substream $\hat{T}_j$ $\forall j \in [w]$. Since the input cells (and corresponding points within cells) are disjoint in each substream due to hashing, this preserves $\frac{\varepsilon}{4\log(\Lambda)\log^2(k)}$-DP by parallel composition (Item 2 of Theorem 6) which over $2\log(\Lambda)\log^2(k)$ parallel processes preserves $(\varepsilon/2)$-DP by sequential composition (Item 1 of Theorem 6).

Finally, since only the centerpoints of cells identified as heavy hitters by DP-HH are released, these centers maintain ε-DP by sequential composition (Item 1 of Theorem 6).

□

Lemma 7 (Accuracy of Theorem 4)**.** *With probability at least* $1 - 1/k^{O(\mathrm{poly}(k,\log(\Lambda))}$,

$$cost(\mathcal{F},\mathcal{S}) \le O(d^3) cost(C_{\mathcal{S}}^{opt},\mathcal{S}) + \tilde{O}\left(\frac{d^2\Lambda^2 k}{\varepsilon}\,\mathrm{poly}\left(\log\left(T\cdot k\cdot\Lambda\right)\right)\right)$$

where $cost(C_{\mathcal{S}}^{opt},\mathcal{S})$ *is the optimal* k*-means cost for* $\mathcal{S}$*.*

Proof. We first state some geometric properties regarding the cells within the quadtree construction.

Proposition 1. *Cohen-Addad et al. [2022b] Let* $\mathcal{B}$ *be an* ℓ_∞ *ball of radius* r *contained in* $[-\Lambda,\Lambda]^d$ *(it forms a* d*-dimensional cube with each side length* $2r$*). Then for a randomly shifted quadtree and any level* ℓ *with grid size at least* $r' \ge 2r$*,* $\mathcal{B}$ *is split by the grid in each dimension* $j \in [d]$ *independently with probability* $\frac{2r}{r'}$*.*

Let $C_{\mathcal{S}}^{opt} = \{c_1,\ldots,c_k\}$ be the optimal set of k centers for the input set $\mathcal{S} = \{x_1,\ldots,x_T\}$. For any radius, define n_r as the number of points $x \in \mathcal{S}$ such that $d(x, C_{\mathcal{S}}^{opt}) \ge r$. Note that the opt cost of k-means (and k-median) is given by $\sum_{p\in\mathbb{Z}} 2^{2p}\cdot n_{2^p}$ and $\sum_{p\in\mathbb{Z}} 2^{p}\cdot n_{2^p}$ (up to an $O(1)$-approximation).

Fix some radius $r = 2^p$ where $p \in \mathbb{Z}$ and consider a randomly shifted grid of size $20rd$. The following lemma characterizes cells containing $\cup_{i=1}^k \mathcal{B}(c_i,r)$ with respect to the grid size.

Lemma 8. *Cohen-Addad et al. [2022b]* $\cup_{i=1}^k \mathcal{B}(c_i,r)$ *is contained in at most* $4k$ *cells of grid length* $20rd$ *by the corresponding level of the quadtree with probability at least 1/2.*

Let G_ℓ where $0 \le \ell \le \log(\Lambda)$ be the set of $4k$ good cells of length $20rd$ (equivalently ℓ_2-radius of $10rd^{3/2}$) at level ℓ. Let the number of points in $\mathcal{S}$ uncovered by G_ℓ be n_{G_ℓ}. Observe that by Lemma 8, since G_ℓ contains $\cup_{i=1}^k \mathcal{B}(c_i,r)$ with probability at least 1/2, we have that $n_{G_\ell} \le n_r$. It follows that

$$\sum_{\ell=0}^{\log(\Lambda)} (\text{grid length at level } \ell)^2 \cdot n_{G_\ell} \le O(d^3) \sum_{p\in\mathbb{Z}\,:\,r=2^p\le\Lambda} r^2\cdot n_r \le O(d^3)\cdot cost(C_{\mathcal{S}}^{opt},\mathcal{S}) \tag{26}$$

Observe that we can define a one-one mapping between the level ℓ and the radius r, i.e., the radius r (ranging from 1 to Λ) maps to the grid length of a cell which is at most $\Lambda/2^\ell$ (level ℓ ranges from $\log(\Lambda)$ to 0). Since the grid length of a cell in G_ℓ at level ℓ is $20rd$ which maps to $20d\frac{\Lambda}{2^\ell}$, we can replace the leftmost term in the expression above as follows

$$O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 n_{G_\ell} \le O(d^3) \sum_{p\in\mathbb{Z}\,:\,r=2^p\le\Lambda} r^2\cdot n_r \le O(d^3)\cdot cost(C_{\mathcal{S}}^{opt},\mathcal{S}) \tag{27}$$

Recall that we define $\mathcal{F}_t$ as the set of centers until time step t. For a fixed level ℓ, let the set of cells the algorithm DP-HH marks as heavy at timestep t at level ℓ as $H_{\ell,t}$. Note that although there is an extra pruning step in DPFindCenters after the cells are marked heavy by DP-HH, we do not account for this here — as if a cell is an α-HH and marked heavy by DP-HH, and it survives the pruning step, it will still be an α-HH. Then,

$$cost(\mathcal{F}_t) \le O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot \mathbb{1}[x_t \text{ uncovered by } H_{\ell,t}]$$

Observe that

$$\begin{aligned}
&cost(\mathcal{F}) \\
&= \sum_{t=1}^{T} cost(\mathcal{F}_t) \\
&\leq O(d^2) \sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot \sum_{t=1}^{T} \mathbb{1}[x_t \text{ uncovered by } H_{\ell,t}]
\end{aligned} \tag{28}$$

Lemma 9. *For a fixed level ℓ, with probability at least $1 - \frac{12}{k}$,*

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{1}[x_t \textit{ uncovered by } H_{\ell,t}] \\
&\leq (1+\theta) n_{G_\ell} \\
&+ \frac{4k \log^2 \Lambda \log k}{\varepsilon \eta} \operatorname{poly}\left(\log\left(\frac{T \cdot 2^\ell}{\theta \xi \gamma_h}\right)\right)
\end{aligned} \tag{29}$$

Proof. Observe that

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{1}[x_t \text{ uncovered by } H_{\ell,t}] \\
&= \sum_{t=1}^{T} (\mathbb{1}[(x_t \text{ uncovered by } H_{\ell,t}) \wedge (x_t \text{ uncovered by } G_\ell)] \\
&+ \mathbb{1}[(x_t \text{ uncovered by } H_{\ell,t}) \wedge (x_t \text{ covered by } G_\ell)]) \\
&= \sum_{t=1}^{T} \mathbb{1}[(x_t \text{ uncovered by } H_{\ell,t}) \wedge (x_t \text{ uncovered by } G_\ell)] \\
&+ \sum_{t=1}^{T} \mathbb{1}[(x_t \text{ uncovered by } H_{\ell,t}) \wedge (x_t \text{ covered by } G_\ell)]
\end{aligned}$$

The first sum in the above expression can be upper bounded by n_{G_ℓ}, thus it remains to bound the second sum. In order to bound the second sum, we will need some properties of good cells that are hashed to buckets in DPFindCenters, the proofs of these claims can be found in Appendix E. In the sequel, we denote $N_{\ell,\mathbf{c}}$ as the number of points in the cell $\mathbf{c}$ at level ℓ and the hash buckets as $\mathcal{B}_j$ where $j \in [w]$. For simplicity, we consider the number of hash buckets $w := 40k$. We first show that for any good cell $\mathbf{c}$, it is unlikely that the bucket it is hashed to contains another good cell $\mathbf{c}' \neq \mathbf{c}$.

Proposition 2. *Let $\mathbf{c} \in G_\ell$, then with probability at least $1/2$, for any $\mathbf{c}' \in G_\ell$ such that $\mathbf{c}' \neq \mathbf{c}$, we have that $h(\mathbf{c}') \neq h(\mathbf{c})$.*

In the next claim we give a bound on the size of the hash bucket denoted as $\mathcal{B}_j$ where $j \in [w]$ in terms of the size of a good cell that is hashed to it and n_r.

Proposition 3. *For each $\mathbf{c} \in G_\ell$, suppose the hash bucket $\mathcal{B}_j$ where $j \in [w]$, contains only one good cell which is $\mathbf{c}$. Let $N_{\ell,\mathbf{c}} := y$. Then with probability at least 1/2, $|\mathcal{B}_j| \leq 2(y + \frac{n_{G_\ell}}{40k})$.*

Note that since the hashing procedure is run $\log(k^2)$ times in parallel, we can boost the success probabilities in the above claims to be $1 - 1/k^2$.

Observe that for a fixed hash bucket $\mathcal{B}_j$, any cell $\mathbf{c}$ such that $N_{\ell,\mathbf{c}} \geq \theta \cdot 2(y + \frac{n_{G_\ell}}{40k})$ qualifies as a θ-heavy hitter since $N_{\ell,\mathbf{c}} \geq \theta \cdot 2(y + \frac{n_{G_\ell}}{40k}) \geq \theta|\mathcal{B}_j|$ (by Proposition 3). In particular, for good cell $\mathbf{c}_y$ such that $N_{\ell,\mathbf{c}_y} = y$, if $\mathbf{c}_y$ is an θ-HH then $y \geq \theta \cdot 2(y + \frac{n_{G_\ell}}{40k})$ or $y \geq \frac{\theta n_{G_\ell}}{20k}$. We formalize this intuition in the claim below where we use the accuracy guarantees of DP-HH given by Theorem 8 to characterize the good cells that are reported as θ-HHs.

Proposition 4. *Let $\mathbf{c} \in G_\ell$. If $N_{\ell,\mathbf{c}} \geq \frac{\theta n_{G_\ell}}{20k}$, and $N_{\ell,\mathbf{c}} \geq \frac{2\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h} \operatorname{poly}(\log(\frac{T \cdot k \cdot 2^\ell}{\theta\gamma_h}))$, then with probability at least $1 - \frac{12}{k}$, $\mathbf{c}$ is reported as an θ-heavy hitter by DP-HH.*

Finally, we give an upper bound for the number of points that are covered by good cells but for which DP-HH fails to report as heavy.

Proposition 5. *With probability $1 - \frac{12}{k}$,*

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{1}[(x_t \textit{ uncovered by } H_{\ell,t}) \wedge (x_t \textit{ covered by } G_\ell)] \\
&\leq \theta n_{G_\ell} + \frac{8k\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h} \operatorname{poly}\left(\log\left(\frac{T \cdot k \cdot 2^\ell}{\theta\gamma_h}\right)\right)
\end{aligned}$$

Thus by combining Proposition 5 with our observation about the first sum being upper bounded by n_{G_ℓ} in the decomposition of $\sum_{t=1}^{T} \mathbb{1}[x_t \text{ uncovered by } H_{\ell,t}]$, we obtain our desired statement in Lemma 9. □

Note that we have shown Lemma 9 is true with probability at least $1 - \frac{12}{k}$, for a fixed level. Since we have $\log(\Lambda)$ many levels in a specific quadtree, and $\log(k)$ many quadtree instances in parallel — we can boost our probability of success to be sufficiently high. It remains to bound the total k-means cost for the set of centers $\mathcal{F}$ output by our algorithm. Combining eq. (26), eq. (27) and eq. (28) along with Lemma 9, we obtain the following.

$$
\begin{aligned}
&cost(\mathcal{F},\mathcal{S})\\
&= \sum_{t=1}^{T} cost(\mathcal{F}_t,\mathcal{S})\\
&\le O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot \sum_{t=1}^{T} \mathbb{1}[x_t \text{ uncovered by } H_{\ell,t}]\\
&\le O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot ((1+\theta)n_{G_\ell}\\
&+ \frac{8k\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h}\operatorname{poly}\big(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\gamma_h})\big))\\
&= O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot (1+\theta)n_{G_\ell}\\
&+ O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot \frac{8k\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h}\cdot\\
&\operatorname{poly}\big(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\gamma_h})\big)\\
&\le O(d^3)(1+\theta)\sum_{p\in\mathbb{Z}\,:\,r=2^p\le\Lambda} r^2\cdot n_r\\
&+ O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot \frac{8k\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h}\cdot\\
&\operatorname{poly}\big(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\gamma_h})\big)\\
&\le O(d^3)(1+\theta)cost(C_{\mathcal{S}}^{opt},\mathcal{S})\\
&+ O(d^2)\sum_{\ell=0}^{\log(\Lambda)} (\Lambda/2^\ell)^2 \cdot \frac{8k\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h}\cdot\\
&\operatorname{poly}\big(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\gamma_h})\big)\\
&= O(d^3)(1+\theta)cost(C_{\mathcal{S}}^{opt},\mathcal{S})\\
&+ O\left(d^2\Lambda^2\frac{k\log(\Lambda)\log^2(k)}{\varepsilon\gamma_h}\operatorname{poly}\big(\log(\frac{T\cdot k\cdot\Lambda}{\theta\gamma_h})\big)\right)
\end{aligned}
$$

Finally, we can set θ (threshold for HHs) and γ_h (approximation factor for frequency of a cell marked as heavy from Theorem 8) to appropriate constants. The accuracy claim follows. □

Lemma 10 (Space of Theorem 4)**.** *BicriteriaDPCenters uses $O(k\log(\Lambda)\log^2(k)\operatorname{poly}(\log(T\Lambda k)))$ space.*

Proof. We analyze the total space usage for DP-HH in Algorithm 3 as this dominates space usage for the entire algorithm. From Theorem 8, one instance of DP-HH uses $\operatorname{poly}(\log(T\Lambda k))$. Since we run DP-HH on $O(k)$ many hash substreams, and $2\log(\Lambda)\log^2(k)$ parallel processes, the total space is $O(k\log(\Lambda)\log^2(k)\operatorname{poly}(\log(T\Lambda k)))$. □

Lemma 11 (Size of $\mathcal{F}$ in Theorem 4)**.** *For all $j \in [w]$, suppose the size of the hash bucket $\mathcal{B}_j$ is $|\mathcal{B}_j| = \Omega(\frac{\log(\Lambda)\log^3(k)}{\varepsilon}\log^{2.5}T)$ then with high probability, the total number of heavy hitters at the end of the stream is $O(k\log^2(k)\log(\Lambda)\log T)$.*

In other words, $\mathcal{F}$ has atmost $O(k\log^2(k)\log(\Lambda)\log T)$ centers.

Proof. The algorithm runs independent instances of the DP-HH algorithm for each bucket of each level in each instantiation of the quadtree, thus it is sufficient to first show that for a fixed quadtree Q, a fixed level ℓ, and a fixed bucket $\mathcal{B}_j$ where $j \in [w]$, the total number of heavy hitters is at most $O(\frac{(1+\gamma_h)}{\theta}\log T)$.

Let the timestamps of points that end up in $\mathcal{B}_j$ be $t_i = 2^i$, where $0 \le i \le \log(T)$. Let the state of the hash bucket at time step t be $\mathcal{B}_j^{(t)}$. We set the failure probability in Theorem 7 as $\xi := \frac{1}{k^2}$. From Theorem 7 we know that with probability $1-\frac{1}{k^2}$, the DP count of the hash bucket $\hat{T}_j$ at timestep t has additive error $O(\frac{\log(\Lambda)\log^3(k)}{\varepsilon}\log^{2.5}(T))$. Thus for a fixed timestamp t, if $|\mathcal{B}_j^{(t)}| = \Omega(\frac{\log(\Lambda)\log^3(k)}{\varepsilon}\log^{2.5}(T))$, then we can see that a cell $\mathbf{c}$ is added to $\mathcal{F}_t$ only if $\hat{f}(\mathbf{c}) > \frac{2\theta}{1000}|\mathcal{B}_j^{(t)}|$. Recall from Condition 1 of Theorem 8 that $\hat{N}_{\ell,\mathbf{c}} \in (1\pm\gamma_h)N_{\ell,\mathbf{c}}$. Thus if $\mathbf{c} \in \mathcal{F}_t$ and $|\mathcal{B}_j^{(t)}| = \Omega(\frac{\log(\Lambda)\log^3(k)}{\varepsilon}\log^{2.5}(T))$ then it must be the case that with probability $1-\frac{1}{k^2}$, $N_{\ell,\mathbf{c}} \ge \frac{2\theta}{1000(1+\gamma_h)}\|\mathcal{B}_j^{(i)}\|_1 \ge \frac{2\theta}{1000(1+\gamma_h)}t_{i-1}$.

Now, suppose for a contradiction, that the number of heavy hitters between t_{i-1} and t_i is at least $\frac{1000(1+\gamma_h)}{\theta}$. Then for each such cell $\mathbf{c}$, we have that $N_{\ell,\mathbf{c}} \ge \frac{2\theta}{1000(1+\gamma_h)}t_{i-2}$. Since there are at least $\frac{2000(1+\gamma_h)}{2\theta}$ such cells, this implies that the total number of points between t_{i-1} and t_i is $\ge \frac{2\theta}{1000(1+\gamma_h)}t_{i-2}\frac{2000(1+\gamma_h)}{2\theta} = 2^i = t_i$, which is a contradiction. Thus there must be at most $\frac{1000(1+\gamma_h)}{\theta}$ cells marked as heavy hitters between consecutive intervals, and since there are $\log(T)$ such intervals, we have that the total number of ℓ_1 heavy hitters for a fixed bucket is $O(\frac{(1+\gamma_h)}{\theta}\log T)$.

Boosting the success probability over $O(k)$ buckets, $\log(k^2)$ parallel processes, $\log(\Lambda)$ quadtree levels, and $\log(k)$ parallel processes of the quadtree instantiation, accounting for the additional number of heavy hitters, and taking θ and γ_h as appropriate constants, we obtain the claim as stated. □

D DP MERGE AND REDUCE ALGORITHM

We give a differentially-private variant of the widely-known Merge and Reduce framework Har-Peled and Mazumdar [2004], Agarwal et al. [2004], Feldman et al. [2020] that is used to efficiently release a coreset for a stream of points. The main idea behind the Merge and Reduce technique is to partition the input stream into blocks, compute a coreset for each block, take the union of the resulting coresets (merge step), and compute the coreset of the union (reduce step). The merging and reducing of the coresets is done in a tree-like fashion. In order to reduce the error introduced by merging, the number of levels in the tree must be small. On a high-level, our framework computes coresets at the base level (of the tree) using a DP semicoreset Algorithm $\mathcal{A}$ (e.g. Stemmer and Kaplan [2018], Ghazi et al. [2020]) and then computes coresets for subsequent levels using a non-DP coreset Algorithm $\mathcal{B}$ (e.g. Cohen-Addad et al. [2021]).

Recall that our main algorithm runs multiple concurrent instances of DP-Merge-Reduce at any given timestep, so N in the theorem statement below represents the number of points seen by a specific instance of DP-Merge-Reduce. We use a non-DP coreset algorithm in DP-Merge-Reduce which we fix in the analysis to be the current state-of-the-art clustering result byCohen-Addad et al. [2021] (see formal guarantees in Theorem 10).

First, we show that the semicoreset definition (i.e., Definition 2) satisfies the Merge and Reduce properties, i.e., the union of semicoresets is a semicoreset and the coreset of a union of semicoresets is a valid semicoreset for the underlying points.

Lemma 12. *1. (Merge) If Q is a $(1+\gamma, \eta_1, \eta_2)$-semicoreset of P, Q' is a $(1+\gamma, \eta_1, \eta_2)$-semicoreset of P' and P, P' are disjoint, then $Q \cup Q'$ is a $(1+\gamma, \eta_1, 2\eta_2)$-semicoreset of $P \cup P'$.*

2. (Reduce) If R is a $(1+\gamma)$-coreset of $Q \cup Q'$, then R is a $((1+\gamma)^2, (1-\gamma)\eta_1, (1+\gamma)2\eta_2)$-semicoreset of P.

Proof. We first prove the merge property. For any set of k-centers $C \subseteq \mathbb{R}^d$,

$$\begin{aligned}\mathsf{cost}(C, Q \cup Q') &\leq \mathsf{cost}(C, Q) + \mathsf{cost}(C, Q') \\ &\leq (1+\gamma)\mathsf{cost}(C, P) + \eta_2 \\ &+ (1+\gamma)\mathsf{cost}(C, P') + \eta_2 \\ &\leq (1+\gamma)\mathsf{cost}(C, P \cup P') + 2\eta_2\end{aligned}$$

Algorithm 4: Algorithm DP-Merge-Reduce

Data: Point x'_t to be added to coreset, DP parameters $\varepsilon, \varepsilon_1, \delta_1$
Initialize$(\varepsilon, \varepsilon_1, \delta_1)$:
Block size M;
▷ Value of M depends on our choice of DP clustering algorithm. See Theorem 2 and Theorem 3 for exact values
$P_0 = \emptyset$;
Size of P_0 as $p_0 = 0$;
▷ We only store P_0 in the actual algorithm, the rest of the P_i's are only used in the accuracy analysis, so we do not run the following line in the actual algorithm. N_r denotes the number of points in ring R_r.
$P_i = \emptyset$ for all $i = 1, \ldots, u$ where $u = \lfloor \log(2N_r/M) \rfloor + 1$;
$\gamma_i := \gamma/Ci^2$ for $i = 1, \ldots, u-1$;
Update(x'_t):
if $x'_t \neq \perp$ **then**
  Insert x'_t into P_0;
  $p_0 = p_0 + 1$ ▷ update the size of P_0;
  if ***LevelZero-AboveThreshold***$(p_0, M, \varepsilon) \to \top$ **then**
    ▷ See Algorithm 5
    $Q_1 \leftarrow \mathcal{A}(P_0, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2)$ ▷ $\mathcal{A}$ computes a $(\varepsilon_1, \delta_1)$-DP (κ, η_1, η_2)-semicoreset;
    $P_1 = P_1 \cup P_0$;
    Delete elements in P_0, and set $p_0 = 0$;
    $i = 1$;
    while $\exists\, Q'_i$ *such that Q_i and Q'_i are both at level i* **do**
      ▷ **Merge-Reduce** computes $(1+\gamma_i)$-coreset Q_{i+1} of $Q_i \cup Q'_i$ and deletes Q_i, Q'_i
      $Q_{i+1} =$ **Merge-Reduce**$(\mathcal{B}, Q_i, Q'_i, \gamma_i)$;
      ▷ $\mathcal{B}$ computes a non-DP $(1+\gamma_i)$-coreset, e.g., Cohen-Addad et al. [2021]
      $P_{i+1} = P_{i+1} \cup P_i$;
      ▷ Current line and the one above are for analysis purpose only
      Delete elements in P_i;
      $i = i + 1$;
    return **Merge-Reduce**$(\mathcal{B}, \{Q_i\}_{i \leq u}, \gamma/3)$;
  else
    return $\emptyset$;
else
  return $\emptyset$;

$$\begin{aligned}\mathsf{cost}(C, Q \cup Q') &\geq \mathsf{cost}(C, Q) + \mathsf{cost}(C, Q') \\ &\geq (1-\gamma)\mathsf{cost}(C, P) \\ &- \eta_1 \mathsf{cost}(C_P^{opt}, P) - \eta_2 \\ &+ (1-\gamma)\mathsf{cost}(C, P') \\ &- \eta_1 \mathsf{cost}(C_{P'}^{opt}, P') - \eta_2\end{aligned}$$

Algorithm 5: Algorithm LevelZero-AboveThreshold

Data: Size of set P_0 denoted as p_0, Block size M, Privacy parameter ε

Result: $\top$ if the size of set P_0 exceeds noisy threshold and $\perp$ otherwise

$\hat{M} = M + \mathsf{Lap}(2/\varepsilon)$;

$\nu = \mathsf{Lap}(4/\varepsilon)$;

if $p_0 + \nu \geq \hat{M}$ **then**

  return $\top$;

else

  return $\perp$;

Next we prove the Reduce property below. For any set of k-centers $C \subseteq \mathbb{R}^d$,

$$\begin{aligned}\mathsf{cost}(C, R) &\leq (1+\gamma)\mathsf{cost}(C, Q \cup Q') \\ &\leq (1+\gamma)((1+\gamma)\mathsf{cost}(C, P \cup P') + 2\eta_2) \\ &\leq (1+\gamma)^2\mathsf{cost}(C, P \cup P') + (1+\gamma)2\eta_2\end{aligned}$$

$$\begin{aligned}\mathsf{cost}(C, R) &\geq (1-\gamma)\mathsf{cost}(C, Q \cup Q') \\ &\geq (1-\gamma)((1-\gamma)\mathsf{cost}(C, P \cup P') \\ &- \eta_1\mathsf{cost}(C_P^{opt}, P \cup P') - 2\eta_2) \\ &\geq (1-\gamma)^2\mathsf{cost}(C, P \cup P') \\ &- (1-\gamma)\eta_1\mathsf{cost}(C_P^{opt}, P \cup P') - (1-\gamma)2\eta_2\end{aligned}$$

□

Observe that at any timestep the total set of input points seen so far (denoted as P) is partitioned into subsets $P_0, \ldots, P_u$ where some P_i's can be empty and $u = \lfloor \log(2N/M) \rfloor + 1$. Note that we simulate this step of partitioning P into P_i's in Algorithm 4 solely for the analysis. It is not necessary to store $P_1, \ldots, P_u$ explicitly in the actual algorithm.

We first prove some claims about LevelZero-AboveThreshold.

Lemma 13. *[Privacy] LevelZero-AboveThreshold is ε-DP under the continual release setting.*

Proof. Recall that LevelZero-AboveThreshold checks whether the size of the set P_0 is above a certain threshold. In other words, it checks whether the total count of the *group* of elements $x_i \in P_0$ is above the given threshold. Once a positive response is returned by LevelZero-AboveThreshold, elements in P_0 are deleted and the process is repeated with a new group of elements. This algorithm is equivalent to grouping a stream of counts in Epasto et al. [2023]. In particular, the proof of privacy is identical except that in our algorithm we do not release the actual noisy counts but instead we just release a positive/negative response to the same query. □

Lemma 14 (Accuracy of LevelZero-AboveThreshold)**.** *For all $t \in [T]$, with probability $1 - \xi$, we have that*

1. $|\nu| < \frac{4}{\varepsilon}\log(\frac{2T}{\xi})$
2. $|\hat{M} - M| < \frac{2}{\varepsilon}\log(\frac{2T}{\xi})$

Proof. This follows from standard application of tail bounds for Laplace distribution and union bound over all $t \in [T]$. □

Lemma 15. *If $p_0 + \nu \geq \hat{M}$ then with probability $1 - \xi$, we have that $p_0 \geq M/2$.*

Proof. We can simplify $p_0 + \nu \geq \hat{M}$ by applying the noise bounds from Lemma 14. Thus with probability at least $1 - \xi$, we have that $p_0 \geq M - \frac{6}{\varepsilon}\log(\frac{2T}{\xi})$. Finally using the assumption about $M > \frac{12}{\varepsilon}\log(\frac{2T}{\xi})$, the statement follows. □

Lemma 16. *The number of levels is given by $u = \lceil \log(2N/M) \rceil + 1$.*

Proof. By Lemma 15, $p_0 \geq M/2$, thus the total number of blocks at level 0 is $\leq \frac{2N}{M}$. The statement follows. □

D.1 Proof of Theorem 5

Lemma 17 (Privacy of Theorem 5)**.** *DP-Merge-Reduce framework is $(\varepsilon_1 + \varepsilon, \delta_1)$-DP under the continual release setting.*

Proof. First, by Lemma 13, LevelZero-AboveThreshold is ε-DP under the continual release setting. Since the semicoresets computed by $\mathcal{A}$ at level 1 are $(\varepsilon_1, \delta_1)$-DP, we can release these DP semicoresets as they are computed. Subsequent computations on these DP semicoresets via non-DP algorithm $\mathcal{B}$ preserve DP by postprocessing. Thus by sequential composition (Item 1 of Theorem 6) the claim follows. □

Lemma 18 (Accuracy of Theorem 5)**.** *With probability at least $1 - \xi - \xi_A - \xi_B$, DP-Merge-Reduce framework releases a $((1+\gamma)\kappa, (1-\gamma)\eta_1, (\frac{4N}{M} - 1)(1+\gamma)\eta_2 + \tilde{M})$-semicoreset of P. Where $\tilde{M} := M + \frac{6}{\varepsilon}\log(\frac{2T}{\xi})$.*

Proof. Recall that P is partitioned by $P_1, \ldots, P_u$. We first prove the following claim about the coreset for a non-empty subset $P_r \subseteq P$.

Lemma 19. *Suppose P_r is non-empty. Then Q_r is a $((1+\gamma/3)\kappa, (1-\gamma/3)\eta_1, (1+\gamma/3)2^{r-1}\eta_2)$-semicoreset of P_r.*

Proof. We will first prove the claim that Q_r is a $(\prod_{j=1}^{r-1}(1+\gamma_j)\kappa, \prod_{j=1}^{r-1}(1-\gamma_j)\eta_1, \prod_{j=1}^{r-1}(1+\gamma_j)2^{r-1}\eta_2)$-semicoreset for P_r where $r \geq 2$ by induction. Note that $Q_0 = P_0$ for $p_0 + \nu < \hat{M}$. For $p_0 + \nu \geq \hat{M}$, $P_1 = P_1 \cup P_0$. Since we apply DP semicoreset algorithm $\mathcal{A}$ to P_1, the resulting coreset Q_1 is a (κ, η_1, η_2)-semicoreset for P_1.

Base Case. By Lemma 12, $Q_1 \cup Q_1'$ is a $(\kappa, \eta_1, 2\eta_2)$-semicoreset for $P_1 \cup P_1'$ and Q_2 is a $(\kappa(1+\gamma_1), (1+\gamma_1)\eta_1, (1+\gamma_1)2\eta_2)$-semicoreset of $Q_1 \cup Q_1'$. Here we use the notation Q_i' or P_i' to differentiate between sets or coresets at the same level i.

Inductive Hypothesis. Suppose the claim is true for $r = i$, i.e., Q_i is a $(\kappa\prod_{j=1}^{i-1}(1+\gamma_j), \prod_{j=1}^{i-1}(1-\gamma_j)\eta_1, \prod_{j=1}^{i-1}(1+\gamma_j)2^{i-1}\eta_2)$-semicoreset for P_i and Q_i' is a $(\kappa\prod_{j=1}^{i-1}(1+\gamma_j), \prod_{j=1}^{i-1}(1-\gamma_j)\eta_1, \prod_{j=1}^{i-1}(1+\gamma_j)2^{i-1}\eta_2)$-semicoreset for P_i'.

By Lemma 12, the Merge step implies

$$\mathsf{cost}(C, Q_i \cup Q_i') \leq \kappa \prod_{j=1}^{i-1}(1+\gamma_j)\mathsf{cost}(C, P_i \cup P_i') + \prod_{j=1}^{i-1}(1+\gamma_j)2^i\eta_2 \tag{30}$$

$$\mathsf{cost}(C, Q_i \cup Q_i') \geq \kappa \prod_{j=1}^{i-1}(1-\gamma_j)\mathsf{cost}(C, P_i \cup P_i') - \prod_{j=1}^{i-1}(1-\gamma_j)\eta_1\mathsf{cost}(C^{opt}_{P_i \cup P_i'}, P_i \cup P_i') - \prod_{j=1}^{i-1}(1-\gamma_j)2^i\eta_2 \tag{31}$$

The Reduce step implies that the resulting $(1+\gamma_{i+1})$-coreset of $Q_i \cup Q_i'$ denoted as Q_{i+1} is such that

$$\mathsf{cost}(Q_{i+1}) \leq \kappa \prod_{j=1}^{i}(1+\gamma_j)\mathsf{cost}(P_{i+1}) + \prod_{j=1}^{i}(1+\gamma_j)2^i\eta_2 \tag{32}$$

$$\mathsf{cost}(Q_{i+1}) \geq \kappa \prod_{j=1}^{i}(1-\gamma_j)\mathsf{cost}(P_{i+1}) - \prod_{j=1}^{i-1}(1-\gamma_j)\eta_1\mathsf{cost}(C^{opt}_{P_{i+1}}, P_{i+1}) - \prod_{j=1}^{i}(1-\gamma_j)2^i\eta_2 \tag{33}$$

Finally, provided c is large enough, we have:

$$\prod_{j=1}^{i}(1+\gamma_j) \leq \prod_{j=1}^{i}\exp(\frac{\gamma}{cj^2}) = \exp(\frac{\gamma}{c}\sum_{j=1}^{i}\frac{1}{j^2}) \leq \exp(\frac{\gamma}{c}\cdot\frac{\pi^2}{6}) \leq 1+\gamma/3 \tag{34}$$

$$\prod_{j=1}^{i}(1-\gamma_j) \leq \prod_{j=1}^{i}\exp(-\frac{\gamma}{cj^2}) = \exp(-\frac{\gamma}{c}\sum_{j=1}^{i}\frac{1}{j^2}) \leq \exp(-\frac{\gamma}{c}\cdot\frac{\pi^2}{6}) \leq 1-\gamma/3 \tag{35}$$

The statement follows.

□

Finally, we release a $(1+\gamma/3)$-coreset of $\cup_{i \leq u} Q_i$ of $\cup_{i \leq u} P_i$ for all non-empty P_i which by similar arguments as above is a $((1+\gamma)\kappa, (1-\gamma)\eta_1, (1+\gamma)\eta_2(2^u-1))$-semicoreset of P. The statement follows by plugging in the value for the total number of levels u.

Note that if $p_0 + \nu < \hat{M}$ then we do not release anything. So we have to account for an additional additive error of $M + \frac{6}{\varepsilon}\log(\frac{2T}{\xi})$ w.p. $1-\xi$ in this case. If $p_0 + \nu \geq \hat{M}$, then we proceed by computing a DP coreset using Algorithm $\mathcal{A}$ with failure probability ξ_A. We also compute all the coresets past the first level using Algorithm $\mathcal{B}$ and a failure probability of $\xi_B/2u$, where u is the number of levels. Thus for a fixed run of DP-Merge-Reduce, by a union bound, the total failure probability for this part is at most ξ_B. □

Lemma 20 (Space of Theorem 5)**.** *DP-Merge-Reduce framework uses $S_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_A) + S_{\mathcal{B}}(SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2, \xi_A), k, d, \gamma, \xi_B) + \lceil\log(2N/M)\rceil \cdot S_{\mathcal{B}}(SZ_{\mathcal{B}}(M, k, d, \gamma, \xi_B), k, d, \gamma, \xi_B) + 3M/2$ space.*

Proof. First we need an upper bound for the size of the blocks at level 0, i.e., p_0 which is given by the contrapositive statement of the claim below.

Lemma 21. *If $p_0 \geq \frac{3M}{2}$ then with probability $1-\xi$, $p_0 + \nu \geq \hat{M}$.*

Proof. By applying the noise bounds from Lemma 14 and using the assumption that $p_0 \geq \frac{3M}{2}$, we have that with probability $1-\xi$,

$$\begin{aligned} p_0 + \nu \geq \frac{3M}{2} - \frac{4}{\varepsilon}\log(\frac{2T}{\xi}) &> \frac{3M}{2} - \frac{M}{3} \\ &= M + \frac{M}{6} \\ &> M + \frac{2}{\varepsilon}\log(\frac{2T}{\xi}) > \hat{M} \end{aligned}$$

□

Thus, we only need to store at most $3M/2$ points for the block of input points at level 0, plus the additional space required to execute the coreset construction. Note that the semicoreset computation of level 0 blocks consumes space $S_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2)$ and since the largest coreset construction wrt non-DP algorithm $\mathcal{B}$ is the union of at most u coresets that is reduced to a single coreset, and the largest input to non-DP coreset algorithm is the resulting DP semicoreset size — the additional storage size is at most $S_{\mathcal{B}}(SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2), k, d, \gamma) + (u-1) \cdot S_{\mathcal{B}}(SZ_{\mathcal{B}}(M, k, d, \gamma), k, d, \gamma)$. Note that the resulting coreset size for non-DP coreset algorithm $\mathcal{B}$ is independent of the input set size.

□

Lemma 22 (Semicoreset Size of Theorem 5)**.** *The resulting semicoreset has size at most $\tilde{O}(k\log(k) \cdot \gamma^{-4})$.*

Proof. Since $Q_1 \leftarrow \mathcal{A}(P_0, k, d, \varepsilon, \delta, \kappa, \eta)$, size of Q_1 is $SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2)$. Now $Q_2, \ldots, Q_u$ are obtained by running non-DP algorithm $\mathcal{B}$. In order to simplify our notation, we invoke the state of the art non-DP coreset algorithm as $\mathcal{B}$ (see Theorem 10). Thus the semicoreset size of Q_i where $2 \leq i \leq u$ is $\tilde{O}(k \log k \cdot \gamma^{-4})$. Finally we take the union of the semicoresets $\cup_{1 \leq i \leq u} Q_i$ which has size at most $SZ_{\mathcal{A}}(M, k, d, \varepsilon, \delta, \kappa, \eta_1, \eta_2) + (u-1)O(k \log k \cdot \gamma^{-4})$ and then apply non-DP algorithm $\mathcal{B}$ to obtain a semicoreset of size at most $\tilde{O}(k \log k \cdot \gamma^{-4})$. □

E MISSING PROOFS FROM APPENDIX C

Proof of Proposition 2.

Proof. For any good cell $\mathbf{c}' \neq \mathbf{c}$, define

$$X_{\mathbf{c}'} = \begin{cases} 1 & \text{if } \mathbf{c}' \text{ collides with } \mathbf{c} \\ 0 & \text{otherwise} \end{cases}$$

Now for a fixed $\mathbf{c}' \neq \mathbf{c}$, the expected number of collisions with $\mathbf{c}$ is given by $\mathbb{E}_h[X_{\mathbf{c}'}] = \Pr[X_{\mathbf{c}'} = 1] = \Pr[h(\mathbf{c}) = h(\mathbf{c}')] = \frac{1}{w} = \frac{1}{40k}$. Thus the total number of collisions for $\mathbf{c}'$ with $\mathbf{c}$, in expectation, is given by $\mathbb{E}_h[\sum_{\mathbf{c}'} X_{\mathbf{c}'}] \leq 4k/w = 1/10 \leq 1/2$.

Therefore, by Markov, $\Pr[\sum_{\mathbf{c}'} X_{\mathbf{c}'} > 1] \leq 1/2$. The claim follows. □

Proof of Proposition 3.

Proof.

$$\begin{aligned} \mathop{\mathbb{E}}_h |\mathcal{B}_j| &= \mathop{\mathbb{E}}_h[y + \sum_{\mathbf{c} \notin G_\ell} N_{\ell,\mathbf{c}}] && (36) \\ &= y + \mathop{\mathbb{E}}_h[\sum_{\mathbf{c} \notin G_\ell} N_{\ell,\mathbf{c}}] && (37) \\ &= y + \frac{n_{G_\ell}}{40k} && (38) \end{aligned}$$

The claim follows by Markov inequality. □

Proof of Proposition 4.

Proof. We set the failure probability of DP-HH (see Theorem 8) to be $\xi := \frac{1}{k^2}$. For $\mathbf{c} \in G_\ell$, define $\mathcal{E}_{\mathbf{c}} := \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3$ where $\mathcal{E}_1$ is the event that DP-HH algorithm is correct on all instances, $\mathcal{E}_2$ is the event that there are no collisions between $\mathbf{c}$ and other good cells, and $\mathcal{E}_3$ is the event that there exists a hash bucket that contains only $\mathbf{c}$, and if $N_{\ell,\mathbf{c}} := y$, then the size of that hash bucket is $\leq 2(y + \frac{n_{G_\ell}}{40k})$. We know that $\Pr[\mathcal{E}_1] \geq 1 - \xi = 1 - \frac{1}{k^2}$, by the accuracy guarantee of DP-HH algorithm, $\Pr[\mathcal{E}_2] \geq 1 - \frac{1}{k^2}$ by Proposition 2 (and boosting the success probability), and $\Pr[\mathcal{E}_3] \geq 1 - \frac{1}{k^2}$, by Proposition 3 (and boosting the success probability). Thus by a union bound, we have that for a fixed $\mathbf{c} \in G_\ell$, $\Pr[\mathcal{E}_{\mathbf{c}}] \geq 1 - \frac{3}{k^2}$. By taking a union bound over all $4k$ good cells, with probability at least $1 - \frac{12}{k}$, the claim holds. □

Proof of Proposition 5.

Proof. Recall that $N_{\ell,\mathbf{c}}^{(t)}$ is the number of points in cell $\mathbf{c}$ at time step t. Because x_t is covered in G_ℓ, this means

that we only need to care about the points covered by a good cell $\mathbf{c} \in G_\ell$ but $\mathbf{c} \notin H_{\ell,t}$. Using Proposition 4, we know that if $N_{\ell,\mathbf{c}}^{(t)} \geq \frac{\theta n_{G_\ell}}{20k}$, and $N_{\ell,\mathbf{c}}^{(t)} \geq \frac{2\log(\Lambda)\log^2(k)}{\varepsilon\eta}\operatorname{poly}(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\eta}))$, then with probability at least $1-\frac{12}{k}$, we have that $\mathbf{c} \in H_{\ell,t}$.

If $\mathbf{c} \in G_\ell$, and $\mathbf{c} \notin H_{\ell,t}$, this means that either (1) $N_{\ell,\mathbf{c}}^{(t)} < \frac{2\log(\Lambda)\log^2(k)}{\varepsilon\eta}\operatorname{poly}(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\eta}))$, or, (2) $\mathbf{c}$ is not an θ-HH. For case (1), since there are $4k$ such good cells, the total number of uncovered points in such cells are $\frac{4k\cdot 2\log(\Lambda)\log^2(k)}{\varepsilon\eta}\operatorname{poly}(\log(\frac{T\cdot k\cdot 2^\ell}{\theta\eta}))$. For case (2), this means that $N_{\ell,\mathbf{c}}^{(t)} < \frac{\theta n_{G_\ell}}{20k}$. Again, since there are $4k$ such good cells, the total number of uncovered points in such cells are $< \frac{\theta n_{G_\ell}}{20k}\cdot 4k < \theta n_{G_\ell}$. $\square$