Making Old Things New: A Unified Algorithm for Differentially Private Clustering

Max Dupré la Tour [*1] **Monika Henzinger** [*2] **David Saulpic** [*3]

Abstract

As a staple of data analysis and unsupervised learning, the problem of private clustering has been widely studied under various privacy models. Centralized differential privacy is the first of them, and the problem has also been studied for the local and the shuffle variation. In each case, the goal is to design an algorithm that computes privately a clustering, with the smallest possible error. The study of each variation gave rise to new algorithms: the landscape of private clustering algorithms is therefore quite intricate. In this paper, we show that a 20-year-old algorithm can be slightly modified to work for any of these models. This provides a unified picture: while matching almost all previously known results, it allows us to improve some of them and extend it to a new privacy model, the continual observation setting, where the input is changing over time and the algorithm must output a new solution at each time step.

1. Introduction

The massive, continuous and automatic collection of personal data by public as well as private organisations has raised privacy concerns, both legally (Union, 2018), and in terms of citizens' demands (The Editorial Board of the New York Times, 2020; Deming, 2021). To address those concerns, formal privacy standards for algorithms were defined and developed, with the most prominent one being Differential Privacy (Dwork et al., 2006). This standard allows for a formal definition of privacy, enabling the development of algorithms with provable privacy guarantees. Differentially private algorithms are now widely deployed. For instance, the U.S. Census Bureau uses them to release information from a private Census (Abowd, 2018), Apple employs them to collect data from its phone users (WWDC, 2016; Apple, 2016), and Google has developed an extensive library of private algorithms ready to be used (Guevara, 2019).

The standard definition of *centralized* Differential Privacy ensures that these algorithms "behave roughly the same way" on two databases differing by only a single element. The motivation behind this is to prevent inferring from the result whether a specific element is present in the database, thereby protecting against membership-inference attacks. Stronger notions of privacy exist: most notably, the *local* model, where the adversary observes not only the result but also all communications between a server and data owners. The communication must not reveal the presence of a specific element in the database. This guarantee is much stronger but comes at a price: achieving it significantly degrades the accuracy of the algorithm's answers. Therefore, some intermediate models have been defined, offering stronger privacy guarantees than the centralized model while maintaining better accuracy than in the local model.

Those guarantees are only valid for a *static* database. However, real-life data often evolves over time, as seen in Apple's case, where personal data is collected and transferred daily. Therefore, a definition of privacy that accounts for such changes is necessary. This is formalized in the *continual observation (or continual release) model* (Dwork et al., 2010).

In this article, we consider one of the most common data-analysis and unsupervised learning primitive, namely k-means clustering. This problem has been extensively studied under various notions of privacy: essentially each privacy model gives rise to a new algorithm, with a new and often delicate analysis. We describe this complex landscape in detail in Appendix A.1. However, all these privacy definitions share common ground: it seems possible that, instead of having specialized algorithms for each of them, one could identify the key properties of the private k-means problem and exploit them to design a unified algorithm. This is precisely the question we consider in this paper:

[*]Equal contribution [1]McGill University, Montreal, Canada [2]Institute for Science and Technology Austria (ISTA), Klosterneuburg, Austria [3]CNRS & IRIF, Université Paris Cité, Paris, France. Correspondence to: Max Dupré la Tour <max.duprelatour@gmail.com>, David Saulpic <david.saulpic@irif.fr>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Is there a single clustering algorithm that could perform well in all privacy models?

We answer this question positively for all privacy models in which clustering is known to be possible. This has a significant benefit: when studying a new privacy model (there are already more than 225 variations of differential privacy! (Desfontaines & Pejó, 2020)), there is a go-to algorithm to try that is likely to succeed. Indeed, we show that this algorithm can easily be made private under continual observation, which is the first such result for that model.

1.1. Our Results

We show that a 20-year-old greedy algorithm from Mettu and Plaxton (Mettu & Plaxton, 2000) can be easily made differentially private (DP). This algorithm provides a non-private approximation to the more general (k, z)-clustering problem. In this problem, the input data consists of a set of points P in $\mathbb{R}^d$, and the goal is to find a set S of k points (the *centers*) in order to minimize the cost, defined as

$$\text{cost}(P, S) = \sum_{p \in P} \min_{s \in S} \text{dist}(p, s)^z.$$

We focus especially on the case where $z = 2$, which is the popular k-means problem (while $z = 1$ is k-median), and also provide results for general z. We denote the optimal cost for the (k, z)-clustering problem as $\text{OPT}_{k,z}$.

We show that a slight variation of the algorithm from Mettu and Plaxton is private, provided that one can privately solve a generalized version of the *max summation* problem. Given a fixed, non-private set of balls in $\mathbb{R}^d$ and a private set of points $P \subset \mathbb{R}^d$, each point in P contributes a value to each ball that contains it. The goal is to output a ball with approximately the maximum value; an algorithm for this problem has error θ if the absolute value of the difference between the actual maximum value and the one returned is at most θ. This is a simplified version of the problem we need to solve, referred to as the *generalized summation problem*, which we formally define in Section 4.

Before stating our result, we note that the quality of the private k-means solution S has to be measured by two parameters: S has *multiplicative approximation* M and *additive error* A when $\text{cost}(S) \leq M \cdot \text{OPT}_{k,z} + A$. Since even the non-private problem is NP-hard to approximate within 1.06 (Cohen-Addad et al., 2022b), we must have $M > 1$ if we insist on a polynomial-time algorithm. The privacy constraints enforce $A > 0$ as well; when the input is in the d-dimensional ball $B_d(0, \Lambda)$ in $\mathbb{R}^d$, (Chaturvedi et al., 2021) showed that A has to be at least $k\sqrt{d} \cdot \Lambda^2$ for any (ε, δ) differentially-private mechanism. In light of this lower bound, we will assume $\Lambda = 1$ in the following.

Our meta-theorem shows how to reduce the computation of a k-means solution to a *repeated* application of an algorithm solving the max summation problem. To provide some intuition, a (perhaps too much) simplified version of the algorithm from Mettu and Plaxton repeats the following process k times: select a ball with approximately maximum value, and remove all balls intersecting with the selected one. Thus, we can use an algorithm for repeated max summation as a black-box. Our main result relates the error of the max summation algorithm to the error of the clustering algorithm:

Informal Theorem 1.1 (see Theorem 3.4 Theorem 4.3, and Lemma F.1)**.** *Let $\beta > 0$. If one can solve privately the repeated max summation problem such that, with probability at least $2/3$, the error is θ, then one can solve DP k-means such that, with probability $1 - \beta$, either of the following guarantees is achievable:*

- *multiplicative approximation $O(1)$ and additive error $\approx k \,\text{polylog}(n/\beta) \cdot \theta$,*
- *or multiplicative approximation $w^*(1+\alpha)$ and additive error $\approx \sqrt{d}\,\text{poly}(k, \log(n/\beta)) \cdot \theta$, where w^* is the best non-private approximation ratio.*[1]

For the more general (k, z)-clustering, in the second case the multiplicative approximation is $w^*(2^z + \alpha)$ and additive error $\approx \sqrt{d}\,\text{poly}(k/\beta, \log(n)) \cdot \theta$.

To illustrate the above informal theorem, in centralized (ε, δ)-DP we can use the exponential mechanism to solve the repeated max summation problem. This is formalized in Lemma F.1, with $\theta = \sqrt{d}\,\text{polylog}(n/\delta)/\varepsilon$.

We highlight a few features of our k-means results: in the first case, the additive error is optimal, as it matches the lower bound of (Chaturvedi et al., 2021). In the second case, the multiplicative approximation is close to optimal, in the sense that it is almost as good as any polynomial-time non-private algorithm. Furthermore, even starting from an algorithm with constant probability of success, we show that the probability can be boosted arbitrarily high.

Our result is actually even stronger: it computes a solution not only to k-means, but to all k'-means for $k' \leq k$, with the same multiplicative and additive error guarantee as above. This allows the use of the so-called elbow method to select the 'correct' value for k without any further loss of privacy. We refer to Appendix A.3.

We apply this meta-theorem to several different privacy settings, and state the bounds obtained in Table 1. We present in this table the bounds for (ε, δ)-privacy (see Section 1.3); we address the particular case $\delta = 0$ in Appendix G.

To summarize our contribution, we match the previous bounds in almost all settings, and make improvements in

[1] For this result, we actually need something slightly stronger than max summation, see Definition 4.1

several cases. For Local and Shuffle DP in one round, we improve exponentially the dependency in the probability for k-means and extend the results to (k, z)-clustering, which partially answers an open question from (Chang et al., 2021). In the MPC model, we improve the dependency in k to get an optimal bound. Finally, we present the first result in the Continual Observation setting. We summarize our bounds in Table 1, and discuss in greater detail the previous algorithms – and why they do not work in full generality – in Appendix A.1.

1.2. Brief Overview

To show the first point of Informal Theorem 1.1, we rely on the algorithm from Mettu and Plaxton. We reinterpret this algorithm, introducing some key changes: first, to make it private, and second, to enable implementation based on an algorithm for the generalized summation problem. To provide some intuition, this algorithm iteratively chooses cluster centers, intuitively by selecting smaller and smaller regions that are far away from any center previously selected, based on the region's "value" (a proxy for the contribution to the cost). For any k, the first k centers form a constant-factor approximation to (k, z)-clustering. It turns out that we can repeatedly use a generalized summation algorithm to compute those "values": we show that, if we have a private generalized summation algorithm with an error of θ, then we can solve (k, z)-clustering with additive error $k\theta$ (see Section 3).

We start this paper by formalizing some general building blocks for private clustering in Section 2, namely techniques that can be used to simplify the input and the problem. We show how to perform all of them based only on estimating the size of the clusters and other related quantities. This includes for k-means (a) a dimension-reduction technique; (b) a technique that improves the approximation ratio from $O(1)$ to almost w^*, the best non-private approximation ratio; (c) a new way of boosting the success probability for k-means. (a) and (b) are well-known for k-means, we extend them for the general (k, z)-clustering problem.

We combine these techniques in Section 4 to obtain the near-optimal approximation factor that we presented in Table 1. For this, we show that the max summation problem can be solved privately using histograms to estimate the value of each ball, and then to select the maximum estimated value. This leads directly to a novel private algorithm for the centralized model that is also much simpler than prior algorithms. However, applying the other building blocks (a)-(c) requires estimating the size of each cluster. This would be doable with histograms *if the clusters were known a priori and remained fixed throughout the algorithm*; however, the clusters depend on the input data and are not known a priori, and, thus, computing their size cannot be reduced to a simple histogram query. To solve this issue, we introduce a structural result on the shape of clusters: we show that each cluster is the disjoint union of a small number of *pre-determined* sets. Therefore, to estimate the size of each cluster, it is enough to apply a general summation algorithm on the pre-determined sets, and combine the results on those.

Finally, we present another option: instead of histograms, one can use the exponential mechanism in Section 5 to show that the max summation problem can be solved (in some privacy settings) with a very tiny θ, resulting in a near-optimal additive error.

1.3. Privacy Models

As it is common in the differential privacy literature (see e.g. (Dwork & Roth, 2014)), we will assume our input is given as a multiset, as formalized in Appendix B.1.

Central Differential Privacy: We will use the formalism of (Dwork & Roth, 2014). A *dataset* is a multiset P of points of a universe X. We say that two datasets P, P' are *neighboring* when they differ by a single point, namely $\sum_{x \in X} |P(x) - P'(x)| = 1$. We say that a mechanism $\mathcal{M}$ is (ε, δ)-*differentially private* if for any two neighboring datasets P, P' and any set S, we have:

$$\mathbb{P}(\mathcal{M}(P) \in S) \leq \exp(\varepsilon) \cdot \mathbb{P}(\mathcal{M}(P') \in S) + \delta.$$

We say that an algorithm is ε-*differentially private* if it is $(\varepsilon, 0)$-differentially private. To emphasize the difference to other models, we will refer to this privacy model as *central differential privacy*.

In this paper, we will also study the following other privacy models, deferring the formal definitions to Appendix B.1.

Local Model (Kasiviswanathan et al., 2011): In the local model, there is no trusted central server with access to the entire raw database. Instead, we have n individuals, each with one data point. Two inputs are adjacent if the data of a single user changes. The *transcript* of an algorithm is the sequence of messages exchanged between clients and the server: an algorithm is (ε, δ)-*local differentially-private (LDP)* if the transcript is (ε, δ)-DP. In this paper, we focus on the local privacy model with a single round of communication from clients to the server, also known as the non-interactive model.

Shuffle Model (Bittau et al., 2017): Similarly to the local model, we have n individuals, each with one data point. However, in the shuffle model, a trusted intermediary comes into play between the individuals and the server: the *shuffler*. The shuffler gathers the messages from the individuals and shuffles them randomly before sending them to the server,

Table 1. Comparison with the previous state-of-the-art for (ε, δ) privacy. The success probability is $1 - \beta$. $\alpha \in (0, 1/4]$ and $c > 0$ are precision parameters. For simplicity, dependency in $\log(1/\beta), 1/\varepsilon, \log(1/\delta), \text{polylog}(nd)$ and $\log \log T$ (for continual observation) are hidden, and the diameter is assumed to be $\Lambda = 1$. The notation $O_\alpha(1)$ is to insist that the constant hidden depends on α – here it is $\log(1/\alpha)/\alpha^2$.

Model	Approximation	Error	
Centralized DP	$w^*(1+\alpha)$	$k^{O_\alpha(1)} + k\sqrt{d}$	(Ghazi et al., 2020), Cor. 4.4
	$O(1)$	$k\sqrt{d}$	(Chaturvedi et al., 2021), Lem. F.1
Local DP	$w^*(1+\alpha)$	$\sqrt{n} \cdot \left((k/\beta)^{O_\alpha(1)} + k\sqrt{d}\right)$	k-means only, 1 round, (Chang et al., 2021)
	$w^*(1+\alpha)$	$\sqrt{n} \cdot \left(k^{O_\alpha(1)} + k\sqrt{d}\right)$	k-means only, 1 round, Cor. 4.4
	$w^*(2^z+\alpha)$	$\sqrt{n} \cdot \left((k/\beta)^{O_\alpha(1)} + k\sqrt{d}\right)$	(k,z)-clustering, 1 round, Cor. 4.4
	$O(1/c)$	$\sqrt{nd} \cdot k^{1+O_c(1)}$	k-means only, (Chaturvedi et al., 2022)
Shuffle DP, 1 round	$w^*(1+\alpha)$	$\left((k/\beta)^{O_\alpha(1)} + k\sqrt{d}\right)$	k-means only, (Chang et al., 2021)
	$w^*(1+\alpha)$	$k^{O_\alpha(1)} + k\sqrt{d}$	k-means only, Cor. 4.4
	$w^*(2^z+\alpha)$	$(k/\beta)^{O_\alpha(1)} + k\sqrt{d}$	(k,z)-clustering, Cor. 4.4
MPC	$w^*(1+\alpha)$	$k^{O_\alpha(1)} + k\sqrt{d}$	(Cohen-Addad et al., 2022a), Thm.5.2
	$O(1)$	$k^{2.5} + k^{1.01}\sqrt{d}$	(Cohen-Addad et al., 2022a)
	$O(1)$	$k\sqrt{d}$	Theorem 5.2
Continual observation	$w^*(1+\alpha)$	$(k^{O_\alpha(1)} + k\sqrt{d}) \log^{1.5}(T)$	k-means only, Cor. 4.4
	$w^*(2^z+\alpha)$	$\left((k/\beta)^{O_\alpha(1)} + k\sqrt{d}\right) \log^{1.5}(T)$	(k,z)-clustering, Cor. 4.4

preventing the server from attributing a specific message to a particular individual.[2] Only the transcript of interactions between the server and the shuffler has to be DP.

Continual Observation Model (Dwork et al., 2010): In the continual observation model, the input is not static but evolves over time. The algorithm is given a stream of updates (insertion or deletion) to its dataset, one per time step, and outputs a solution for the input so far at each time step. Two streams are *(event-level) neighboring* if they differ by a single update. The algorithm is (ε, δ)-DP under continual observation if the algorithm mapping a stream to a sequence of outputs is (ε, δ)-DP.

Massively Parallel Computing Model (MPC) : The MPC model is a model for distributed, scalable computation – not necessarily private. The input is initially split among several machines, each of them having local memory sub-polynomial in the total database size (n^κ, for some fixed $\kappa \in (0, 1)$). The machines can send and receive messages from other machines, but the message length cannot exceed the machine's local memory. As opposed to the Local Model, the messages exchanged don't have to be private: the algorithm is (ε, δ)-private if its output is (ε, δ)-DP.

[2]The original motivation behind this is that the random shuffling can be done via secure cryptographic protocols.

2. General Building Blocks for Private Clustering

In this section, we present several techniques used in the literature as preprocessing or postprocessing steps to simplify the task of computing a private clustering. We also extend some of these techniques to allow for greater generality. We will not fixate on a specific privacy model in order to present the results in a modular way. The lemmas in this section will apply to any privacy model, assuming that we are given a partition of the space into k subsets $S_1, \ldots, S_k$ corresponding to a clustering, and that we can estimate for all i, $|S_i \cap P|$, $\sum_{p \in P \cap S_i} p$, and $\sum_{p \in P \cap S_i} |p|_2$. We will prove the existence of private algorithms to compute such a partition and the corresponding estimation in the next sections.

Property 2.1. We say that three sequences $(n_i)_{1 \le i \le k}$, $(\textsc{Sum}_i)_{1 \le i \le k}$, $(\textsc{SumNorm}_i)_{1 \le i \le k}$ verify the Property 2.1 for a partition of the space $\mathbb{R}^d = S_1 \cup \cdots \cup S_k$ with error parameter $e \ge 0$ if:

- $|n_i - |P \cap S_i|| \le e$.
- $\|\textsc{Sum}_i - \sum_{p \in P \cap S_i} p\|_2 \le e$.
- $|\textsc{SumNorm}_i - \sum_{p \in P \cap S_i} \|p\|_2| \le e$.

2.1. Reducing the dimension

Dimension-reduction techniques based on the Johnson-Lindenstrauss lemma allow the projection of the dataset

onto $\hat{d} = O\left(z^4 \cdot \log(k/\beta)\alpha^{-2}\right)$ dimensions, such that with probability $1 - \beta/2$ the clustering cost is preserved up to a $(1 \pm \alpha)$ factor. We call $\pi(P)$ the projected and rescaled dataset. We provide a more detailed description in Appendix C.2 and concentrate here on the main challenge, which is to "lift up" the solution: given a clustering of the projected dataset, how can we compute centers in the original space?

Our main "lifting" technique is modular, as it merely requires approximating the size of each cluster and the sum of the points inside each cluster. Following the analysis of (Ghazi et al., 2020), by applying a standard concentration bound, it holds with probability at least $1 - \beta/2$ that all the points of the projected and rescaled dataset $\pi(P)$ lie within the ball $B_{\hat{d}}(0, \sqrt{2\log(n/\beta)})$. Using a private algorithm $\mathcal{A}$, we compute a solution to (k, z)-clustering $\mathcal{C}$ of $\pi(P)$, with multiplicative approximation M and additive error A. The set of centers $\mathcal{C}$ induces a *private* partition $\hat{S}_1, \ldots, \hat{S}_k$ of $\mathbb{R}^{\hat{d}}$, defined by the Voronoi diagram of $\mathcal{C}$, and a *private* partition $S_1, \ldots, S_k$ of the original space $\mathbb{R}^d$, defined as the preimage $\pi^{-1}(S_1, \ldots, S_k)$.

The natural way of defining centers in $\mathbb{R}^d$ to lift the partition would be to take the k optimal centers μ_z for the $(1, z)$-clustering of each $S_i \cap P$. We say that the *cost induced* by the partition $S_1, \ldots, S_k$ is $\sum_{i=1}^{k} \mathrm{cost}(P \cap S_i, \mu_z(P \cap S_i))$. To lift privately a partition, we will use an approximation of each average $\mu_2(S_i \cap P) = \frac{\sum_{p \in P_i} p}{|P_i|}$ by the quantity $\frac{\mathrm{SUM}_i}{n_i}$ where $(n_i)_{1 \le i \le k}, (\mathrm{SUM}_i)_{1 \le i \le k}$ are two sequences computed *privately* and verifying Property 2.1 for the partition $S_1, \ldots, S_k$. This is formalized in the next lemma.

Lemma 2.2. *Let $\hat{d} = O(z^4 \cdot \log(k/\beta)\alpha^{-2})$, and let $\pi(P)$ be the projected and rescaled dataset in $\mathbb{R}^{\hat{d}}$. Suppose that we are given a partition $\hat{S}_1, \ldots, \hat{S}_k$ of $\mathbb{R}^{\hat{d}}$ that induces a (M, A)-approximation of the (k, z)-clustering problem on $\pi(P)$.*

Let $S_1, \ldots, S_k$ be the partition of $\mathbb{R}^d$ that we obtain by taking the preimage π^{-1} of $\hat{S}_1, \ldots, \hat{S}_k$. Assume that we have two sequences (n_i) and (SUM_i) verifying Property 2.1 for this partition, and consider the set of centers $S = \left\{\frac{\mathrm{SUM}_1}{n_1}, \ldots, \frac{\mathrm{SUM}_k}{n_k}\right\}$. The following holds with probability $1 - \beta$:

In the case of k-means $(z = 2)$, we have $\mathrm{cost}(P, S) \le (1 + \alpha)M \cdot \mathrm{OPT}_{k,2}(P) + \mathrm{polylog}\, n \cdot A + O(ke)$. For general (k, z)-clustering, we have instead $\mathrm{cost}(P, S) \le 2^z(1 + \alpha)M \cdot \mathrm{OPT}_{k,z}(P) + \mathrm{polylog}\, n \cdot A + O(ke)$.

The previous result for k-means is a direct application of the dimension reduction results, but we introduce here the generalization to (k, z)-clustering. This relies on the following new lemma:

Lemma 2.3. *Let P be a multiset of points in $\mathbb{R}^d$ with optimal center μ_z for $(1, z)$-clustering and optimal $(1, 2)$-clustering solution $\mu = \mu_2$. Then,*

$$\sum_{p \in P} \|p - \mu\|^z \le 2^z \sum_{p \in P} \|p - \mu_z\|^z.$$

2.2. Boosting the multiplicative approximation

Let w^* be the best approximation ratio achievable non-privately for (k, z)-clustering. An observation of (Nguyen, 2020) allows the conversion of any private algorithm with constant approximation into an algorithm with approximation almost w^* while increasing the additive error: any $O(1)$-approximation of (k', z)-clustering, with $k' = \alpha^{-O(d)} k \log(n/\alpha)$, is an α-approximation for (k, z)-clustering. If we can compute privately such a solution, one can convert it into a true solution for k-means with approximation $w^*(1 + \alpha)$, while preserving the additive error (see Lemma C.1). This yields the following result:

Lemma 2.4 (Theorem 4 in (Nguyen, 2020))**.** *Suppose we are given a private algorithm $\mathcal{A}$ for (k, z)-clustering that has multiplicative approximation M and additive error $A(k, d)$. Suppose we can privately compute a sequence (n_i) verifying Property 2.1 for the Voronoi diagram of the centers output by $\mathcal{A}$. And let w^* be the best approximation ratio achievable non-privately for (k, z)-clustering. Then, for any $1/4 > \alpha > 0$, there is a private algorithm that computes a solution for (k, z)-clustering with cost at most*

$$w^*(1 + \alpha) \cdot \mathrm{OPT}_{k,z} + O(A(k', d) + k'e), \text{ where } k' = (\alpha/M)^{-O(d)} k \log(nM/\alpha).$$

This can be combined with dimension reduction from the previous section, i.e., we apply this lemma in dimension $\log(k)/\alpha^2$, resulting in $k' = k^{O_\alpha(1)} \log(n/\alpha)$.

2.3. Boosting the success probability

Assume we are given a private algorithm $\mathcal{A}$ that computes with probability $2/3$ (or any constant $> 1/2$) a (M, A)-approximation to (k, z)-clustering. To increase the success probability to $1 - \beta$, the standard technique is to run $\log(1/\beta)$ copies of $\mathcal{A}$ in parallel, and select (privately) the one with the best output. This can be easily implemented using the exponential mechanism, which requires computing the cost of each solution. While this is possible in many settings, in some settings (e.g., Local DP or continual observation), it is not obvious how to do so without "losing" too much privacy.

In the particular case of k-means, we show how this can be done using mere histogram queries. This relies on the following new lemma: the k-means cost of a cluster can be expressed as a function of the points of a cluster, regardless of the location of its center.

Lemma 2.5. *For any multiset E,*

$$\mathrm{cost}(E, \mu(E)) = \sum_{p \in E} \|p\|_2^2 - \frac{\left\|\sum_{p \in E} p\right\|_2^2}{|E|}.$$

Together with the algorithms of the second point of Property 2.1, this yields the following corollary:

Corollary 2.6. *Let $S_1, \ldots, S_k$ be a partition of $B_d(0, \Lambda)$, and suppose we can privately compute the three sequences $(n_i)_{1 \le i \le k}$, $(\mathrm{SUM}_i)_{1 \le i \le k}$, $(\mathrm{SUMNORM}_i)_{1 \le i \le k}$ verifying the Property 2.1. Then one can estimate the k-means cost induced by the partition up to an additive error $O(ke)$.*

Therefore, given an (ε, δ)-DP algorithm $\mathcal{A}$ with multiplicative approximation $M(\varepsilon, \delta)$ and additive error $A(\varepsilon, \delta)$ that succeeds with probability $2/3$, there exists a private algorithm that succeeds with probability $1-\beta$ with multiplicative approximation $M(\varepsilon/\log(1/\beta), \delta/\log(1/\beta))$ and additive error $A(\varepsilon/\log(1/\beta), \delta/\log(1/\beta)) + O(ke)$.

3. The Greedy Algorithm

In this section, we present the original non-private algorithm by Mettu and Plaxton (Mettu & Plaxton, 2000). This algorithm takes as input a multiset P of points of X, and outputs a sequence $(c_1, \ldots, c_n)$ in such a way that, for all $k \le n$, the set $C_k = \{c_1, \ldots, c_k\}$ approximates the optimal (k, z)-clustering cost. [3]

Definition 3.1. Given a ball $B = B(x, r) := \{y \in X, \mathrm{dist}(x, y) \le r\}$, the *value* of B is $\mathrm{Value}(B) := \sum_{p \in B \cap P} (r - \mathrm{dist}(x, p))^z$.

A *child* of a ball $B(x, r)$ is any ball $B(y, r/2)$, where $y \in P$ and $\mathrm{dist}(x, y) \le 10r$.

For any point $x \in X$ and a set of centers C, let $\mathrm{isolated}(x, C)$ denote the ball $B(x, \mathrm{dist}(x, C)/100)$ if C is not empty; and $B(x, \max_{y \in P} d(x, y))$ if $C = \emptyset$. Intuitively, this corresponds to very large ball centered at x that is far away from any center of C.[4]

The algorithm is a simple greedy procedure, that starts with $C = \emptyset$ and repeats n times the following steps: start with the ball $\mathrm{isolated}(x, C)$ with maximum value over all $x \in P$ (with ties broken arbitrarily), and as long as this ball has more than one child (i.e. as long that there are at least two distinct points of the input P "close" to the ball) replace it with the child with maximum value. Let x be the center of the last chosen ball: add x to C, and repeat. We give the pseudo-code of the procedure in the appendix (see Algorithm 4). We call C_k the solution C after k repetitions of the loop.

Theorem 3.2. *(Mettu & Plaxton, 2000) For any fixed z and for all k, the cost of C_k is a $O(1)$-approximation of the optimal (k, z)-clustering cost.*

This algorithm is not private for two reasons: the balls in the sequences σ_i are centered at points of the input, and the value is computed non-privately. To ensure privacy, we introduce the following key modifications to Algorithm 4, resulting in Algorithm 1.

Algorithm 1 RECURSIVEGREEDYMODIFIED(P, θ)

$\mathcal{A} = \mathcal{B}$, ($\mathcal{A}$ is the set of *available* balls)
Let $C_0 = \emptyset$
for i from 0 to $n - 1$ **do**
  Let σ_i denote the singleton sequence (B) where $B \in \mathcal{A}$ has a maximum value up to θ among the available balls.
  while The last element of σ_i has level less than $\lceil \log n \rceil$ **do**
    Select a child with maximum value up to θ of the last element of σ_i, and append it to σ_i.
  end while
  c_{i+1} is the center of the last ball of σ_i
  $C_{i+1} = C_i \cup \{c_{i+1}\}$
  Remove from $\mathcal{A}$ the balls forbidden by c_{i+1}
end for

Modification 1 : We fix a set of balls $\mathcal{B}$ independent from the data: $\mathcal{B}$ consists essentially of balls centered at the points of a fine-grained discretization of the space based on *nets* instead of input data. An η-net of X is a subset $\mathcal{N} \subset X$ satisfying the two following properties: *Packing:* for all distinct $x, x' \in \mathcal{N}$ we have $\mathrm{dist}(x, x') > \eta$, and *Covering:* for all $x \in X$ we have $\mathrm{dist}(x, \mathcal{N}) \le \eta$ (see Appendix A.2).

For any $i \in \{1, \ldots, \lceil \log n \rceil\}$, we let $\mathcal{N}_i$ be a $2^{-i}/2$-net of $B(0, 1)$. An element of $\mathcal{N}_i$ is a *net point* of *level* i. We use the notation $\mathrm{level}(x)$ to denote the level of a net point x.[5] We further define $\mathcal{B}_i$ to be the set of balls of radius 2^{-i} centered in the points of $\mathcal{N}_i$, and $\mathcal{B} = \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_{\lceil \log n \rceil}$.

We change the while condition to stop when we reach a ball of the last level $\lceil \log n \rceil$: at that level, the region considered is so small that any point in it is a good center.

To replace the (non-private) concept of isolated balls $\mathrm{isolated}(x, C)$, we introduce the following new definitions: We say that a ball $B(x, 2^{-i}) \in \mathcal{B}_i$ is *forbidden* by a center $c \in B(0, 1)$ if $\mathrm{dist}(x, c) \le 100 \cdot 2^{-i}$. A ball is *available*

[3] This particular variant, initially referred to as "online" by Mettu and Plaxton, has undergone a name change in more recent literature to "incremental." This change was made to avoid any confusion with other, more common meanings of "online."

[4] In those definitions, we chose the scalar constants $2, 10, 100$ for convenience: the whole analysis can be parameterized more carefully in order to optimize the approximation ratio. We opted for simplicity.

[5] In order to uniquely define the level of a net point, we make the assumption that all $\mathcal{N}_i$ are disjoint.

when it is not forbidden. Note that the radius under consideration is now only dependent on the level i and no longer on the distance of x to the centers. This is a new point of view on the algorithm of (Mettu & Plaxton, 2000), that we believe sheds a new, simplified light on the algorithm.

Modification 2 : The algorithm does not have access to the actual values of the balls. Instead of selecting the ball with the maximum value, the modified algorithm chooses any ball whose value is sufficiently close to the maximum value, with an additive error of θ. Depending on the privacy model, this selection will be performed using either the standard exponential mechanism (Lemma B.1), or using private noisy values instead of the actual values.

Definition 3.3. Let $\theta > 0$. For any set of balls S, we say that a ball $B \in S$ has a maximum value in S *up to* θ, if $\text{Value}(B) \geq \max_{A \in S} \text{Value}(A) - \theta$.

Theorem 3.4. *For any fixed $z > 0$, for all integer $k > 0$ and for all $\theta \geq 1$, the centers C_k produced by Algorithm 1 have cost at most $O(1) \cdot \text{OPT}_{k,z} + O(k\theta)$.*

Thus Algorithm 1 and Theorem 3.4 reduces private (k, z)-clustering to the problem of privately maintaining the ball with maximum value. We show how to solve the latter problem in the next section.

4. Near Optimal Multiplicative Approximation from Histograms

The combination of results from Section 3 and Section 2 shows that, to compute a solution to (k, z)-clustering privately (in any privacy model), one merely needs to compute privately the values of each ball (for Theorem 3.4), the size of each cluster, and $\sum_{p \in P_i} p$ (for Property 2.1).

All this can be done through *generalized bucketized vector summation* (which we abbreviate to *generalized summation*) which is a generalization of histograms: given sets, standard histograms estimate the size of each set, while generalized summation allows for the summation over more complex functions of the sets' elements:

Definition 4.1. [Generalized bucketized vector summation] Let $S_1, ..., S_m \subset X$ be fixed subsets of the universe X, and for $i = 1, \ldots, m$, let $f_i : X \to B_D(0, 1)$ be a function, where $B_D(0, 1)$ is the D-dimensional unit ball. Given a set $P \subseteq X$, a generalized bucketized vector summation algorithm computes an m-dimensional vector v, whose i-th entry $v_i \in \mathbb{R}^D$ for $i = 1, ..., m$, estimates $\sum_{p \in P \cap S_i} f_i(p)$. It has additive error η if $\eta \geq \max_{i \in [1,m]} \left\| v_i - \sum_{p \in P \cap S_i} f_i(p) \right\|_2$.

One important parameter for the performance of such an algorithm is the maximum frequency b such that each item $x \in X$ appears in at most b sets: $b = \max_x |\{i : x \in S_i\}|$.

For example, generalized summation for $f_i(p) = 1$ estimates the number of input points $|S_i \cap P|$ in each S_i ; in that case, the summation problem can be solved by a histogram algorithm, where each column/item corresponds to a set S_i. We show in Appendix B.3 how to turn an algorithm for private histogram into one for private generalized summation. As an illustration, we prove in Lemma B.3 that there exists a private algorithm for generalized summation under continual observation, with additive error roughly $\eta = b\sqrt{D}\log(mDT)^{3/2}\log(b\log T)/\varepsilon$. (Chang et al., 2021) shows an equivalent result for Local and Shuffle DP.

Using generalized summation for clustering. To privately estimate the value of each ball, we apply a generalized summation algorithm with the sets being the balls, and for the j-th ball $B(x_j, r_j)$, $f_j(e) = r_j - \text{dist}(x_j, e)$ when $e \in B(x_j, r_j)$, $f_j(e) = 0$ otherwise. If the private generalized summation has additive error η, then Theorem 3.4 implies that our algorithm is differentially private with multiplicative approximation $O(1)$ and additive error $k\eta$. Thus, we only need to show the existence of a generalized summation algorithm in the different privacy models, and bound the additive error η.

We additionally *boost the multiplicative approximation and success probability* using the techniques from Section 2. This requires estimating the size of each cluster (see Property 2.1). While this is very similar to a generalized summation problem, there is a key difficulty: *the clusters are not fixed in advance*, as opposed to the sets $S_1, ..., S_m$! Therefore, it is not possible to directly apply a generalized summation algorithm for the cluster size estimation.

To resolve this issue, we introduce in the next subsection a decomposition of the space that allows the expression of each cluster as a union of *pre-determined* sets. Our approach is then to apply private generalized summation to these sets, to estimate the size of each cluster (and all other quantities required for the results in Section 2).

4.1. Structured clusters

We show in this section how to transform the clusters into sets with enough structure, to be able to estimate their size using the generalized summation algorithm.

Lemma 4.2. *[See formal statement in Lemma E.1] Fix $\alpha > 0$. There exists a family of sets $\mathcal{G} = \{G_1, ..., G_m\}$, independent of P, efficiently computable, with $m = n^{O(d)}$, and with the following properties. (1) Any point from $B_d(0, 1)$ is part of at most $O(\log n)$ sets G_i. (2) Any possible cluster can be transformed to consist of the union of at most $k \cdot \alpha^{-O(d)} \log(n)$ sets G_i. (3) This transformation preserves the cost up to an additive error of $\alpha \, \text{cost}(P, \mathcal{C}) + \frac{9}{\alpha}$.*

Using this lemma, we can describe more formally our algorithm, which is made of two main parts: first, an algorithm that computes an $(O(1), A(k,d))$-approximation – with an exponential dependency in d in the additive error; second, an algorithm that uses the techniques from Section 2 with a generalized histogram and Lemma 4.2 to boost the multiplicative approximation and reduce the dependency in the dimension.

Algorithm 2 Private-Clustering-low-Dimension($P, k, \varepsilon, \delta$)

1: **Input**: A dataset P, a number of clusters k
2: Compute the values of balls with the private generalized summation algorithm.
3: Run the algorithm from Section 3 to compute a partition $(S_1, ..., S_k)$
4: **Output:** $(S_1, ..., S_k)$

Algorithm 3 Private-Clustering($P, k, \varepsilon, \delta$)

1: **Input**: A dataset P, a number of clusters k
2: Project P onto a lower-dimensional space $\mathbb{R}^{\hat{d}}$, with $\hat{d} = O(z^4 \log(k)\alpha^{-2})$ as in Lem. 2.2.
3: Use Lem. 2.4 to compute a partition $(S'_1, ..., S'_k)$: the clustering algorithm is first Algorithm 2 (with parameter k' as defined in Lem. 2.4), then transform the partition with Lem. 4.2, and estimate the n_i with the private generalized summation algorithm.
4: Transform the partition $(S'_1, ..., S'_k)$ into a structured one with Lem. 4.2, and estimate n_i, SUM_i, SUMNORM_i for this partition using the private generalized summation algorithm.
5: Use Lemma 2.2 to transform the partition into centers in $\mathbb{R}^d$, and Lem. 2.6 to compute the cost of the clustering.
6: **Output:** the centers, with the cost of the clustering.

This algorithm provides a good approximation with probability $2/3$. The probability can then be easily boosted using Corollary 2.6.

Using this structural lemma, we get our first theorem: if for a given privacy model there exists a private algorithm for generalized summation with small additive error, then we can apply Theorem 3.4, Lemma 4.2 and the results of Section 2 to get an efficient algorithm for (k,z)-clustering with arbitrarily high success probability.

Theorem 4.3. *[See full statement in Lemma E.2] Fix a privacy model where there exists a private generalized bucketized vector summation such that with probability at least $2/3$ the additive error is $A(m, b, D)$, where m is the number of sets, b is the maximal number of sets that any given item is contained in, and D is the dimension of the image of the f_i. Assume that the error A is non-decreasing in all parameters.*

Then, for any $\alpha, \beta > 0$ there is a private algorithm for k-means clustering of points in $\mathbb{R}^d$ that, with probability at least $1-\beta$, computes a solution with multiplicative approximation $w^*(1+\alpha)$ and additive error $A(m, k^{O_\alpha(1)}\log(n), d) \cdot m \operatorname{polylog}(n)\log(1/\beta)$, with $m = n^{O_\alpha(\log(k))}$.

There is also a private algorithm for (k,z)-clustering of points in $\mathbb{R}^d$ that, with probability at least $2/3$, computes a solution with multiplicative approximation $w^*(2^z+\alpha)$ and additive error $A(m, k^{O_\alpha(1)} \cdot \log(n), d) \cdot m \operatorname{polylog}(n)$.

Together with the private generalized summation algorithms that we present in the appendix, this leads to the results presented in Table 1:

Corollary 4.4. *There are algorithms for k-means with multiplicative approximation $w^*(1+\alpha)$ that achieve with probability at least $1-\beta$:*

- *additive error $\sqrt{nD} \cdot k^{O_\alpha(1)}/\varepsilon \cdot \operatorname{polylog}(n) \cdot \log(1/\beta)$ in the local (ε, δ)-DP model with one round of communication,*
- *additive error $\sqrt{D} \cdot k^{O_\alpha(1)}/\varepsilon \cdot \operatorname{polylog}(nD/\delta) \cdot \log(1/\beta)$ in the shuffle (ε, δ)-DP model with one round of communication,*
- *additive error $\sqrt{D} \cdot k^{O_\alpha(1)}/\varepsilon \cdot \operatorname{polylog}(n) \cdot \log^{1.5}(DT)\log\log(T)\sqrt{\log(1/\delta)} \cdot \log(1/\beta)$ in (ε, δ)-DP under continual observation.*

The same additive guarantee hold for (k,z)-clustering, with multiplicative approximation $w^*(2^z+\alpha)$ and probability $2/3$.

5. Low Additive Error via Exponential Mechanism

A different view on the algorithm of Algorithm 1 is the following: "the algorithm iteratively selects the ball with the largest value. To initialize the i-th sequence σ_i, the algorithm selects a ball out of the set of available balls (line 4 of Algorithm 1), and to extend the sequence it chooses a child of the current ball (line 6). Instead of using a histogram to compute privately the value of the balls, one can use the exponential mechanism on the non-private values to make each decision. This approach works directly for the Centralized DP setting – though it requires a delicate privacy proof – and its adaptation to the MPC setting requires new ideas.

5.1. Centralized DP

Our first result is an application to centralized DP. For this, we directly apply the algorithm of Section 3, where all balls are chosen with the exponential mechanism: out of a set S of balls, select a ball B randomly with probability proportional to $\exp(\varepsilon' \operatorname{Value}(B))$. Properties of the exponential distribution ensure that the value of each ball selected

will be, with high probability, off from the optimal choice by an additive error $O(\log(|S|n)/\varepsilon')$. In that case, Theorem 3.4 ensures that the algorithm computes a solution with constant multiplicative approximation and additive error $\frac{k\log(k)\log(n/\delta)+k\sqrt{d}}{\varepsilon'}$. We formalize this in Lemma F.1, and show in Lemma F.2 that when run with $\varepsilon' = \frac{\varepsilon}{4\log(n/\delta)}$, the algorithm is (ε, δ)-DP.

5.2. MPC

Implementing Algorithm 1 in a parallel setting is not obvious because of the iterative nature of the greedy choices: selecting the i-th center influences dramatically the subsequent choices, and it is not clear how to parallelize those choices in fewer than k parallel rounds.

Thus, we present a slightly different algorithm building heavily on the analysis of Algorithm 1. We introduce two key modifications. First, we show that instead of iteratively selecting smaller and smaller balls each level can be treated *independently*. Our observation is that it is enough to run Algorithm 1 for all levels of the decomposition, without the inner loop of line 5–7. The second modification allows us to run a *relaxed* version of this greedy algorithm, tailored to a MPC setting: we extract the key elements of the proof of Theorem 3.4 to allow some slack in the implementation.

To formalize this, we refine the definition of availability: a ball $B(x, 2^{-i})$ is *available at scale D for a set of centers C* if for all $c \in C, \mathrm{dist}(x, c) \geq D2^{-i}$ (in Algorithm 1, we defined $D = 100$). For each level $\ell \in 1, \ldots, \lceil \log n \rceil$, our algorithm will compute a set C_ℓ of $2k$ centers, and returns $C = \cup C_\ell$.

Lemma 5.1. *Suppose there is a constant $c_{\mathcal{A}}$ such that, for any level ℓ, (1) the distance between two distinct centers of C_ℓ is greater than $3 \cdot 2^{-\ell}$; and (2) for any center $c \in C_\ell$, the value of the ball $B(c, 2^{-\ell})$ is greater (up to an additive error θ) than the value of any ball of $\mathcal{B}_\ell$ available at scale $c_{\mathcal{A}}$ from C_ℓ. Then,* $\mathrm{cost}(P, C) \leq O(1) \cdot \mathrm{OPT}_{k,z} + O(k\theta)$.

Therefore, each level can be treated independently, and we are left with the following problem: given the set $\mathcal{B}_\ell$ of all balls at level ℓ, compute *privately* a set of $2k$ balls that are (1) far apart and (2) have larger value than any available ball at scale $c_{\mathcal{A}}$ from them. Doing so yields:

Theorem 5.2. *Suppose each machine has memory $k^{O(1)}n^\kappa$, for $\kappa \in (0, 1)$. Then, there is an (ε, δ)-private MPC algorithm, running in $O(1)$ rounds of communication that computes with probability $1 - \beta$ a solution to (k, z)-clustering with cost at most $O(1)\mathrm{OPT}_{k,z} + k\sqrt{d}/\varepsilon \cdot \mathrm{polylog}(n, 1/\delta, 1/\beta)$. Alternatively, the algorithm can compute a solution with cost $w^*(1+\alpha)\mathrm{OPT}_{k,z} + k^{O_\alpha(1)}/\varepsilon \cdot \mathrm{polylog}(n, 1/\delta, 1/\beta)$.*

If all balls of a single level were to fit in a single machine, we would be done, as a simple greedy algorithm would achieve the guarantee of Lemma 5.1: until $2k$ balls have been selected, select privately (using the exponential mechanism) the largest value ball that is available at scale $D = 3$ from the previously selected ones. This would ensure all selected balls are at distance at least $3 \cdot 2^{-\ell}$, and that they have value greater than any available ball at scale 3. However, the memory is limited to $k^{O(1)}n^\kappa$ for some fixed $\kappa \in (0, 1)$, and there are $n^{O(d)}$ many balls.

We propose instead a "merge-and-reduce" algorithm. Each machine gets assigned a set of balls, and runs the greedy algorithm restricted to those balls, with scale parameter $D_0 = 3 \cdot 2^{1/\kappa}$. The outcome in each machine is $2k$ balls. Then, one can merge the results from n^κ machines as follows: centralize the $2kn^\kappa$ selected balls on a single machine, and run the greedy only on those balls, with parameter $D_1 = D_0/2$. Therefore, in one communication round, we can reduce the number of selected balls by a factor n^κ. Repeating this process with $D_i = D_{i-1}/2$ yields $2k$ balls after $1/\kappa$ rounds. We show in Lemma F.5 that they satisfy the guarantee of Lemma 5.1 with $c_{\mathcal{A}} = 2^{1/\kappa+1} \cdot 3$. This concludes the proof of Theorem 5.2.

6. Conclusion

We show that a simple greedy algorithm unifies the problem of private clustering. We believe this unification will help further research and that our algorithm can be used as a baseline for new privacy models, as we illustrate with continual observation.

To achieve this, we revisited an old greedy algorithm from (Mettu & Plaxton, 2000). Despite the complexity of its analysis, this algorithm seems to be quite flexible and powerful. Through our novel analysis, we aim to enhance the understanding of this algorithm, thereby encouraging its application in new contexts.

Acknowledgments

Monika Henzinger: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019564) and the Austrian Science Fund (FWF) grant DOI 10.55776/Z422, grant DOI 10.55776/I5982, and grant DOI 10.55776/P33775 with additional funding from the netidee SCIENCE Stiftung, 2020–2024.

This work was partially done while David Saulpic was at the Institute for Science and Technology, Austria (ISTA). David Sauplic has received funding from the European Union's Horizon 2020 research and innovation programme under the

Marie Sklodowska-Curie grant agreement No 101034413.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abowd, J. M. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2867–2867, 2018.

Apple. WWDC 2016 Keynote, june 2016. https://www.theverge.com/2016/6/17/11957782/apple-differential-privacy-ios-10-wwdc-2016.

Balcer, V., Cheu, A., Joseph, M., and Mao, J. Connecting robust shuffle privacy and pan-privacy. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2384–2403. SIAM, 2021.

Balle, B., Bell, J., Gascón, A., and Nissim, K. The privacy blanket of the shuffle model. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pp. 638–667. Springer, 2019.

Beame, P., Koutris, P., and Suciu, D. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017. doi: 10.1145/3125644. URL https://doi.org/10.1145/3125644.

Becchetti, L., Bury, M., Cohen-Addad, V., Grandoni, F., and Schwiegelshohn, C. Oblivious dimension reduction for *k*-means: beyond subspaces and the johnson-lindenstrauss lemma. In Charikar, M. and Cohen, E. (eds.), *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 1039–1050. ACM, 2019. doi: 10.1145/3313276.3316318. URL https://doi.org/10.1145/3313276.3316318.

Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th symposium on operating systems principles*, pp. 441–459, 2017.

Chan, T. H., Shi, E., and Song, D. 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.

Chang, A., Ghazi, B., Kumar, R., and Manurangsi, P. Locally private k-means in one round. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1441–1451. PMLR, 2021. URL http://proceedings.mlr.press/v139/chang21a.html.

Chaturvedi, A., Nguyen, H. L., and Xu, E. 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, Virtual Event, February 2-9, 2021*, pp. 9101–9108. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17099.

Chaturvedi, A., Jones, M., and Nguyen, H. L. Locally private k-means clustering with constant multiplicative approximation and near-optimal additive error. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 6167–6174. AAAI Press, 2022. doi: 10.1609/AAAI.V36I6.20565. URL https://doi.org/10.1609/aaai.v36i6.20565.

Cohen-Addad, V., Epasto, A., Mirrokni, V., Narayanan, S., and Zhong, P. Near-optimal private and scalable k-clustering. In *NeurIPS*, 2022a.

Cohen-Addad, V., Karthik C. S., and Lee, E. Johnson coverage hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics. In Naor, J. S. and Buchbinder, N. (eds.), *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pp. 1493–1530. SIAM, 2022b. doi: 10.1137/1.9781611977073.63. URL https://doi.org/10.1137/1.9781611977073.63.

Deming, D. Balancing privacy with data sharing for the public good, 2021. https://www.nytimes.com/2021/02/19/business/privacy-open-data-public.html.

Desfontaines, D. and Pejó, B. Sok: Differential privacies. *Proc. Priv. Enhancing Technol.*, 2020(2):288–313, 2020.

doi: 10.2478/POPETS-2020-0028. URL https://doi.org/10.2478/popets-2020-0028.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In Halevi, S. and Rabin, T. (eds.), *Theory of Cryptography*, pp. 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 715–724, 2010.

Epasto, A., Mukherjee, T., and Zhong, P. Differentially private clustering in data streams. 2023. doi: 10.48550/arXiv.2307.07449. URL https://doi.org/10.48550/arXiv.2307.07449.

Evfimievski, A., Gehrke, J., and Srikant, R. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 211–222, 2003.

Fichtenberger, H., Henzinger, M., and Ost, W. Differentially private algorithms for graphs under continual observation. In Mutzel, P., Pagh, R., and Herman, G. (eds.), *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pp. 42:1–42:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPIcs.ESA.2021.42. URL https://doi.org/10.4230/LIPIcs.ESA.2021.42.

Ghazi, B., Kumar, R., and Manurangsi, P. Differentially private clustering: Tight approximation ratios. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, 2020.

Ghazi, B., Golowich, N., Kumar, R., Pagh, R., and Velingker, A. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 463–488. Springer, 2021a.

Ghazi, B., Kumar, R., Manurangsi, P., Pagh, R., and Sinha, A. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *International Conference on Machine Learning*, pp. 3692–3701. PMLR, 2021b.

Goodrich, M. T., Sitchinava, N., and Zhang, Q. Sorting, searching, and simulation in the mapreduce framework. In Asano, T., Nakano, S., Okamoto, Y., and Watanabe, O. (eds.), *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, volume 7074 of *Lecture Notes in Computer Science*, pp. 374–383. Springer, 2011. doi: 10.1007/978-3-642-25591-5_39. URL https://doi.org/10.1007/978-3-642-25591-5_39.

Guevara, M. Enabling developers and organizations to use differential privacy, 2019. URL https://developers.googleblog.com/2019/09/enabling-developers-and-organizations.html.

Gupta, A., Krauthgamer, R., and Lee, J. R. Bounded geometries, fractals, and low-distortion embeddings. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pp. 534–543. IEEE Computer Society, 2003. doi: 10.1109/SFCS.2003.1238226. URL https://doi.org/10.1109/SFCS.2003.1238226.

Gupta, A., Ligett, K., McSherry, F., Roth, A., and Talwar, K. Differentially private combinatorial optimization. In Charikar, M. (ed.), *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pp. 1106–1125. SIAM, 2010. doi: 10.1137/1.9781611973075.90. URL https://doi.org/10.1137/1.9781611973075.90.

Har-Peled, S. and Mendel, M. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006. doi: 10.1137/S0097539704446281. URL https://doi.org/10.1137/S0097539704446281.

Inaba, M., Katoh, N., and Imai, H. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, June 6-8, 1994*, pp. 332–339, 1994. doi: 10.1145/177424.178042. URL https://doi.org/10.1145/177424.178042.

Jain, P., Raskhodnikova, S., Sivakumar, S., and Smith, A. The price of differential privacy under continual observation. *arXiv preprint arXiv:2112.00828*, 2021.

Karloff, H. J., Suri, S., and Vassilvitskii, S. A model of computation for mapreduce. In Charikar, M. (ed.), *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pp. 938–948. SIAM,

2010. doi: 10.1137/1.9781611973075.76. URL https://doi.org/10.1137/1.9781611973075.76.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

Makarychev, K., Makarychev, Y., and Razenshteyn, I. P. Performance of johnson-lindenstrauss transform for *k*-means and *k*-medians clustering. In Charikar, M. and Cohen, E. (eds.), *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 1027–1038. ACM, 2019. doi: 10.1145/3313276.3316350. URL https://doi.org/10.1145/3313276.3316350.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pp. 94–103, 2007. doi: 10.1109/FOCS.2007.66.

Mettu, R. R. and Plaxton, C. G. The online median problem. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pp. 339–348. IEEE Computer Society, 2000. doi: 10.1109/SFCS.2000.892122. URL https://doi.org/10.1109/SFCS.2000.892122.

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

Stemmer, U. and Kaplan, H. Differentially private k-means with constant multiplicative error. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 5436–5446, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/32b991e5d77ad140559ffb95522992d0-Abstract.html.

The Editorial Board of the New York Times. Privacy cannot be a casualty of the coronavirus, april 2020. https://www.nytimes.com/2020/04/07/opinion/digital-privacy-coronavirus.html.

Union, E. General data protection regulation, 2018. https://gdpr.eu/.

Warner, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American statistical association*, pp. 63–69, 1965.

WWDC. Engineering privacy for your users, june 2016. https://developer.apple.com/videos/play/wwdc2016/709/.

A. Extended Introduction

A.1. Previous works and their limits

Most of the works in central DP rely on a process that iteratively selects k "dense" balls and removes the points lying in that ball (Stemmer & Kaplan, 2018; Ghazi et al., 2020; Chaturvedi et al., 2021). While such a process can be optimal both for the multiplicative approximation (Ghazi et al., 2020) and the additive error (Chaturvedi et al., 2021), its iterative nature makes it hard to implement in other models. In particular, because the points of dense balls are removed, this method appears very difficult to implement in the MPC model with fewer than k rounds or in the continual observation model.

The Local-DP algorithm of (Chang et al., 2021) requires only histogram queries and uses them to build a private *coreset* – i.e., a set such that the cost of any k-means solution in the coreset is roughly the same as in the original dataset. The additive error for it is inherently 2^d (where d can be assumed to be $O(\log k)$), and it seems hard to use this algorithm to achieve tight additive error. Furthermore, to achieve a success probability of $1-\beta$, the dimension d has to be $\log(k/\beta)$: therefore, the additive error 2^d reduces to $\mathrm{poly}(k/\beta)$, and it is not clear how to boost the success probability using a single round of communication.

The MPC model differs from the others (e.g. Local DP) as only the output needs to be private, not the communication. Here, the key is to find a summary of the data that fits on a single machine and allows for computing a solution. This summary does not need to be completely private, which is different from the algorithm for Local DP (and from our algorithm for continual observation). In particular, the MPC algorithms of (Cohen-Addad et al., 2022a) compute a summary consisting of weighted input points. This is clearly not private, and cannot be applied to Local DP or other models. Furthermore, the lowest additive error they achieve is a suboptimal $(k^{2.5}+k^{1.01}d)\,\mathrm{polylog}(n)$.

We describe in slightly more detail the MPC algorithms of (Cohen-Addad et al., 2022a) to explain how ours can be seen as a simplification. Both algorithms for low approximation ratio or additive error work in two steps as follows. First, embed the metric into a quadtree: at each level, the space is partitioned by a grid centered at a random location. Then, the algorithm selects, at each level, the k grid cells containing most points. Their centers provide an initial bi-criteria solution, with small additive error but multiplicative $\mathrm{poly}(d)$. To reduce this multiplicative error, the authors use a sampling technique from the coreset literature. Starting from this $\mathrm{poly}(d)$ approximation, they construct $\mathrm{poly}(k,\log n)$ many disjoint instances with the following guarantees. First, each instance has small size, such that it fits on a single machine, allowing the use of any private non-MPC algorithm. Second, as the instances are disjoint, taking the union of the solutions for each of them results in a good bi-criteria solution: it has too many centers, but a good cost. It is then standard to reduce the number of centers, as we describe in greater detail later.

Our algorithm is very similar to the first step, with the key difference being that at each level we allow overlap between the grid cells, instead of a partition. We show that this is directly an $O(1)$-approximation, which avoid the need of building the subinstances.

Parallel work on Continual Observation. Epasto, Mukherjee, and Zhong (Epasto et al., 2023) recently released similar work on private k-means under dynamic updates. Their results are incomparable with ours: their main focus was on designing an algorithm using low memory, which they manage to do in *insertion-only* streams. For those streams, they can achieve a smaller additive error at the cost of a larger constant multiplicative error, see their Corollary 14.

A.2. Technical lemma and Definitions

To work with powered distances, we will often use a modification of the triangle inequality:

Lemma A.1 (Triangle Inequality for Powers, see e.g. (Makarychev et al., 2019))**.** *Let a,b,c be three arbitrary points in a metric space with distance function* dist *and let z be a positive integer. Then for any $\varepsilon>0$*

$$\mathrm{dist}(a,b)^z \le (1+\varepsilon)^{z-1}\,\mathrm{dist}(a,c)^z + \left(\frac{1+\varepsilon}{\varepsilon}\right)^{z-1}\mathrm{dist}(b,c)^z$$

Net Decomposition: A η-net of X is a subset $Z\subset X$ satisfying the following two properties: *Packing:* for all distinct $z,z'\in Z$ we have $\mathrm{dist}(z,z')>\eta$, and *Covering:* for all $x\in X$ we have $\mathrm{dist}(x,Z)\le\eta$. It is well known that the unit ball in $\mathbb{R}^d$ has a η-net for all $\eta>0$ (Gupta et al., 2003). Standard net constructions allow to efficiently *decode*: given any point $p\in B_d(0,1)$, one can find in time $\frac{2^{O(d)}}{r}$ all net points at distance at most r of p. By the packing property of a η-net, there

can be at most $\frac{2^{O(d)}}{r}$ such points, and therefore decoding is done efficiently. We refer to (Har-Peled & Mendel, 2006) for more details, and to the notion of *Efficiently List-Decodable Covers* of (Ghazi et al., 2020).

A.3. About the Elbow Method

The elbow method is a heuristic used to determine the "right" number of clusters. The idea is that the curve of the (k, z)-clustering cost as a function of k is decreasing, and typically admits an "elbow" point, where the gain of adding one new cluster significantly decreases. The elbow method chooses this value of k as the true number of clusters.

To plot this curve, one therefore needs to compute the (k, z)-clustering cost for all value of k: Theorem 3.4 precisely allows that, as for all k' the set of centers $C_{k'}$ computed with Algorithm 4 is a good approximation of optimal k'-means solution. Provided a general summation algorithm, our analysis shows how to run privately the elbow method as a post-processing. Indeed, we can evaluate the cost of all those solutions without leaking any privacy, using Lemma 2.5 shows that we can evaluate the cost of this solution using the structure of clusters and the general summation algorithm (as for boosting the probabilities). Lifting the result up in $\mathbb{R}^d$ can also be done as post-processing, using the result of the general summation algorithm.

B. About Differential Privacy

B.1. Formal Definitions

In this section, we define formally the different models of privacy considered.

All of those use multisets, that are standardly defined as follows. A *multiset* P of points in a *universe* X is a function $P : X \to \mathbb{N}$. We will use set notation and vocabulary for multisets. In particular for any function f we will denote by $\sum_{p \in P} f(p) := \sum_{x \in X} P(x) \cdot f(x)$, and for any subset Y of X, $P \cap Y$ is the multiset $x \mapsto P(x) \cdot \mathbb{1}_{x \in Y}$.

We let $\varepsilon > 0$ and $\delta \in [0, 1]$ be the privacy parameters.

Local Differential Privacy : This model was formally defined in (Kasiviswanathan et al., 2011), although having old roots in the study of randomized response (see e.g. (Warner, 1965; Evfimievski et al., 2003)). We follow here the presentation of (Dwork & Roth, 2014). In the local model of privacy, there are n users. User i holds a datapoint p_i, and the server is interacting with users via *local randomizers*: which are simply algorithms taking as input a database of size 1.

The server runs an algorithm that interacts with users via those local randomizers: this algorithm is (ε, δ)-DP in the local model if, for any user i, the mechanism outputting the results of all local-randomizer run by user i is (ε, δ)-DP.

In this paper, we consider non-interactive setting, where the server interacts only once with each user.

Shuffle Differential Privacy : This model is an attempt to provide the best of the two worlds between centralized and local DP: achieving the utility of centralized while preserving a stronger privacy as in local. This stronger privacy is obtained as the intermediate shuffling step can be implemented via secure cryptographic protocols, therefore strongly protecting individual's privacy.

In the shuffle model of privacy, there are n users and one *shuffler*. Each user holds a datapoint p_i, and the interaction goes in rounds. At each round, users run local-randomizers, and communicate the results to the shuffler. The shuffler releases the results, shuffled uniformly at randomized. The server then receives those messages and goes on with the computation. The procedure is (ε, δ)-DP in the shuffle model when the set of output of the shuffler is (ε, δ)-DP.

This model was introduced by (Bittau et al., 2017), and studied e.g. for aggregation (Balle et al., 2019; Ghazi et al., 2021b), histograms and heavy hitters (Ghazi et al., 2021a), or counting distinct elements (Balcer et al., 2021). Recently clustering (Chang et al., 2021).

Continual Observation Model : Let σ be a stream of updates to a multiset (the dataset): $\sigma(t)$ corresponds to either the addition of a new item to the dataset, the removal of one, or a "no operation" time (where nothing happens to the dataset). Two streams are *neighboring* if they differ by the addition of a single item at a single time step (and possibly its subsequent removal). This is the so-called *event-level* privacy (Dwork et al., 2010). Let $\mathcal{M}$ be a randomized mechanism that associates a stream σ to an output $\mathcal{M}(\sigma) \in \mathcal{X}$. For an $\varepsilon \in \mathbb{R}^+$, $\mathcal{M}$ is (ε, δ)-DP under continual observation if $\mathcal{M}$ is (ε, δ)-DP. Formally,

for any pair of neighboring streams σ, σ', and any possible set of outcomes $\mathcal{S} \subseteq \mathcal{X}$, it holds that

$$\Pr\left[\mathcal{M}(\sigma) \in \mathcal{S}\right] \leq \exp(\varepsilon) \Pr\left[\mathcal{M}(\sigma') \in \mathcal{S}\right] + \delta.$$

The Massively Parellel Computation Model This model is not a privacy model per se, but a restricted model of computation (regardless of privacy). The input data is spread on many machines, each having sublinear memory (typically n^κ, for $\kappa \in (0,1)$). The computation happens in round. At each round, the machines can send and receive messages one message from each machine, but the message length cannot exceed the memory n^κ. In the setting we consider, at the end of the computation each machine knows the solution and can output it. This model was formalized in (Karloff et al., 2010; Goodrich et al., 2011; Beame et al., 2017) and is now a standard for analyzing parallel algorithms designed for large-scale environments. The privacy constraint is that outputing this solution is (ε, δ)-DP. Note that there is no privacy constraints on how exactly the communication happens: we assume the exchanges between machines are trusted. This is different from the Local privacy model, where both the output and the exchanges have to be private.

B.2. The Exponential Mechanism

The *exponential mechanism* is a standard tool in the literature, that allows the following: given a set of n items with private value, select privately an item with value almost maximum. Formally, we have:

Lemma B.1 (Exponential Mechanism (McSherry & Talwar, 2007))**.** *Let $\mathcal{R}$ be some arbitrary range set, and let $f : \mathbb{N}^X \times \mathcal{R} \to \mathbb{R}$. Let $\Delta_f := max_{r \in \mathcal{R}} max_{\|P - P'\|_1 = 1} |f(P,r) - f(P',r)|$ be the ℓ_1-*sensitivity *of f. The* exponential mechanism $\mathcal{M}_E(P, f, \varepsilon)$ *outputs an element $r \in \mathcal{R}$ with probability proportional to $exp(\frac{\varepsilon \cdot f(P,r)}{2 \cdot \Delta_f})$.*

- *The exponential mechanism is ε-differentially private.*
- *For all $t > 0$, with probability at least $1 - e^{-t}$ we have:*

$$\max_{r \in \mathcal{R}} f(P,r) - f\left(P, \mathcal{M}_E(P, f, \varepsilon)\right) \leq \frac{2 \cdot \Delta_f}{\varepsilon} \cdot \ln\left(|\mathcal{R}| + t\right)$$

B.3. Private generalized bucketized vector summation algorithm

Recall first the definition of general summation.

Definition 4.1. [Generalized bucketized vector summation] Let $S_1, ..., S_m \subset X$ be fixed subsets of the universe X, and for $i = 1, \ldots, m$, let $f_i : X \to B_D(0,1)$ be a function, where $B_D(0,1)$ is the D-dimensional unit ball. Given a set $P \subseteq X$, a generalized bucketized vector summation algorithm computes an m-dimensional vector v, whose i-th entry $v_i \in \mathbb{R}^D$ for $i = 1, ..., m$, estimates $\sum_{p \in P \cap S_i} f_i(p)$. It has additive error η if $\eta \geq \max_{i \in [1,m]} \left\| v_i - \sum_{p \in P \cap S_i} f_i(p) \right\|_2$.

One important parameter for the performance of such an algorithm is the maximum frequency b such that each item $x \in X$ appears in at most b sets: $b = \max_x |\{i : x \in S_i\}|$.

Note that we use a somewhat unusual error definition: The error is the maximum over all sets S_i of the ℓ_2-norm of the error within each set.

We now show how to use a private histogram algorithm to create a private algorithm for generalized summation. Private histogram is a well-studied problem, which is the general summation problem with $b = 1$ and $\forall i, p, f_i(p) = 1$. We will show first how to turn a generalized summation problem with $b > 1$ into a generalized summation problem with $b = 1$, and will show afterwards how to solve the latter in different settings.

B.3.1. From $b = 1$ to $b \geq 1$

We start by presenting a simple lemma, that uses group privacy to reduce the general summation to the case where each item is in a single set.

Lemma B.2. *Let $\delta, \beta > 0$, and $1 \geq \varepsilon > 0$. Assume we are given an (ε, δ)-differentially private algorithm for generalized summation with $b = 1$ that has, with probability at least $1 - \beta$, additive error $\eta(m, |P|, \varepsilon, \delta, \beta)$ and runs in time $T(m, |P|, \varepsilon, \delta)$, where m is the total number of sets and $|P|$ is the size of the input. Then there exists an (ε, δ)-differentially private algorithm for generalized summation with $b > 1$, that has, with probability at least $1 - \beta$, additive error $\eta(m, b|P|, \frac{\varepsilon}{b}, \frac{\delta}{3b}, \beta)$ and runs in time $T(m, b|P|, \frac{\varepsilon}{b}, \frac{\delta}{3b}, \beta)$.*

Proof. We transform the input to the generalized summation problem with $b > 1$ to a generalized summation problem with $b = 1$ by creating a new input as follows: for each input data x in the original input, x is duplicated $|\{i : x \in S_i\}|$ times, and each copy is added to the new input as part of a different single set S_i, followed by $b - |\{i : x \in S_i\}|$ all-zero inputs. Thus each data item in the original input is replaced by exactly b data items in the new input.

As a consequence the size of the new input is exactly $b|P|$, and each item belongs to a single set. Running the generalized summation algorithm for $b = 1$ with parameters $\varepsilon', \delta', \beta$, it is (ε', δ')-differentially private and has, with probability at least $1 - \beta$, an additive error of $\eta(m, b|P|, \varepsilon', \delta', \beta)$ and runs in time $T(m, b|P|, \varepsilon', \delta', \beta)$.

This is, however, not directly private with respect to the original input, as two neighboring inputs for the original problem (which, by definition differ by at most one item) might differ in up to b input items in the new problem. By group privacy, the algorithm is, thus, $(b\varepsilon', b\exp((b-1)\varepsilon')\delta')$-DP. Choosing $\varepsilon' = \varepsilon/b$ and $\delta' = \delta/(3b)$ yields an algorithm that is (ε, δ)-DP, as desired (where we used $\varepsilon \leq 1$ to bound $\exp(\varepsilon((b-1)/b)) \leq 3$). □

B.3.2. Generalized summation under continual observation

We now show how to build a general summation algorithm that is private under continual observation. Recall that there are T time steps and a stream σ of updates is given as input one by one such that $\sigma(t)$ corresponds to either the addition, resp., removal of an item of X to resp. from the dataset P or to a "no operation", where the dataset remains unchanged. Two neighboring input streams differ in the operation in at most one time step.

Lemma B.3. *There exists an algorithm for generalized bucketized vector summation that is (ε, δ)-DP under continual observation, and has with probability $1 - \beta$ additive error (simultaneously over all time steps) of $O\left(b\sqrt{D} \cdot \varepsilon^{-1} \log(DT)\sqrt{\log(mDT/\beta)\log(b\log(T)/\delta)}\right)$.*

When $\delta = 0$, the additive error (simultaneously over all time steps) is, with probability $1 - \beta$, $O(bD\varepsilon^{-1}\log^2 T \log(m/\beta))$.

Proof. In our setting, at each time step t the algorithm needs to output an m-dimensional vector v whose i-th entry v_i is a D-dimensional vector that approximates $\sum_{p \in P \cap S_i} f_i(p)$, where P is the current dataset. We first describe and analyze the algorithm for one set S_i, i.e., the case $m = 1$, and then extend it to $m > 1$.

Case 1: $m = 1$. This is the case where only one set with a corresponding function exists, which we call S_1, resp. f_1 for simplicity of notation. For $\delta > 0$, we will use the Gaussian mechanism, and, therefore, we need to bound the ℓ_2-sensitivity of $\sum_{p \in P \cap S_1} f_1(p)$. As two neighboring input streams differ in the operation in at most one time step, the datasets of two neighboring input streams differ in at most two points. As the function f_1 returns a value from the unit ℓ_2-ball $B_D(0, 1)$, the ℓ_2-sensitivity of the function $\sum_{p \in P \cap S_1} f_1(p)$ is $L_2 = 2$.

For $\delta = 0$, we will use the Laplace mechanism and need to bound the ℓ_1-sensitivity of the function $\sum_{p \in P \cap S_1} f_1(p)$: The Cauchy-Schwarz inequality shows that it is $2\sqrt{D}$.

Case 1.1: $D = 1$. In the special case $D = 1$ (i.e., f_1 returnsa scalar) and $m = 1$ (i.e., there is a single set), the problem is called *continual counting*, which can be solved by the so-called binary-tree mechanism, studied by (Dwork et al., 2010; Chan et al., 2011; Fichtenberger et al., 2021).[6] It works as follows: there is a data structure storing, for each multiple of a power of 2 (namely $j \cdot 2^\ell$), the value of the sum between time steps $j \cdot 2^\ell$ and $(j+1)2^\ell - 1$, plus an additive noise drawn from the Laplace distribution with parameter $2\log(T)/\varepsilon$. Let $U_\ell(j)$ be this noisy sum. Then, to compute the sum of all values before time t, the algorithm computes the binary decomposition of t, say $t = \sum_j 2^{\ell_j}$ with ℓ_j increasing, and outputs $\sum_j U_{\ell_j}\left(\sum_{j' > j} 2^{\ell_{j'} - \ell_j}\right)$. One can verify that each time $t' \in [1, t]$ appears exactly once in this sum: more visually, the algorithm maintains a binary tree, where the leafs are the input sequence and each node contains a noisy sum of all leaves below it, and decomposes the segment $(0, t]$ as a disjoint union of nodes.

The privacy of this mechanism stems from the fact that each input value appears in $\log T$ many $U_\ell(j)$ – one for each ℓ with $2^\ell \leq T$. The ℓ_1-sensitivity of continual counting is at most 2, as each input value is in $[-1, 1]$: therefore, standard properties of the Laplace mechanism show that the whole process is ε-DP. More generally, if f_1 has ℓ_1-sensitivity L_1, the mechanism adds to this sum Laplace noise with parameter $\Theta(L_1 \log(T)/\varepsilon)$ and remains ε-DP.

[6] Note that the proof given in these references only shows the claim for numbers from $\{-1, +1\}$, but the same proof goes through verbatim for real numbers from $[-1, 1]$.

For (ε, δ)-DP we will replace the Laplace mechanism by the Gaussian. As a consequence we need to show privacy and accuracy for the sum of up to $\log T$ Gaussian variables. Theorem A.1 in (Dwork & Roth, 2014) shows that adding Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the value of a function f gives an (ε, δ)-DP mechanism if $\sigma \geq \sqrt{2\ln(1.25/\delta)}L_2/\varepsilon$, where L_2 is the ℓ_2-sensitivity of the function computed. Theorem B.1 in (Dwork & Roth, 2014) shows that the composition of $k \geq 2$ (ε, δ)-DP mechanisms is $(k\varepsilon, k\delta)$-DP. Note that the privacy loss of outputting $\sum_j U_{\ell_j}\left(\sum_{j'>j} 2^{\ell_{j'}-\ell_j}\right)$ is no larger than outputting each $U_{\ell_j}\left(\sum_{j'>j} 2^{\ell_{j'}-\ell_j}\right)$ value individually, which corresponds to the composition of up to $\log T$ many Gaussian mechanisms. If each Gaussian mechanism is $((\varepsilon/\log T), (\delta/\log T))$-DP, then the resulting mechanism is (ε, δ)-DP. Thus to guarantee (ε, δ)-DP, we choose the noise for each Gaussian variable to be $\mathcal{N}(0, \sigma'^2)$ with $\sigma' = \sqrt{2\ln((1.25\log T)/\delta)}L_2(\log T)/\varepsilon = \Theta(L_2\sqrt{\log(\log(T)/\delta)}\cdot\log(T)/\varepsilon)$.

For the utility, (Jain et al., 2021) observed that for ε-DP, with probability $1-\beta'$, the bound on the additive error for any *individual* time step is $O(L_1\varepsilon^{-1}\log(T)\sqrt{\log(1/\beta')}(\sqrt{\log(T)}+\sqrt{\log(1/\beta')}))$. This is because to output the sum over time steps between 1 and t, the noise computed is $O(\log T)$ Laplace noise with parameter $\Theta(L_1\log(T)\varepsilon^{-1})$. The concentration bound of random Laplacian variable from (Chan et al., 2011) bounds this sum of noise as desired. Rescaling β' by T, a union-bound over all time steps ensures that with probability $1-\beta'$, over all time steps simultaneously, the maximum error is $O\left(L_1\varepsilon^{-1}\log(T)^2\log(1/\beta')\right)$. For (ε, δ)-DP, we can get an improved error bound (see (Jain et al., 2021)): with probability $1-\beta'$, simultaneously over all time steps the error is at most $O\left(L_2\varepsilon^{-1}\log(T)\sqrt{\log(T/\beta')\log(\log(T)/\delta)}\right)$.

Case 1.2: $D > 1$**.** We extend this result to the case $D > 1$ as follows: instead of scalar values, the inputs and outputs are (D-dimensional) vectors in $B_D(0, 1)$ and, in the same way as for $D = 1$, the noisy sum of a suitable subset of these vectors is stored at the nodes of the binary tree mechanism. We call this a *D-dimensional continual counting algorithm*.

Case $\delta = 0$**:** Recall that the ℓ_1-sensitivity L_1 of $\sum_{p\in P\cap S_i} f_i(p)$ is $O(\sqrt{D})$ in this case. Thus, to ensure ε-DP, we use at each node of the binary tree D-dimensional Laplace noise with parameter $\Theta(\sqrt{D}\log(T)\varepsilon^{-1})$ for each coordinate and the standard privacy proof applies. This gives, with probability at least $1-\beta'$ simultaneously for all time steps, an additive ℓ_∞-error over all D dimensions of $O\left(\sqrt{D}\cdot\varepsilon^{-1}\log(T)^2\log(1/\beta')\right)$. This then implies, with probability at least $1-\beta'$ simultaneously for all time steps, an additive ℓ_2-error over the D dimensions of $O\left(D\cdot\varepsilon^{-1}\log(T)^2\log(1/\beta')\right)$.

Case $\delta > 0$: To ensure (ε, δ)-DP, we use a D-dimensional Gaussian noise vector with standard deviation $O\left(\sqrt{\log(\log(T)/\delta)}\log(T)/\varepsilon\right)$: since the ℓ_2-sensitivity of $\sum_{p\in P\cap S_i} f_i(p)$ is $O(1)$, this noise ensures privacy. Furthermore, with probability at least $1-\beta'$ simultaneously for all time steps, it results in an additive ℓ_∞-error over all D dimensions for *all* time steps simultaneously of $O\left(\varepsilon^{-1}\log(DT)\sqrt{\log(DT/\beta')\log(\log(T)/\delta)}\right)$. This in turn implies, with probability at least $1-\beta'$ simultaneously for all time steps, an additive ℓ_2-error over all D dimensions of $O\left(\sqrt{D}\cdot\varepsilon^{-1}\log(DT)\sqrt{\log(DT/\beta')\log(\log(T)/\delta)}\right)$.

Case 2: $m > 1$**.** Now we extend this result to the histogram case: for this, it is enough to run m D-dimensional continual counting algorithm in parallel, one for each set. Since each element is part of at most b sets, the difference between the input of two neighboring streams is only on b different executions of continual counting mechanism: therefore, this composition of algorithms is $(b\varepsilon, b\delta)$-DP. For completeness, we provide a detailed proof in Claim B.4. We therefore rescale ε and δ by b, to get an ε-DP algorithm, with error at any individual time step and for any fixed set of $O\left(bD\cdot\varepsilon^{-1}\log(T)^2\log(1/\beta')\right)$ with probability $1-\beta'$. We set $\beta' = \beta/m$ to do a union-bound for all sets, and conclude the bound of the lemma. The same settings of parameters conclude the bound for (ε, δ)-DP.

□

Claim B.4. *Consider the following procedure for generalized summation under continual observation. Let $\mathcal{A}_1, ..., \mathcal{A}_m$ be D-dimensional continual counting algorithms that use independent randomness. The procedure creates m streams of input, one for each set S_i, and feeds it to $\mathcal{A}_i$. If each $\mathcal{A}_i$ is ε-DP and each input element is part of b sets, then the procedure is $b\varepsilon$-DP. When $\delta > 0$, the procedure is $(b\varepsilon, b\delta)$-DP.*

Proof. On an input stream $\mathcal{D}$, the algorithm creates one stream per set, say $\mathcal{D}_i$ for set S_i, and feeds it to a continual counting algorithm $\mathcal{A}_i$. Let $\mathcal{A}$ be the whole mechanism that outputs $(\mathcal{A}_1(\mathcal{D}_1), ..., \mathcal{A}_m(\mathcal{D}_m))$. For a dataset $\mathcal{D}$ and any potential solution $s = (s_1, ..., s_m)$ we have $\Pr[\mathcal{A}(\mathcal{D}) = s] = \prod_{i=1}^m\prod_{j=1}^D \Pr[\mathcal{A}_i(\mathcal{D}_i) = s_i]$, because the random choices of the $\mathcal{A}_i$

are independent.

Consider two neighboring streams $\mathcal{D}, \mathcal{D}'$: by assumption, only b of the $\mathcal{D}_i$ differ from $\mathcal{D}'_i$, and those at a single time step.

Case $\delta = 0$**.** We analyze the ratio of probability

$$\frac{\Pr[\mathcal{A}(\mathcal{D}) = s]}{\Pr[\mathcal{A}(\mathcal{D}') = s]} = \prod_{i=1}^{m} \frac{\Pr[\mathcal{A}_i(\mathcal{D}_i) = s_i]}{\Pr[\mathcal{A}_i(\mathcal{D}'_i) = s_i]}.$$

The ratio inside the product is equal to 1 for at least $(m - b)$ many $\mathcal{D}_i$ (the ones where $\mathcal{D}_i = \mathcal{D}'_i$), and since the $\mathcal{A}_i$ are ε-DP, the ratio is at most $\exp(\varepsilon)$ for the remaining b of them. Therefore,

$$\frac{\Pr[\mathcal{A}(\mathcal{D}) = s]}{\Pr[\mathcal{A}(\mathcal{D}') = s]} \leq \exp(b\varepsilon),$$

which concludes the proof when $\delta = 0$.

Case $\delta > 0$**.** For simplicity, we reorder the sets such that only the streams $\mathcal{D}_1, ..., \mathcal{D}_b$ differ. The standard proof of composition (see e.g. Corollary B.2 in (Dwork & Roth, 2014)) gives that

$$\Pr[(\mathcal{A}_1(\mathcal{D}_1), ..., \mathcal{A}_b(\mathcal{D}_b)) = (s_1, ..., s_b)] \leq \exp(b\varepsilon) \cdot \Pr[(\mathcal{A}_1(\mathcal{D}'_1), ..., \mathcal{A}_b(\mathcal{D}'_b)) = (s_1, ..., s_b)] + b\delta.$$

Indeed, for $b = 2$ we have (using that the $\mathcal{A}_i$ are independent, which simplifies compared to (Dwork & Roth, 2014)):

$$\begin{aligned}
\Pr[(\mathcal{A}_1(\mathcal{D}_1), \mathcal{A}_2(\mathcal{D}_2)) = (s_1, s_2)] &= \Pr[\mathcal{A}_1(\mathcal{D}_1) = s_1] \cdot \Pr[\mathcal{A}_2(\mathcal{D}_2) = s_2] \\
&\leq \Pr[\mathcal{A}_1(\mathcal{D}_1) = s_1] \cdot \min\left(1, \exp(\varepsilon) \Pr[\mathcal{A}_2(\mathcal{D}'_2) = s_2] + \delta\right) \\
&\leq \delta + \Pr[\mathcal{A}_1(\mathcal{D}_1) = s_1] \cdot \min\left(1, \exp(\varepsilon) \Pr[\mathcal{A}_2(\mathcal{D}'_2) = s_2]\right) \\
&\leq \delta + \left(\exp(\varepsilon) \Pr[\mathcal{A}_1(\mathcal{D}'_1) = s_1] + \delta\right) \cdot \min\left(1, \exp(\varepsilon) \Pr[\mathcal{A}_2(\mathcal{D}'_2) = s_2]\right) \\
&\leq 2\delta + \exp(2\varepsilon) \Pr[\mathcal{A}_1(\mathcal{D}'_1) = s_1] \Pr[\mathcal{A}_2(\mathcal{D}'_2) = s_2] \\
&\leq 2\delta + \exp(2\varepsilon) \Pr[(\mathcal{A}_1(\mathcal{D}'_1), \mathcal{A}_2(\mathcal{D}'_2)) = (s_1, s_2)].
\end{aligned}$$

Generalizing to larger b is a simple induction.

For any $i \geq b + 1$, we have $\Pr[\mathcal{A}_i(\mathcal{D}_i) = s_i] = \Pr[\mathcal{A}_i(\mathcal{D}'_i) = s_i]$, and furthermore all $\mathcal{A}_i$ use independent randomness. Therefore, $\Pr[\mathcal{A}(\mathcal{D}) = s] \leq \exp(b\varepsilon) \Pr[\mathcal{A}(\mathcal{D}') = s] + b\delta$.

Using the same argument as for $\delta = 0$ to consider streams $\mathcal{D}_i$ that do not differ concludes the proof. □

C. Missing Proof and Statement for Section 2

C.1. From Bi-criteria to Proper Solutions

One of the idea used repeatedly in literature is to relax the constraint on k: instead of looking for a set of exactly k centers with good cost, one can first look for k' centers (with $k' \geq k$). Given k' centers, one can compute a private version of the dataset, where each center is weighted by the number of points in its cluster (with appropriate noise). This is private, and therefore one can use any clustering algorithm to find k centers. Furthermore, triangle inequality ensures that the quality of the solution computed will be good, provided that the k' centers are good as well.

Formally, we have:

Lemma C.1. *Let S be a set of k' centers, with $\text{cost}(P, S) \leq M\,\text{OPT}_{k,z} + A$. For each center $s_i \in S$, let S_i be its cluster, and n_i be such that $|n_i - |S_i|| \leq e$. Let C be a set of k centers that is a M' approximation on the dataset containing n_i copies of each s_i. Then, for any $\alpha \geq 0$ C is a solution with multiplicative approximation $\left(2(1 + 1/\alpha)^{z-1}M + (1 + \alpha)^{z-1}\right) M'$ and additive error $(1 + 1/\alpha)^{z-1}A + O(k'e)$.*

To help parse the bounds of this lemma, we note here the two applications we will make: when $M = O(1)$, we will use $\alpha = 1$ to get a multiplicative approximation $O(1)$ and additive error $O(A + k'e)$; and when is much more tiny (e.g. $M = \alpha'^z$ for some $\alpha' < 1$) we will use $\alpha = \alpha'/z$ to get multiplicative approximation $1 + O(\alpha')$ and additive error $O(A/\alpha'^{z-1} + k'e)$.

Proof. Let OPT be the optimal solution for (k,z)-clustering on P. Fix a point $p \in P$, in cluster S_i. We have, by the generalized triangle inequality (Lemma A.1):

$$\text{cost}(s_i, \text{OPT}) \le (1+1/\alpha)^{z-1}\,\text{cost}(p, \text{OPT}) + (1+\alpha)^{z-1}\,\text{cost}(p, s_i).$$

Since C is a M' approximation, we have

$$\begin{aligned}
\sum_{i=1}^{k'} n_i\,\text{cost}(s_i, \text{OPT}) &\le \sum_{i=1}^{k'} |S_i|\,\text{cost}(s_i, \text{OPT}) + k'e \\
&= \sum_{i=1}^{k'}\sum_{p\in S_i} \text{cost}(s_i, \text{OPT}) + k'e \\
&\le \sum_{i=1}^{k'}\sum_{p\in S_i} (1+1/\alpha)^{z-1}\,\text{cost}(p, s_i) + (1+\alpha)^{z-1}\,\text{cost}(p, \text{OPT}) + k'e \\
&= (1+1/\alpha)^{z-1}\,\text{cost}(P,S) + (1+\alpha)^{z-1}\text{OPT}_{k,z} + k'e \\
&\le (1+1/\alpha)^{z-1}\,\text{cost}(P,S) + (1+\alpha)^{z-1}\text{OPT}_{k,z} + k'e \qquad (1)
\end{aligned}$$

Then, we can bound the cost of the full dataset P in solution C:

$$\begin{aligned}
\text{cost}(P,C) &= \sum_{p\in P} \text{cost}(p,C) \\
&\le \sum_{i=1}^{k'}\sum_{p\in S_i} (1+1/\alpha)^{z-1}\,\text{cost}(p,s_i) + (1+\alpha)^{z-1}\,\text{cost}(s_i,C) \\
&= (1+1/\alpha)^{z-1}\,\text{cost}(P,S) + (1+\alpha)^{z-1}\sum_{i=1}^{k'} |S_i|\,\text{cost}(s_i,C) \\
&\le (1+1/\alpha)^{z-1}\,\text{cost}(P,S) + (1+\alpha)^{z-1}\sum_{i=1}^{k'} n_i\,\text{cost}(s_i,C) + (1+\alpha)^{z-1}k'e \\
&\le (1+1/\alpha)^{z-1}\,\text{cost}(P,S) + (1+\alpha)^{z-1}M'\sum_{i=1}^{k'} n_i\,\text{cost}(s_i,\text{OPT}) + (1+\alpha)^{z-1}k'e
\end{aligned}$$

Using $\text{cost}(P,S) \le M\text{OPT}_{k,z} + A$ and Equation (1), we conclude

$$\text{cost}(P,C) \le \left(2(1+1/\alpha)^{z-1}M + (1+\alpha)^{z-1}\right)M' \cdot \text{OPT}_{k,z} + (1+1/\alpha)^{z-1}A + O(k'e).$$

□

This implies in particular Lemma 2.4 since (Nguyen, 2020) showed that, for $k' = k\alpha^{-O(d)}\log(n/\alpha)$, the optimal cost for (k',z)-clustering is at most αOPT_k, namely an α-fraction of the optimal cost with k centers. Thus, we can chose S in Lemma C.1 to be an M-approximation to (k',z)-clustering, which yields Lemma 2.4.

C.2. Reducing the dimension

In the next theorem, a clustering is viewed as a partition $P_1, ..., P_k$, and the center serving part P_i is the optimal center $\mu_z(P_i)$ for $(1,z)$-clustering on P_i. Therefore, the clustering cost of a partition is $\sum_i \sum_{p\in P_i} \text{dist}(p, \mu_z(P_i))^z$. The next theorem states that one can randomly project into $O(\log k)$ dimension while preserving the clustering cost of any partition:

Lemma C.2 (see Theorem 1.3 in (Makarychev et al., 2019), also Becchetti et al. (Becchetti et al., 2019))**.** *Fix some $1/4 \ge \alpha > 0$ and $1 > \beta > 0$. There exists a family of random projection $\pi : \mathbb{R}^d \to \mathbb{R}^{\hat{d}}$ for some $\hat{d} = O(z^4 \cdot \log(k/\beta)\alpha^{-2})$ such that, for any multiset $P \in \mathbb{R}^d$, it holds with probability $1-\beta$ that for any partition of P into k parts $P_1, ..., P_k$,*

$$\sum_{j=1}^{k} \text{cost}\left(\pi(P_j), \mu_z(\pi(P_j))\right) \in (1\pm\alpha)\cdot\left(\frac{\hat{d}}{d}\right)^{z/2} \cdot \sum_{j=1}^{k} \text{cost}(P_j, \mu_z(P_j)).$$

Lemma 2.2. *Let $\hat{d} = O(z^4 \cdot \log(k/\beta)\alpha^{-2})$, and let $\pi(P)$ be the projected and rescaled dataset in $\mathbb{R}^{\hat{d}}$. Suppose that we are given a partition $\hat{S}_1, \dots, \hat{S}_k$ of $\mathbb{R}^{\hat{d}}$ that induces a (M, A)-approximation of the (k, z)-clustering problem on $\pi(P)$.*

Let $S_1, \dots, S_k$ be the partition of $\mathbb{R}^d$ that we obtain by taking the preimage π^{-1} of $\hat{S}_1, \dots, \hat{S}_k$. Assume that we have two sequences (n_i) and $(\textsc{Sum}_i)$ verifying Property 2.1 for this partition, and consider the set of centers $S = \left\{\frac{\textsc{Sum}_1}{n_1}, \dots, \frac{\textsc{Sum}_k}{n_k}\right\}$. The following holds with probability $1 - \beta$:

In the case of k-means ($z = 2$), we have $\mathrm{cost}(P, S) \leq (1 + \alpha)M \cdot \textsc{opt}_{k,2}(P) + \mathrm{polylog}\, n \cdot A + O(ke)$. For general (k, z)-clustering, we have instead $\mathrm{cost}(P, S) \leq 2^z(1 + \alpha)M \cdot \textsc{opt}_{k,z}(P) + \mathrm{polylog}\, n \cdot A + O(ke)$.

Proof. We start with the k-means case. With probability $1 - \beta/2$, Lemma C.2 ensures that the cost of any partition is preserved up to a factor $(1 \pm \alpha)$, then any (a, b)-approximation for $\pi(P)$ is an $((1 + \alpha)a, b)$-approximation for P. Moreover with probability at least $1 - \beta/2$, all the points of projected and rescaled dataset $\pi(P)$ lie within the ball $B_{\hat{d}}(0, \sqrt{2\log(n/\beta)})$. We condition on both of those events: by our assumptions, clustering each P_i to its optimal center is then a $(M, \sqrt{2\log(n/\beta)}^z \cdot A(k, \hat{d}))$-approximation.

Note that the optimal 1-mean center for a cluster P_i is its average $\mu(P_i) := \frac{\sum_{p \in P_i} p}{|P_i|}$. We now bound the additive error incurred by using n_i and $\textsc{Sum}_i$ instead $|P_i|$ and $\sum_{p \in P_i} p$. By assumption, we have $n_i = |P_i| \pm e$ and $\left\|\textsc{Sum}_i - \sum_{p \in P_i} p\right\| \leq e$.

We let $\mathfrak{c}_i = \frac{\textsc{Sum}_i}{n_i}$ be our estimate on $\mu(P_i)$. We first prove that $\mathfrak{c}_i$ is close to $\mu(P_i)$ and then bound the desired cost difference: For this, we use that both the numerator and denominator are well approximated.

First, if a cluster contains fewer than e points, then regardless of the position of $\mathfrak{c}_i$ its 1-means cost is upper-bounded by e, and this is accounted for by the additive error. Otherwise, we can show that $\mathfrak{c}_i$ is very close to $\mu(P_i)$ as follows:

$$\begin{aligned}\left\|\frac{\textsc{Sum}_i}{n_i} - \mu(P_i)\right\|_2 &= \left\|\left(\frac{\textsc{Sum}_i}{n_i} - \frac{\textsc{Sum}_i}{|P_i|}\right) + \left(\frac{\textsc{Sum}_i}{|P_i|} - \frac{|P_i|\mu(P_i)}{|P_i|}\right)\right\|_2 \\ &\leq \|\textsc{Sum}_i\|_2 \left(\frac{1}{|P_i| - e} - \frac{1}{|P_i|}\right) + \frac{\|\textsc{Sum}_i - |P_i|\mu(P_i)\|_2}{|P_i|}\end{aligned}$$

Using the bound on $\textsc{Sum}_i$, and $(1 - x)^{-1} \leq 1 + 2x$ for $x \leq 1/2$, this is upper-bounded by:

$$\left\|\frac{\textsc{Sum}_i}{n_i} - \mu(P_i)\right\|_2 \leq \|\textsc{Sum}_i\|_2 \left(\frac{1}{|P_i|} + \frac{2e}{|P_i|^2} - \frac{1}{|P_i|}\right) + \frac{e}{|P_i|}$$

Now, note that $\|\textsc{Sum}_i\|_2 \leq \left\|\sum_{p \in P_i} p\right\|_2 + e \leq |P_i| + e$. Therefore, we get the upper bound

$$\begin{aligned}\left\|\frac{\textsc{Sum}_i}{n_i} - \mu(P_i)\right\|_2 &\leq \frac{2e(|P_i| + e)}{|P_i|^2} + \frac{e}{|P_i|} \\ &\leq \frac{5e}{|P_i|}, \end{aligned} \tag{2}$$

where the last line follows from $|P_i| \geq e$.

Finally, we get, using the folklore property that for any set P and point x, $\mathrm{cost}(P, x) = \mathrm{cost}(P, \mu(P)) + |P| \, \|\mu(P) - x\|_2^2$ (see e.g. (Inaba et al., 1994) proof of Theorem 2):

$$\begin{aligned}\mathrm{cost}(P_i, \mathfrak{c}_i) &= \mathrm{cost}(P_i, \mu(P_i)) + |P_i| \cdot \|\mathfrak{c}_i - \mu(P_i)\|_2^2 \\ &\leq \mathrm{cost}(P_i, \mu(P_i)) + \frac{O(e^2)}{|P_i|} \\ &\leq \mathrm{cost}(P_i, \mu(P_i)) + O(e),\end{aligned}$$

where the last line uses again $|P_i| \geq e$. Summing over all clusters concludes the proof: the additive error in addition to $\text{polylog}(n) \cdot A(\hat{d})$ is $k \cdot e$, and the multiplicative approximation is $(1+\alpha)M$ (from the dimension reduction guarantee).

For (k,z)-clustering, we use the fact that $\mu(P_i)$ is a 2^z-approximation for $(1,z)$-clustering on P_i, as shown in Lemma 2.3 below. Therefore, $\text{cost}(P_i, \mu(P_i)) \leq 2^z \,\text{cost}(P_i, m_i)$, where m_i is the optimal $(1,z)$-center for P_i: using Equation (2) and Generalized triangle inequality Lemma A.1 therefore yield

$$\begin{aligned}\text{cost}(P_i, \mathfrak{c}_i) &\leq (1+\alpha)\,\text{cost}(P_i, \mu(P_i)) + (4z/\alpha)^{z-1}|P_i| \cdot \|\mathfrak{c}_i - \mu(P_i)\|_2^z \\ &\leq 2^z(1+\alpha)\,\text{cost}(P_i, m_i) + O(e).\end{aligned}$$

Summing over all clusters, this shows that the $\mathfrak{c}_i$ yields an multiplicative approximation $2^z(1+\alpha)M$ and additive error $\text{polylog}(n) \cdot A(\hat{d}) + k \cdot e$. □

Lemma 2.3. *Let P be a multiset of points in $\mathbb{R}^d$ with optimal center μ_z for $(1,z)$-clustering and optimal $(1,2)$-clustering solution $\mu = \mu_2$. Then,*

$$\sum_{p \in P} \|p - \mu\|^z \leq 2^z \sum_{p \in P} \|p - \mu_z\|^z.$$

Proof. Our goal is to bound the distance between μ_z and μ. Consider the line ℓ going through μ_z and μ. Let P_ℓ be the multiset P projected onto ℓ, and p_ℓ the projection of any point p. Since the mean is linear, μ is also the mean of P_ℓ, i.e., $\mu = \frac{\sum_{p_\ell \in P_\ell} p_\ell}{|P|}$. Therefore, we have $\mu - \mu_z = \frac{\sum_{p_\ell \in P_\ell} p_\ell - \mu_z}{|P|}$, and $|P|\|\mu - \mu_z\| \leq \sum_{p_\ell \in P_\ell} \|p_\ell - \mu_z\|$: since projection only decrease the norm, $\|p_\ell - \mu_z\| \leq \|p - \mu_z\|$, and the sum is at most $\sum_{p \in P} \|p - \mu_z\|$.

This means that $\|\mu - \mu_z\| \leq \frac{\sum_{p \in P} \|p - \mu_z\|}{|P|}$and Jensen's inequality yields $\|\mu - \mu_z\|^z \leq \frac{\sum_{p \in P} \|p - \mu_z\|^z}{|P|}$. Therefore, $\sum_{p \in P} \|p - \mu\|^z \leq 2^{z-1} \sum_{p \in P} (\|p - \mu_z\|^z + \|\mu - \mu_z\|^z) \leq 2^z \sum_{p \in P} \|p - \mu_z\|^z$. This concludes the lemma. □

C.3. Boosting the probabilities

Lemma 2.5. *For any multiset E,*

$$\text{cost}(E, \mu(E)) = \sum_{p \in E} \|p\|_2^2 - \frac{\left\|\sum_{p \in E} p\right\|_2^2}{|E|}.$$

Proof. Using properties of squared distances, we have the following:

$$\begin{aligned}\text{cost}(E, \mu(E)) &= \sum_{p \in E} \|p - \mu(E)\|_2^2 = \sum_{p \in E} \sum_{i=1}^{d} |p_i - \mu(E)_i|^2 \\ &= \sum_{p \in E} \sum_{i=1}^{d} p_i^2 + \mu(E)_i^2 - 2p_i\mu(E)_i \\ &= \sum_{p \in E} \|p\|_2^2 + \|\mu(E)\|_2^2 - 2\sum_{i=1}^{d} p_i \mu(E)_i.\end{aligned}$$

Here, we note that $\mu(E) = \frac{1}{|E|}\sum_{y \in E} y$: therefore, $\sum_{p \in E} \|\mu(E)\|_2^2 = |E| \cdot \|\mu(E)\|_2^2 = \frac{\left\|\sum_{y \in E} y\right\|_2^2}{|E|}$. Furthermore,

$\mu(E)_i = \frac{1}{|E|}\sum_{y\in E} y_i$, and so:

$$\begin{aligned}\sum_{p\in E}\sum_{i=1}^{d} p_i\mu(E)_i &= \sum_{i=1}^{d}\left(\frac{1}{|E|}\sum_{y\in E} y_i\right)\left(\sum_{p\in E} p_i\right)\\ &= \frac{1}{|E|}\sum_{i=1}^{d}\left(\sum_{p\in E} p_i\right)^2\\ &= \frac{\left\|\sum_{p\in E} p\right\|_2^2}{|E|}.\end{aligned}$$

Combining those equations concludes the lemma:

$$\begin{aligned}\mathrm{cost}(E,\mu(E)) &= \sum_{p\in E}\|p\|_2^2 + \|\mu(E)\|_2^2 - 2\sum_{i=1}^{d} p_i\mu(E)_i\\ &= \sum_{p\in E}\|p\|_2^2 - \frac{\left\|\sum_{p\in E} p\right\|_2^2}{|E|}.\end{aligned}$$

□

We now turn to the proof of Corollary 2.6.

Corollary 2.6. *Let $S_1,\dots,S_k$ be a partition of $B_d(0,\Lambda)$, and suppose we can privately compute the three sequences $(n_i)_{1\le i\le k}$, $(\textsc{Sum}_i)_{1\le i\le k}$, $(\textsc{SumNorm}_i)_{1\le i\le k}$ verifying the Property 2.1. Then one can estimate the k-means cost induced by the partition up to an additive error $O(ke)$.*

Therefore, given an (ε,δ)-DP algorithm $\mathcal{A}$ with multiplicative approximation $M(\varepsilon,\delta)$ and additive error $A(\varepsilon,\delta)$ that succeeds with probability $2/3$, there exists a private algorithm that succeeds with probability $1-\beta$ with multiplicative approximation $M(\varepsilon/\log(1/\beta),\delta/\log(1/\beta))$ and additive error $A(\varepsilon/\log(1/\beta),\delta/\log(1/\beta)) + O(ke)$.

Proof. To boost the probability from $2/3$ to $1-\beta$, one merely needs to run $\log(1/\beta)$ independent copies of the algorithm, estimate the cost and output the solution with cheapest cost. If the estimate of the cost for each cluster is within an additive $O(e)$ of the true cost, Chernoff bounds ensure the desired guarantees. Therefore, we only need to show that one can estimate the cost of a clustering, provided the estimate values of the lemma.

By assumption, we have $n_i = |P_i| \pm e$, $\|\textsc{Sum}_i\|_2 = \left\|\sum_{p\in P_i} p\right\|_2 \pm e$ and $\textsc{SumNorm}_i = \sum_{p\in P_i}\|p\|_2 \pm e$.

First, in the case where $n_i \le 2e$, then we know $|P_i| \le e$ and the cost of the cluster is at most e: therefore, the estimation e is fine enough.

Otherwise, we have from Lemma 2.5 that the cost of the cluster is $\sum_{p\in P_i}\|p\|_2^2 - \frac{\left\|\sum_{p\in P_i} p\right\|_2^2}{|P_i|}$. The first term is estimated by $\textsc{SumNorm}_i$, up to an additive e. For the second one, we have (dropping for simplicity the subscript $p\in P_i$):

$$\left|\frac{\|\sum p\|_2^2}{|P_i|} - \frac{\|\textsc{Sum}_i\|_2^2}{n_i}\right| \le \frac{\left|\|\sum p\|_2^2 - \|\textsc{Sum}_i\|_2^2\right|}{|P_i|} + \|\textsc{Sum}_i\|_2^2\left|\frac{1}{|P_i|} - \frac{1}{n_i}\right|.$$

We bound the first term using the guarantees on $\|\textsc{Sum}_i\|_2$: $\|\textsc{Sum}_i\|_2^2 = \left(\|\sum p\|_2 \pm e\right)^2 = \|\sum p\|_2^2 \pm (2|P_i|e + e^2)$, where we used in the last inequality that $\|\sum p\|_2 \le |P_i|$. Therefore, using $|P_i| \ge e$, the first term is $\frac{\left\|\sum_{p\in P_i} p\right\|_2^2}{|P_i|} \pm O(e)$.

For the second term, we first bound $\left|\frac{1}{|P_i|} - \frac{1}{n_i}\right|$: using standard approximation of $(1+x)^{-1}$, this is at most $\frac{2e}{|P_i|^2}$. Now, the term $\|\textsc{Sum}_i\|_2^2$ can be bounded as follows: $\|\textsc{Sum}_i\|_2^2 \le 2\|\sum p\|_2 + 2e^2 \le 2(|P_i|^2 + e^2)$. Using again $|P_i| \ge e$, we

conclude

$$\|\textsc{Sum}_i\|_2^2 \left| \frac{1}{|P_i|} - \frac{1}{n_i} \right| \leq 2(|P_i|^2 + e^2) \cdot \frac{2e}{|P_i|^2} = O(e).$$

Combining all those guarantees, we get:

$$\textsc{SumNorm}_i + \frac{\|\textsc{Sum}_i\|_2}{n_i} = \sum_{p \in P_i} \|p\|_2^2 - \frac{\left\|\sum_{p \in P_i} p\right\|_2^2}{|P_i|} \pm O(e).$$

This estimation of the cost concludes the proof. □

D. Missing Proof of Section 3

D.1. The Original Algorithm of (Mettu & Plaxton, 2000)

To emphasize that our algorithmic modifications are light, we state the original algorithm of (Mettu & Plaxton, 2000) in Algorithm 4, in order to allow comparison with our Algorithm 1.

Algorithm 4 RECURSIVEGREEDY(P)

Let $C_0 = \emptyset$
for i from 0 to $n-1$ **do**
  Let σ_i denote the singleton sequence (B) where B is a maximum value ball in $\{\text{isolated}(x, C_i) | x \in P \setminus C_i\}$
  while The last element of σ_i has more than one child **do**
    Select a maximum value child of the last element of σ_i, and append it to σ_i.
  end while
  c_{i+1} is the center of the last ball of σ_i
  $C_{i+1} = C_i \cup \{c_{i+1}\}$
end for

D.2. The Proof of Theorem 3.4

This section is devoted to the proof of Theorem 3.4. All the proof follow the original proof of (Mettu & Plaxton, 2000), adapted to our setting. We restate for convenience both the algorithm and theorem we seek to proof:

Algorithm 3 RECURSIVEGREEDYMODIFIED(P, θ)

$\mathcal{A} = \mathcal{B}$, ($\mathcal{A}$ is the set of *available* balls)
Let $C_0 = \emptyset$
for i from 0 to $n-1$ **do**
  Let σ_i denote the singleton sequence (B) where $B \in \mathcal{A}$ has a maximum value up to θ among the available balls.
  while The last element of σ_i has level less than $\lceil \log n \rceil$ **do**
    Select a child with maximum value up to θ of the last element of σ_i, and append it to σ_i.
  end while
  c_{i+1} is the center of the last ball of σ_i
  $C_{i+1} = C_i \cup \{c_{i+1}\}$
  Remove from $\mathcal{A}$ the balls forbidden by c_{i+1}
end for

Theorem 3.4. *For any fixed $z > 0$, for all integer $k > 0$ and for all $\theta \geq 1$, the centers C_k produced by Algorithm 1 have cost at most $O(1) \cdot \text{OPT}_{k,z} + O(k\theta)$.*

In what follows, we consider a fixed solution Γ with k centers. For the purpose of this analysis, we assume that the algorithm stops after selecting k centers and outputs C_k. Our objective is to compare the cost of the solution C_k to the cost of Γ, and show that $\text{cost}(P, C_k) \leq O(1)\,\text{cost}(P, \Gamma) + O(k\theta)$. Fixing Γ to be the optimal (k, z)-clustering solution will conclude.

To reuse the first part of this section in a different context later (Appendix F.2), we introduce a new parameter $c_{\mathcal{A}} \geq 100$. We denote $\mathcal{A}(c_{\mathcal{A}}, C_k)$ as the set of balls of the form $B(x, 2^{-i}) \in \mathcal{B}_i$ such that for all centers $c \in C_k$, $\text{dist}(x, c) > c_{\mathcal{A}} \cdot 2^{-i}$.

When $c_{\mathcal{A}} = 100$, this corresponds exactly to the set of available balls at the end of the algorithm. We will state the next few definitions and lemmas with $c_{\mathcal{A}}$, however in order to prove Theorem 3.4, we will only use the case $c_{\mathcal{A}} = 100$.

For a center $\gamma \in \Gamma$, we let P_γ denote γ's cluster, which consists of all points in P assigned to γ in the solution Γ. We analyze the cost of each cluster independently as follows.

We split Γ into two parts: Γ_0 is the set of $\gamma \in \Gamma$ such that there exists a center of C_k at distance less than $(c_{\mathcal{A}} + 1) \cdot 2^{-\lceil \log n \rceil}$ from γ, and let $\Gamma_1 = \Gamma - \Gamma_0$. Centers in Γ_0 are very close to centers in C_k, and the cost of their cluster is therefore almost the same in C_k. The bulk of the work is to show that clusters in Γ_1 are also well approximated.

To analyze the cost of those clusters, we will consider the largest ball centered at γ that is still available at the end of the algorithm. The next lemma shows that this is well defined; we show later that points outside of this ball have roughly the same cost in C_k and Γ, as they are far from every center in both case; and most of the work is spent on showing points inside of this ball (called the inner cluster) have also a cheap cost: we will relate their cost to the value of the ball. The key observation for this is that, since the ball is still available, the algorithm selected balls with larger value. As the value is a measure of how expensive a region is, we can show that the ball selected as larger cost in Γ than the inner cluster in C_k. A careful analysis concludes from this that the cost of C_k is cheap relative to the one of Γ.

Lemma D.1. *For any $\gamma \in \Gamma_1$, the following set is not empty:*

$$\{B = B(x, r) \in \mathcal{B} : \operatorname{dist}(x, \gamma) \leq r/2 \text{ and } B \in \mathcal{A}(c_{\mathcal{A}}, C_k)\}$$

Proof. Let x be the closest point to γ in $\mathcal{N}_{\lceil \log n \rceil}$. By the covering property of nets, we know that $\operatorname{dist}(x, \gamma) \leq 2^{-\lceil \log n \rceil}/2$. We show that the ball $B(x, 2^{-\lceil \log n \rceil})$ is in $\mathcal{A}(c_{\mathcal{A}}, C_k)$. For this, assume by contradiction that it is not the case. There exists a center $c \in C_k$ such that $\operatorname{dist}(x, c) \leq c_{\mathcal{A}} \cdot 2^{-\lceil \log n \rceil}$.

Using the triangle inequality, we get $\operatorname{dist}(\gamma, c) \leq \operatorname{dist}(\gamma, x) + \operatorname{dist}(x, c) \leq 2^{-\lceil \log n \rceil}/2 + c_{\mathcal{A}} \cdot 2^{-\lceil \log n \rceil} \leq (c_{\mathcal{A}} + 1) \cdot 2^{-\lceil \log n \rceil}$. Therefore γ is in the set Γ_0, contradicting the assumption that it is in Γ_1. This contradiction completes the proof. □

For any $\gamma \in \Gamma_1$, let $B_\gamma = B(x_\gamma, 2^{-l_\gamma}) \in \mathcal{B}$ be a ball of maximum radius in the set defined in Lemma D.1. We split P_γ into two parts: $In(P_\gamma) := P_\gamma \cap B_\gamma$ and $Out(P_\gamma) = P_\gamma - In(P_\gamma)$.

Intuitively, points in $Out(P_\gamma)$ are about the same distance to γ than to a center in C_k, which allows to easily bound their cost. On the other hand, $In(P_\gamma)$ consists of points from the cluster P_γ that are much closer to γ than to any selected center. We can relate the cost of $In(P_\gamma)$ for the solution C_k to the value of B_γ, as done in the following lemma.

Lemma D.2. *For all $\gamma \in \Gamma_1$, we have:*

$$\operatorname{cost}(In(P_\gamma), C_k) \leq (16c_{\mathcal{A}} + 24)^z \cdot (\operatorname{cost}(In(P_\gamma), \Gamma) + \operatorname{Value}(B_\gamma)) \tag{3}$$

$$\operatorname{cost}(Out(P_\gamma), C_k) \leq (4c_{\mathcal{A}} + 3)^z \cdot \operatorname{cost}(Out(P_\gamma), \Gamma). \tag{4}$$

And for all $\gamma \in \Gamma_0$, we have:

$$\operatorname{cost}(P_\gamma, C_k) \leq 2^z \cdot (\operatorname{cost}(P_\gamma, \Gamma) + |P_\gamma| \cdot ((c_{\mathcal{A}} + 1)/n)^z). \tag{5}$$

Proof. We begin by examining the (more interesting) case where $\gamma \in \Gamma_1$. The first step is to establish the existence of a center in C_k at a distance of $O(2^{-l_\gamma})$ from γ.

Essentially, since the ball B_γ has maximal radius among available balls close to γ, there is one center not too far from that ball. We formalize now this idea. When the first center c_1 is selected by the algorithm, the entire universe $B(0, 1)$ is included in $B(c_1, c_{\mathcal{A}} \cdot 2^{-1})$, and therefore none of the balls of level 1 are in $\mathcal{A}(c_{\mathcal{A}}, C_k)$. By definition, B_γ is in $\mathcal{A}(c_{\mathcal{A}}, C_k)$, so $l_\gamma \geq 2$. According to the covering property of nets, there exists therefore $x \in \mathcal{N}_{l_\gamma - 1}$ such that $\operatorname{dist}(\gamma, x) \leq 2^{-(l_\gamma - 1)}/2 = 2^{-l_\gamma}$. Furthermore, the maximality of the radius of B_γ ensures that the ball $B(x, 2^{-(l_\gamma - 1)}) \in \mathcal{B}$ is not in $\mathcal{A}(c_{\mathcal{A}}, C_k)$. Thus, $\operatorname{dist}(x, C_k) \leq c_{\mathcal{A}} \cdot 2^{-(l_\gamma - 1)} = 2c_{\mathcal{A}} \cdot 2^{-l_\gamma}$. Combining these two inequalities, we get:

$$\operatorname{dist}(\gamma, C_k) \leq \operatorname{dist}(\gamma, x) + \operatorname{dist}(x, C_k) \leq 2^{-l_\gamma} + 2c_{\mathcal{A}} \cdot 2^{-l_\gamma} = (2c_{\mathcal{A}} + 1) \cdot 2^{-l_\gamma}. \tag{6}$$

Proof of Equation (3): Let $p \in In(P_\gamma)$, we want to bound the cost of p for the solution C_k. We have, with triangle inequality: $\text{cost}(p, C_k) = \text{dist}(p, C_k)^z \leq (\text{dist}(p, \gamma) + \text{dist}(\gamma, C_k))^z$.

We already have a bound on $\text{dist}(\gamma, C_k)$, so we turn to $\text{dist}(p, \gamma)$. Given $p \in B_\gamma = B(x_\gamma, 2^{-l_\gamma})$, we have $\text{dist}(p, x_\gamma) \leq 2^{-l_\gamma}$ and by the definition of B_γ, $\text{dist}(x_\gamma, \gamma) \leq 2^{-l_\gamma}/2$. Combining these with the triangle inequality, we obtain:

$$\text{dist}(p, \gamma) \leq \text{dist}(p, x_\gamma) + \text{dist}(x_\gamma, \gamma) \leq 2^{-l_\gamma} + 2^{-l_\gamma}/2 \leq 2 \cdot 2^{-l_\gamma}.$$

This yields

$$\text{cost}(p, C_k) \leq (2 \cdot 2^{-l_\gamma} + (2c_{\mathcal{A}} + 1) \cdot 2^{-l_\gamma})^z = (2c_{\mathcal{A}} + 3)^z \cdot 2^{-z \cdot l_\gamma}. \tag{7}$$

We now bound $2^{-z \cdot l_\gamma}$ as follows. Either $\text{dist}(p, \gamma) \leq 2^{-l_\gamma}/4$: then, we get by the triangle inequality

$$\begin{aligned} \text{dist}(p, x_\gamma) &\leq \text{dist}(p, \gamma) + \text{dist}(\gamma, x_\gamma) \\ &\leq 2^{-l_\gamma}/4 + 2^{-l_\gamma}/2 = 3 \cdot 2^{-l_\gamma}/4 \\ \Rightarrow \qquad 2^{-l_\gamma} &\leq 4 \cdot (2^{-l_\gamma} - \text{dist}(p, x_\gamma)). \end{aligned}$$

Or, $2^{-l_\gamma}/4 \leq \text{dist}(p, \gamma)$. Then, we have $2^{-l_\gamma} \leq 4 \cdot \text{dist}(p, \gamma)$. Therefore, in both cases it holds that $2^{-l_\gamma} \leq \max(4 \cdot \text{dist}(p, \gamma), 4 \cdot (2^{-l_\gamma} - \text{dist}(p, x_\gamma))) \leq 4 \cdot (\text{dist}(p, \gamma) + 2^{-l_\gamma} - \text{dist}(p, x_\gamma))$. Raising both sides to the power of z yields:

$$\begin{aligned} 2^{-z \cdot l_\gamma} &\leq 4^z \cdot (\text{dist}(p, \gamma) + 2^{-l_\gamma} - \text{dist}(p, x_\gamma))^z \\ &\leq 8^z \cdot (\text{dist}(p, \gamma)^z + (2^{-l_\gamma} - \text{dist}(p, x_\gamma))^z). \end{aligned}$$

Plugging this inequality in the bound on $\text{cost}(p, C_k)$ given by Equation (7), we obtain

$$\text{cost}(p, C_k) \leq (2c_{\mathcal{A}}+3)^z \cdot 8^z \cdot (\text{dist}(p, \gamma)^z + (2^{-l_\gamma} - \text{dist}(p, x_\gamma))^z) = (16c_{\mathcal{A}}+24)^z \cdot (\text{dist}(p, \gamma)^z + (2^{-l_\gamma} - \text{dist}(p, x_\gamma))^z).$$

Summing this inequality over all $p \in In(P_\gamma)$, we get

$$\begin{aligned} \text{cost}(In(P_\gamma), C_k) &\leq (16c_{\mathcal{A}} + 24)^z \cdot \Big(\sum_{p \in In(P_\gamma)} \text{dist}(p, \gamma)^z + \sum_{In(P_\gamma)} (2^{-l_\gamma} - \text{dist}(p, x_\gamma))^z \Big) \\ &\leq (16c_{\mathcal{A}} + 24)^z \cdot \Big(\text{cost}(In(P_\gamma), \gamma) + \sum_{p \in B_\gamma \cap P} (2^{-l_\gamma} - \text{dist}(p, x_\gamma))^z \Big) \\ &= (16c_{\mathcal{A}} + 24)^z \cdot (\text{cost}(In(P_\gamma), \Gamma) + \text{Value}(B_\gamma)). \end{aligned}$$

Proving Equation (4): We turn to $Out(P_\gamma)$, and let $p \in Out(P_\gamma)$. As previously, we have from Equation (6): $\text{cost}(p, C_k) \leq (\text{dist}(p, \gamma) + (2c_{\mathcal{A}} + 1) \cdot 2^{-l_\gamma})^z$. We provide in this case as well a bound on 2^{-l_γ}.

The point p is outside the ball $B_\gamma = B(x_\gamma, 2^{-l_\gamma})$ and therefore $\text{dist}(p, x_\gamma) \geq 2^{-l_\gamma}$. Moreover, by definition of B_γ, $\text{dist}(\gamma, x_\gamma) \leq 2^{-l_\gamma}/2$. Thus, we get:

$$\begin{aligned} \text{dist}(p, \gamma) &\geq \text{dist}(p, x_\gamma) - \text{dist}(\gamma, x_\gamma) \\ &\geq 2^{-l_\gamma} - 2^{-l_\gamma}/2 = 2^{-l_\gamma}/2 \\ \Rightarrow \quad 2^{-l_\gamma} &\leq 2 \cdot \text{dist}(p, \gamma). \end{aligned}$$

Hence

$$\begin{aligned} \text{cost}(p, C_k) &\leq (\text{dist}(p, \gamma) + (2c_{\mathcal{A}} + 1) \cdot 2^{-l_\gamma})^z \\ &\leq (\text{dist}(p, \gamma) + (2c_{\mathcal{A}} + 1) \cdot 2 \cdot \text{dist}(p, \gamma))^z \\ &\leq (4c_{\mathcal{A}} + 3)^z \cdot \text{dist}(p, \gamma)^z. \end{aligned}$$

Summing this inequality over all $p \in Out(P_\gamma)$ concludes:

$$\text{cost}(Out(P_\gamma), C_k) \leq (4c_{\mathcal{A}} + 3)^z \cdot \text{cost}(Out(P_\gamma), \Gamma).$$

Proving Equation (5): Finally, we consider the case $\gamma \in \Gamma_0$: by definition, $\text{dist}(\gamma, C_k) \leq (c_{\mathcal{A}}+1) \cdot 2^{-\lceil \log n \rceil}$. Therefore for any $p \in P_\gamma$

$$\begin{aligned}\text{cost}(p, C_k) = \text{dist}(p, C_k)^z &\leq (\text{dist}(p,\gamma) + \text{dist}(\gamma, C_k))^z \\ &\leq 2^z \cdot (\text{dist}(p,\gamma)^z + \text{dist}(\gamma, C_k)^z) \\ &\leq 2^z \cdot \left(\text{dist}(p,\gamma)^z + (c_{\mathcal{A}}+1)^z \cdot 2^{-z\lceil \log n \rceil}\right) \\ &\leq 2^z \cdot (\text{dist}(p,\gamma)^z + ((c_{\mathcal{A}}+1)/n)^z)\,.\end{aligned}$$

Summing this inequality over all $p \in P_\gamma$

$$\text{cost}(P_\gamma, C_k) \leq 2^z \cdot (\text{cost}(P_\gamma, \Gamma) + |P_\gamma| \cdot ((c_{\mathcal{A}}+1)/n)^z)\,. \qquad \square$$

Lemma D.2 allows us to derive a first bound on $\text{cost}(P, C_k)$. Indeed, summing Equation (3) for all $\gamma \in \Gamma_1$, we get

$$\begin{aligned}\text{cost}(\bigcup_{\gamma\in\Gamma_1} In(P_\gamma), C_k) = \sum_{\gamma\in\Gamma_1} \text{cost}(In(P_\gamma), C_k) &\leq (16c_{\mathcal{A}}+24)^z \cdot (\sum_{\gamma\in\Gamma_1} \text{cost}(In(P_\gamma), \Gamma) + \sum_{\gamma\in\Gamma_1} \text{Value}(B_\gamma)) \\ &\leq (16c_{\mathcal{A}}+24)^z \cdot \text{cost}(\bigcup_{\gamma\in\Gamma_1} In(P_\gamma), \Gamma) + (16c_{\mathcal{A}}+24)^z \cdot \sum_{\gamma\in\Gamma_1} \text{Value}(B_\gamma).\end{aligned}$$

Summing Equation (4) for all $\gamma \in \Gamma_1$, we get

$$\begin{aligned}\text{cost}(\bigcup_{\gamma\in\Gamma_1} Out(P_\gamma), C_k) = \sum_{\gamma\in\Gamma_1} \text{cost}(Out(P_\gamma), C_k) &\leq (4c_{\mathcal{A}}+3)^z \cdot \sum_{\gamma\in\Gamma_1} \text{cost}(Out(P_\gamma), \Gamma) \\ &\leq (4c_{\mathcal{A}}+3)^z \cdot \text{cost}(\bigcup_{\gamma\in\Gamma_1} Out(P_\gamma), \Gamma).\end{aligned}$$

Summing Equation (5) for all $\gamma \in \Gamma_0$, we get

$$\begin{aligned}\text{cost}(\bigcup_{\gamma\in\Gamma_0} P_\gamma, C_k) &\leq 2^z \cdot (\sum_{\gamma\in\Gamma_0} \text{cost}(P_\gamma, \Gamma) + \sum_{\gamma\in\Gamma_0} |P_\gamma| \cdot ((1c_{\mathcal{A}}+1)/n)^z) \\ &\leq 2^z \cdot \text{cost}(\bigcup_{\gamma\in\Gamma_0} P_\gamma, \Gamma) + (2c_{\mathcal{A}}+2)^z \cdot n/n^z.\end{aligned}$$

And finally summing the three parts, we obtain

$$\text{cost}(P, C_k) \leq (16c_{\mathcal{A}}+24)^z \cdot \text{cost}(P, \Gamma) + (2c_{\mathcal{A}}+2)^z \cdot n^{1-z} + (16c_{\mathcal{A}}+24)^z \cdot \sum_{\gamma\in\Gamma_1} \text{Value}(B_\gamma). \tag{8}$$

From now on, we will fix the constant $c_{\mathcal{A}} = 100$ for the rest of the proof. Equation (8) allows us to prove Theorem 3.4 in the easy case where all the available balls at the end of the algorithm have values less than θ. Indeed all the balls B_γ are available at the end of the algorithm, in that case $\sum_{\gamma\in\Gamma_1} \text{Value}(B_\gamma) \leq k \cdot \theta$ and the theorem follows immediately from Equation (8) (Recall that $\theta \geq 1$).

We note M_{Value} the maximum value of an available ball at the end of the algorithm. In the rest of the proof, we show how to bound $\sum_{\gamma\in\Gamma_1} \text{Value}(B_\gamma)$ in the remaining case, when $M_{\text{Value}} \geq \theta$.

D.3. Bounding the Values

To do so, we start by showing a first lemma lower bounding the cost of the balls that do not intersect Γ. We say that a ball $B \in \mathcal{B}$ is *covered* by Γ if $B \cap \Gamma \neq \emptyset$ (and recall that in our analysis Γ plays the role of the optimal solution).

Lemma D.3. *If a ball $B \in \mathcal{B}$ is not covered by Γ, then* $\text{cost}(B \cap P, \Gamma) \geq \text{Value}(B)$.

Proof. Consider $B = B(x, r) \in \mathcal{B}$, a ball not covered by Γ. Here, $\text{dist}(x, \Gamma) \geq r$. For any $p \in B \cap P$, the triangle inequality implies $\text{dist}(p, \Gamma) \geq \text{dist}(x, \Gamma) - \text{dist}(x, p) \geq r - \text{dist}(x, p)$. Raising both sides to the power of z and summing for all $p \in B \cap P$, we get $\text{cost}(B \cap P, \Gamma) = \sum_{p\in B\cap P} \text{dist}(p, \Gamma)^z \geq \sum_{p\in B\cap P} (r - \text{dist}(x, p))^z = \text{Value}(B)$. $\square$

The strategy for bounding the sum of values $\sum_{\gamma\in\Gamma_1}\text{val}(B_\gamma)$ relies on the preceding lemma. Our objective is to match each B_γ (for $\gamma\in\Gamma_1$) with an uncovered ball of at least the same value (approximately), ensuring that all the uncovered balls are disjoint. Consequently, we can then upper bound the sum of values by the cost for Γ of those uncovered balls, as established in Lemma D.3.

In order to define the matching, we recall the notations from Algorithm 1. During the i-th loop, the algorithm defines a sequence of balls $\sigma_i=(\sigma_i^1,\sigma_i^2,...)$, that are smaller and smaller, such that the initial ball has maximum value (up to θ) among the available balls, and for $j\geq 2$ the ball σ_i^j has maximum value (up to θ) among the children of σ_i^{j-1}.

In the following lemma, we show that we can *prune* all the sequences σ_i to establish some separation property. This pruning removes some balls in each sequence, ensuring that the value of the first remaining ball in each sequence is at least $M_{\text{Value}}-\theta$, while guaranteeing that none of the remaining balls intersect. We denote x_i^l the center of the ball σ_i^l.

Lemma D.4. *For all $i\in\{1,\dots,k\}$, there exists an index l_i such that:*

- $\text{Value}(\sigma_i^{l_i})\geq M_{\text{Value}}-\theta$.
- *For all $i,i'\in\{1,\dots,k\}$, and for all $l\geq l_i$, $l'\geq l_{i'}$, $\sigma_i^l\cap\sigma_{i'}^{l'}=\emptyset$*

D.3.1. Proof of the Lemma D.4

To compute each l_i, the pruning procedure works as follows. Start with $l_i=1$ for all i. The first condition is clearly satisfied: when σ_i^1 is selected, it maximizes the value among the available balls up to θ.

The procedure enforces the second condition as follows: as long as there exist sequences $\sigma_i,\sigma_{i'}$ with $i<i'$ and two levels $\ell\geq l_i$, $\ell'\geq l_{i'}$ such that $\sigma_i^\ell\cap\sigma_{i'}^{\ell'}\neq\emptyset$, set $l_i=\ell+1$ (i.e., *prune* the sequence σ_i to remove its entries before $\ell+1$)

We will demonstrate that this procedure is well-defined and inductively preserves the first property. Consequently, when it terminates, both conditions are satisfied.

Before proving Lemma D.4, we need some preliminary results. Our first remark on the algorithm is as follows: at the time when a ball σ_i^j is selected by the algorithm, it is available. This is clearly true when the first ball of the sequence is picked in line 4. To prove that it is also true for the balls picked in line 6, it suffices to prove the following lemma.

Lemma D.5. *At any moment of Algorithm 4, if a ball is available, its children are also available.*

Proof. Let $B_1=B(x_1,r)\in\mathcal{B}$ be a ball and let $B_2=B(x_2,r/2)\in\mathcal{B}$ be a child of B_1. By definition, we have $\text{dist}(x_1,x_2)\leq 10\cdot r$. Suppose B_2 is forbidden by some center c: then $\text{dist}(c,x_2)\leq 100\cdot r/2$. Using the triangle inequality yields

$$\text{dist}(c,x_1)\leq\text{dist}(c,x_2)+\text{dist}(x_2,x_1)\leq 10\cdot r+50\cdot r<100\cdot r.$$

Therefore B_1 is also forbidden by c. □

We now establish a simple lemma that sets a limit on the distance between the center of two balls appearing in a same sequence σ_i.

Lemma D.6. *Let $u_0=B(x_0,r_0),\dots,u_l=B(x_l,r_l)$ be a sequence of balls of $\mathcal{B}$ such that for all i, u_{i+1} is a child of u_i. Then* $\text{dist}(x_1,x_l)\leq 20\cdot r_0$.

Proof. By definition, the distance between the center of a ball $u=B(x,r)$ and the center of any child is at most $10\cdot r$. By induction, we get:

$$\begin{aligned}\text{dist}(u_0,u_l)&\leq\sum_{j=0}^{l-1}\text{dist}(u_j,u_{j+1})\\&\leq\sum_{j=0}^{l-1}10\cdot r_j=\sum_{j=0}^{l-1}\frac{10\cdot r_0}{2^j}\\&\leq 20\cdot r_0.\end{aligned}$$

□

The next Lemma is the key to prove that the procedure to compute the l_i's terminates and verifies the conditions of Lemma D.4.

Lemma D.7. *For every i, i', l, l' such that $i < i'$ and $\sigma_i^l \cap \sigma_i^{l'} \neq \emptyset$,*

- $\text{level}(x_{i'}^1) \geq \text{level}(x_i^l) + 2$
- $\text{Value}(\sigma_i^{l+1}) \geq \text{Value}(\sigma_{i'}^1) - \theta$.

Proof. Let i, i', l, l' such that $i < i'$ and $\sigma_i^l \cap \sigma_{i'}^{l'} \neq \emptyset$. We start by proving the first point by contradiction: suppose that $\text{level}(x_{i'}^1) \leq \text{level}(x_i^l) + 1$. We extend the sequence starting from $x_{i'}^{l'}$ to prove the existence of a "descendant" of $x_{i'}^1$ of level $\text{level}(x_i^l) + 1$ close to x_i^l. We will then prove that this descendant became unavailable when c_i was selected, contradicting Lemma D.5.

More precisely, we pick recursively a sequence $y_0, \dots, y_{j_{max}}$ such that $y_0 = x_{i'}^{l'}$ and $B(y_{j+1}, 2^{-\text{level}(y_{j+1})})$ is an arbitrary child of $B(y_j, 2^{-\text{level}(y_j)})$ such that in $\sigma_i^l \cap B(y_{j+1}, 2^{-\text{level}(y_{j+1})}) \neq \emptyset$ and we stop when $\text{level}(y_{j_{max}}) \geq \text{level}(x_i^l) + 1$.

We prove by induction that this sequence can be defined: assume that $B(y_j, 2^{-\text{level}(y_j)}) \cap \sigma_i^l \neq \emptyset$, and let x be a point lying in the intersection. By the covering property of nets, there exists a net point y_{j+1} in $\mathcal{N}_{\text{level}(y_{j+1})}$ such that $\text{dist}(x, y_{j+1}) \leq 2^{-\text{level}(y_{j+1})}/2$. We have $\text{dist}(y_j, y_{j+1}) \leq \text{dist}(y_j, x) + \text{dist}(x, y_{j+1}) \leq 2^{-\text{level}(y_j)} + 2^{-\text{level}(y_{j+1})}/2 \leq 10 \cdot 2^{-\text{level}(y_j)}$. Therefore $B(y_{j+1}, 2^{-\text{level}(y_{j+1})})$ is a child of $B(y_j, 2^{-\text{level}(y_j)})$ and by construction $x \in \sigma_i^l \cap B(y_{j+1}, 2^{-\text{level}(y_{j+1})})$.

There exists a net point y in the sequence $x_{i'}^1, \dots, x_{i'}^{l'}, y_1, \dots, y_{j_{max}}$ of level $\text{level}(x_i^l) + 1$. This point y is either $y_{j_{max}}$ if $\text{level}\, x_{i'}^{l'} < \text{level}(x_i^l) + 1$, or some point in the sequence $x_{i'}^1, \dots, x_{i'}^{l'} = y_0 = y_{j_{max}}$ otherwise.

By construction, the two balls σ_i^l and $B(y_{j_{max}}, 2^{-\text{level}(y_{j_{max}})})$ are intersecting, and therefore

$$\text{dist}(x_i^l, y_{j_{max}}) \leq 2^{-\text{level}(x_i^l)} + 2^{-\text{level}(y_{j_{max}})} \leq 2 \cdot 2^{-\text{level}(x_i^l)} = 4 \cdot 2^{-\text{level}(y)}.$$

Applying Lemma D.6 to the sequence $B(y, 2^{-\text{level}(y)}), \dots, B(y_{j_{max}}, 2^{-\text{level}(y_{j_{max}})})$, we get $\text{dist}(y_{j_{max}}, y) \leq 20 \cdot 2^{-\text{level}(y)}$. Using the triangle inequality, we finally obtain

$$\text{dist}(x_i^l, y) \leq \text{dist}(x_i^l, y_{j_{max}}) + \text{dist}(y_{j_{max}}, y) \leq 24 \cdot 2^{-\text{level}(y)}.$$

On the other hand, applying Lemma D.6 again, we get $\text{dist}(c_i, x_i^l) \leq 20 \cdot 2^{-\text{level}(x_i^l)} = 40 \cdot 2^{-\text{level}(y)}$ and therefore $\text{dist}(c_i, y) \leq \text{dist}(c_i, x_i^l) + \text{dist}(x_i^l, y) \leq 64 \cdot 2^{-\text{level}(y)} \leq 100 \cdot 2^{-\text{level}(y)}$. Hence, the ball $B(y, 2^{-\text{level}(y)})$ became unavailable when c_i was selected and is not available when $\sigma_{i'}^1$ is picked because $i' > i$. But Lemma D.5 guarantees the availability of all the balls within the sequence $\sigma_{i'}^1, \dots, \sigma_{i'}^{l'}, B(y_1, 2^{-\text{level}(y_1)}), \dots, B(y_{j_{max}}, 2^{-\text{level}(y_{j_{max}})})$, including $B(y, 2^{-\text{level}(y)})$, when the algorithm selects $\sigma_{i'}^1$. We have a contradiction and this concludes the proof of the first point.

We turn to the second point. The idea is to prove the existence of a net point x of level $\text{level}(x_i^l) + 1$ such that the ball $B(x, 2^{-\text{level}(x)})$ is a child of σ_i^l, and such that $\text{Value}(B(x, 2^{-\text{level}(x)}))$ is greater than $\text{Value}(\sigma_{i'}^1)$. Given that the algorithm selects a child with the maximum value up to θ, this will allow us to conclude.

More precisely, we start by deriving a bound on $\text{dist}(x_i^l, x_{i'}^1)$. Applying Lemma D.6 we get $\text{dist}(x_{i'}^{l'}, x_{i'}^1) \leq 20 \cdot 2^{-\text{level}(x_{i'}^1)}$. By the fist point of the Lemma, we know that $\text{level}(x_{i'}^1) \geq \text{level}(x_i^l) + 2$ and therefore $\text{dist}(x_{i'}^{l'}, x_{i'}^1) \leq 5 \cdot 2^{-\text{level}(x_i^l)}$. On the other hand we have $\sigma_i^l \cap \sigma_{i'}^{l'} \neq \emptyset$ and so $\text{dist}(x_i^l, x_{i'}^{l'}) \leq 2 \cdot 2^{-\text{level}(x_i^l)}$. Using the triangle inequality, we finally get $\text{dist}(x_i^l, x_{i'}^1) \leq \text{dist}(x_i^l, x_{i'}^{l'}) + \text{dist}(x_{i'}^{l'}, x_{i'}^1) \leq 7 \cdot 2^{-\text{level}(x_i^l)}$.

Now let x be a net point of level $\text{level}(x_i^l) + 1$ such that $x_{i'}^1 \in B(x, 2^{-\text{level}(x)}/2)$, such a point exist by the covering property of nets. We have $\text{dist}(x_i^l, x) \leq \text{dist}(x_i^l, x_{i'}^1) + \text{dist}(x_{i'}^1, x) \leq 7 \cdot 2^{-\text{level}(x_i^l)} + 2^{-\text{level}(x)}/2 \leq 10 \cdot 2^{-\text{level}(x_i^l)}$. Therefore the ball $B(x, 2^{-\text{level}(x)})$ is a child of σ_i^l and could have been chosen by the algorithm instead of σ_i^{l+1}. The algorithm selects a child of σ_i^l with the maximum value up to θ. Hence $\text{Value}(\sigma_i^{l+1}) \geq \text{Value}(B(x, 2^{-\text{level}(x)})) - \theta$.

To conclude the proof, it just remains to show that $\text{Value}(B(x, 2^{-\text{level}(x)})) \geq \text{Value}(\sigma_{i'}^1)$. We will show that for any $p \in P \cap \sigma_{i'}^1$, the contribution of p to the value of $\sigma_{i'}^1$ is lower than its contribution to the value of $B(x, 2^{-\text{level}(x)})$. Let p be a point of $P \cap \sigma_{i'}^1$, the contribution of p to the value of $\sigma_{i'}^1$ is $(2^{-\text{level}(x_{i'}^1)} - \text{dist}(p, x_{i'}^1))^z$. We have established that $\text{level}(x_{i'}^1) \geq \text{level}(x_i^l) + 2 = \text{level}(x) + 1$. Therefore we can bound $2^{-\text{level}(x_{i'}^1)} \leq 2^{-\text{level}(x)}/2$. Moreover, $x_{i'}^1$ is by

construction in $B(x, 2^{-\operatorname{level}(x)}/2)$, hence using the triangle inequality we get $\operatorname{dist}(p, x^1_{i'}) \geq \operatorname{dist}(p, x) - \operatorname{dist}(x, x^1_{i'}) \geq \operatorname{dist}(p, x) - 2^{-\operatorname{level}(x)}/2$. We can now bound the contribution of p to the value of $\sigma^1_{i'}$

$$\begin{aligned}(2^{-\operatorname{level}(x^1_{i'})} - \operatorname{dist}(p, x^1_{i'}))^z &\leq (2^{-\operatorname{level}(x)}/2 - (\operatorname{dist}(p, x) - 2^{-\operatorname{level}(x)}/2))^z \\ &\leq (2^{-\operatorname{level}(x)} - \operatorname{dist}(p, x))^z.\end{aligned}$$

This is precisely the contribution of p to the value of $B(x, 2^{-\operatorname{level}(x)})$. Summing this inequality for all $p \in P \cap \sigma^1_{i'}$, we obtain $\operatorname{Value}(B(x, 2^{-\operatorname{level}(x)})) \geq \operatorname{Value}(\sigma^1_{i'})$, concluding the proof. □

Proof Lemma D.4. We are now ready to prove Lemma D.4. We recall for convenience the procedure to compute the l_i's, described at the beginning of Appendix D.3. Start with $l_i = 1$ for all i, and note that with this choice, the first condition is verified by the design of the algorithm: when σ^1_i is picked, it maximizes the value up to θ among the available balls. In particular, σ^1_i has a value larger up to θ than any ball still available at the end of the algorithm.

To enforce the second condition, proceed as follows: as long as we can find $i < i'$ and $l \geq l_i$, $l' \geq l_{i'}$ such that $\sigma^l_i \cap \sigma^{l'}_{i'} \neq \emptyset$, update $l_i = l + 1$. The first item of Lemma D.7 guarantees that this procedure is well-defined, namely that σ^l_i is not the last ball of the sequence, and that σ^{l+1}_i does indeed exist. Furthermore, the second item of that Lemma ensures that $\operatorname{Value}(\sigma^{l_i+1}_i) \geq \operatorname{Value}(\sigma^1_{i'}) - \theta \geq M_{\text{Value}} - \theta$. Therefore, the first condition remains satisfied after each update.

At each step, one of the l_i gets incremented, so this procedure must terminate because the maximum level is $\lceil \log(n) \rceil$: when it ends, both conditions are satisfied, concluding the proof. □

D.3.2. Back to the proof of Theorem 3.4

Our goal is to identify k disjoint balls with values greater than those of B_γ using Lemma D.4 and then apply Lemma D.3 to finalize the proof.

Proof of Theorem 3.4. To prove the theorem, we define a function ϕ that maps the center of Γ_1 to balls of $\mathcal{B}$ such that for all $\gamma \in \Gamma_1$:

1. for all $\gamma' \in \Gamma_1$ with $\gamma \neq \gamma'$, $\phi(\gamma) \cap \phi(\gamma') = \emptyset$,
2. $\phi(\gamma)$ is not covered by Γ,
3. The value of B_γ is less than $\operatorname{Value}(\phi(\gamma)) + \theta$.

Given such a matching ϕ, we can conclude as follows. Summing the inequality of the third property of ϕ gives

$$\sum_{\gamma \in \Gamma_1} \operatorname{Value}(B_\gamma) \leq k \cdot \theta + \sum_{\gamma \in \Gamma_1} \operatorname{Value}(\phi(\gamma)).$$

Each $\phi(\gamma)$ is not covered by the second property of ϕ, so we can apply Lemma D.3. Moreover those balls are disjoint because of the first property of ϕ and we obtain

$$\begin{aligned}\sum_{\gamma \in \Gamma_1} \operatorname{Value}(\phi(\gamma)) &\leq \sum_{\gamma \in \Gamma_1} \operatorname{cost}(\phi(\gamma), \Gamma) \\ &\leq \operatorname{cost}(\bigcup_{\gamma \in \Gamma_1} \phi(\gamma), \Gamma) \\ &\leq \operatorname{cost}(P, \Gamma).\end{aligned}$$

Putting everything together we finally get

$$\sum_{\gamma \in \Gamma_1} \operatorname{Value}(B_\gamma) \leq k \cdot \theta + \operatorname{cost}(P, \Gamma).$$

Combining this with Equation (8), this conclude the proof of Theorem 3.4.

The rest of the proof is dedicated to the construction of the matching ϕ with the three desired properties. We construct a more general function ϕ that can also map the centers of Γ_0. The restriction of ϕ to Γ_1 will verify the desired properties. Let l_i be the indices provided by Lemma D.4. We have 3 steps in the construction of ϕ:

- First, for any i such that the last ball of the sequence σ_i is covered, we let $B = (c_i, 2^{-\lceil \log n \rceil})$ be this last ball (with center at $c_i \in C_k$) and chose arbitrarily $\gamma_i \in \Gamma$ covering B. We define $\phi(\gamma_i) = B$. We note that, in this case, $\gamma_i \in \Gamma_0$ – since $\operatorname{dist}(c_i, \gamma_i) \leq 2^{-\lceil \log n \rceil}$.
- Second, for any i such that at least one of balls of the pruned sequence $(\sigma_i^l)_{l \geq l_i}$ is covered but not the last one. We define $\lambda_i \geq l_i$ to be the smallest index such that for all $l \geq \lambda_i$ σ_i^l is not covered. Let γ_i be an arbitrary element of Γ that covers $\sigma_i^{\lambda_i - 1}$. We define $\phi(\gamma_i) = \sigma_i^{\lambda_i}$.
- Last, for any element in Γ_1 that is still unmatched, we extend ϕ to form an arbitrary one-to-one matching between the remaining elements $\gamma \in \Gamma_1$ and all $\sigma_i^{l_i}$, where i is such that none of the balls in the pruned sequence $(\sigma_i^l)_{l \geq l_i}$ are covered.

Note that the second item of Lemma D.4 guarantees that if $\gamma \in \Gamma$ covers a ball of a pruned sequence, it cannot cover a ball of another pruned sequence (as those balls are disjoints): this ensures that our definition of ϕ is consistent and that ϕ is one-to-one. We can now verify it satisfies the three desired properties.

Property 1. For any $\gamma, \gamma' \in \Gamma_1$, let i such that $\phi(\gamma)$ is a ball of the pruned sequence $(\sigma_i^l)_{l \geq l_i}$, and let i' such that $\phi(\gamma')$ is a ball of the pruned sequence $(\sigma_{i'}^l)_{l \geq l_{i'}}$. By construction of ϕ we have $i \neq i'$ and therefore Lemma D.4 ensures that $\phi(\gamma) \cap \phi(\gamma') = \emptyset$.

Property 2. For any $\gamma \in \Gamma_1$, $\phi(\gamma)$ is not covered by Γ by construction.

Property 3. Fix a $\gamma \in \Gamma_1$. We distinguish two cases, based on whether $\phi(\gamma)$ was defined at the second or last step of the procedure (it cannot be defined at the first, as γ involved there are in Γ_0). We aim at showing $\operatorname{Value}(B_\gamma) \leq \operatorname{Value}(\phi(\gamma)) + \theta$.

- If $\phi(\gamma)$ is defined in the last step, then it is of the form $\sigma_i^{l_i}$, and Lemma D.4 ensures that $\operatorname{Value}(\sigma_i^{l_i}) \geq M_{\operatorname{Value}} - \theta$. Combined with the fact that B_γ is available at the end of the algorithm (and therefore by definition of M_{Value}, $\operatorname{Value}(B_\gamma) \leq M_{\operatorname{Value}}$), we obtain directly $\operatorname{Value}(B_\gamma) \leq \operatorname{Value}(\phi(\gamma)) + \theta$.
- Otherwise, $\phi(\gamma)$ is defined in the second step, and $\phi(\gamma) = \sigma_i^{\lambda_i}$. The proof follows a structure similar to the one of Lemma D.4. We will show that there exists a ball $B(x, 2^{-\operatorname{level}(x_i^{\lambda_i})})$ in $\mathcal{B}$, that contains B_γ and is a child of $\sigma_i^{\lambda_i - 1}$. In that case, since the algorithm opted for $\sigma_i^{\lambda_i}$ over $B(x, 2^{-\operatorname{level}(x_i^{\lambda_i})})$, we get $\operatorname{Value}(\sigma_i^{\lambda_i}) \geq \operatorname{Value}(B(x, 2^{-\operatorname{level}(x_i^{\lambda_i})})) - \theta$. Since this ball contains B_γ, it has greater value, which concludes the proof of Property 3. Therefore, we only need to show the existence of such a ball. Note that we don't actually need that B_γ is contained in it, only that its value is greater than $\operatorname{Value}(B_\gamma)$: we will only show this simpler property.

  Let x_γ be the center of B_γ. We start by proving that $\operatorname{level}(x_\gamma) \geq \operatorname{level}(x_i^{\lambda_i}) + 1$. Assume by contradiction that $\operatorname{level}(x_\gamma) \leq \operatorname{level}(x_i^{\lambda_i})$. We are going to prove that in that case, x_γ is too close to the center c_i and therefore is not available at the end of the algorithm, contradicting the definition of B_γ.

  First, by definition of B_γ we know that $\operatorname{dist}(x_\gamma, \gamma) \leq 2^{-\operatorname{level}(x_\gamma)}/2$. Second, since γ is an element covering $\sigma_i^{\lambda_i - 1}$, it holds that $\operatorname{dist}(\gamma, x_i^{\lambda_i - 1}) \leq 2^{-\operatorname{level}(x_i^{\lambda_i - 1})} = 2 \cdot 2^{-\operatorname{level}(x_i^{\lambda_i})} \leq 2 \cdot 2^{-\operatorname{level}(x_\gamma)}$. Third applying Lemma D.6 we get $\operatorname{dist}(x_i^{\lambda_i - 1}, c_i) \leq 20 \cdot 2^{-\operatorname{level}(x_i^{\lambda_i - 1})} \leq 40 \cdot 2^{-\operatorname{level}(x_\gamma)}$. Combining these three inequalities using the triangle inequality we obtain

  $$\begin{aligned} \operatorname{dist}(x_\gamma, c_i) &\leq \operatorname{dist}(x_\gamma, \gamma) + \operatorname{dist}(\gamma, x_i^{\lambda_i - 1}) + \operatorname{dist}(x_i^{\lambda_i - 1}, c_i) \\ &\leq (0.5 + 2 + 40) \cdot 2^{-\operatorname{level}(x_\gamma)} \\ &\leq 100 \cdot 2^{-\operatorname{level}(x_\gamma)}. \end{aligned}$$

  Therefore B_γ is not available at the end of the algorithm, contradicting the definition of B_γ. This finalize the proof of the inequality $\operatorname{level}(x_\gamma) \geq \operatorname{level}(x_i^{\lambda_i}) + 1$.

We now identify the desired child of $\sigma_i^{\lambda_i-1}$. Let x be a net point of level $\text{level}(x_i^{\lambda_i})$ such that $\text{dist}(x, x_\gamma) \leq 2^{-\text{level}(x_i^{\lambda_i})}/2$ (such a point exist by the covering property of nets). We show that $B(x, 2^{-\text{level}(x)})$ is a child of $\sigma_i^{\lambda_i-1}$. Since γ is covering $\sigma_i^{\lambda_i-1}$, it holds that $\text{dist}(x_i^{\lambda_i-1}, \gamma) \leq 2 \cdot 2^{-\text{level}(x_i^{\lambda_i})}$. Additionally, by definition of B_γ we know that $\text{dist}(\gamma, x_\gamma) \leq 2^{-\text{level}(x_\gamma)}/2 \leq 2^{-\text{level}(x_i^{\lambda_i})}/4$. Combining these three inequalities, and using the triangle inequality, we obtain

$$\begin{aligned}
\text{dist}(x_i^{\lambda_i-1}, x) &\leq \text{dist}(x_i^{\lambda_i-1}, \gamma) + \text{dist}(\gamma, x_\gamma) + \text{dist}(x_\gamma, x) \\
&\leq (2 + 0.25 + 0.5) \cdot 2^{-\text{level}(x_i^{\lambda_i})} \\
&\leq 20 \cdot 2^{-\text{level}(x_i^{\lambda_i})} \\
&\leq 10 \cdot 2^{-\text{level}(x_i^{\lambda_i-1})}.
\end{aligned}$$

Therefore the ball $B(x, 2^{-\text{level}(x)})$ is a child of $\sigma_i^{\lambda_i-1}$ and could have been chosen by the algorithm instead of $\sigma_i^{\lambda_i}$. The algorithm selects a child of σ_i^l with the maximum value up to θ. Hence $\text{Value}(\sigma_i^{\lambda_i}) \geq \text{Value}(B(x, 2^{-\text{level}(x)})) - \theta$.

To conclude the proof, it just remains to show that $\text{Value}(B(x, 2^{-\text{level}(x)})) \geq \text{Value}(B_\gamma)$. We will show that for any $p \in P \cap B_\gamma$, the contribution of p to the value of B_γ is lower than its contribution to the value of $B(x, 2^{-\text{level}(x)})$. Let p be a point of $P \cap B_\gamma$, the contribution of p to the value of B_γ is $(2^{-\text{level}(x_\gamma)} - \text{dist}(p, x_\gamma))^z$. We have established that $\text{level}(x_\gamma) \geq \text{level}(x_i^{\lambda_i-1}) + 2 = \text{level}(x) + 1$. Therefore we can bound $2^{-\text{level}(x_\gamma)} \leq 2^{-\text{level}(x)}/2$. Moreover, x_γ is by construction in $B(x, 2^{-\text{level}(x)}/2)$, hence using the triangle inequality we get $\text{dist}(p, x_\gamma) \geq \text{dist}(p, x) - \text{dist}(x, x_\gamma) \geq \text{dist}(p, x) - 2^{-\text{level}(x)}/2$. We can now bound the contribution of p to the value of B_γ

$$\begin{aligned}
(2^{-\text{level}(x_\gamma)} - \text{dist}(p, x_\gamma))^z &\leq (2^{-\text{level}(x)}/2 - (\text{dist}(p, x) - 2^{-\text{level}(x)}/2))^z \\
&\leq (2^{-\text{level}(x)} - \text{dist}(p, x))^z.
\end{aligned}$$

This is precisely the contribution of p to the value of $B(x, 2^{-\text{level}(x)})$. Summing this inequality for all $p \in P \cap B_\gamma$, we obtain $\text{Value}(B(x, 2^{-\text{level}(x)})) \geq \text{Value}(B_\gamma)$, concluding the proof.

□

E. Missing Proofs of Section 4

E.1. Direct application to centralized DP

As an illustration, we sketch an application of previous results to the centralized DP setting. A more formal and general argument will be given later in Theorem 4.3. In centralized DP, when each element is part of at most b sets, standard result show the existence of generalized histograms with additive error $O\left(\sqrt{bD} \cdot \log(m/(\delta\beta))/\varepsilon\right)$. To compute the values of each cell, $D = 1$, there are $m = n^{O(d)}$ many cells, and $b = 2^{O(d)} \log n$. Therefore, those results and Theorem 3.4 yield an (ε, δ)-DP algorithm $\mathcal{A}$ for (k, z)-clustering with multiplicative approximation $O(1)$ and additive error $O\left(k2^{O(d)} \cdot \log(n/(\beta\delta)) \log^{1/2}(n)/\varepsilon\right)$, with probability $1 - \beta$.

This can be combined with the results of Section 2 to get multiplicative approximation $w^*(1 + \alpha)$ and additive error $k^{O(1)} \log^{3/2}(n)/\varepsilon + O\left(k\sqrt{d} \log(k)/\varepsilon\right)$ as follows: first reduce the dimension to $O\left(\log(k)/\alpha^2\right)$, then use algorithm $\mathcal{A}$ and the Laplace mechanism to apply Lemma 2.4. To lift the result back up to the original space, estimate the mean of each cluster as follow: simply estimate the size of each cluster ($D = 1, b = 1, m = k$) and $\sum_{p \in P_i} p$ with a Laplace mechanism (the ℓ_1 sensitivity is 1 in the first case, $\sqrt{d}$ in the second): Property 2.1 is satisfied with $e = \sqrt{d}/\varepsilon$. Lemma 2.2 shows that this gives a solution with the desired cost. Finally, repeat $\log(1/\beta)$ times to boost the probability, which increases the additive error by $\log(1/\beta)$.

For (k, z)-clustering, instead of lifting the clustering using the average of each cluster, we can solve the $(1, z)$-clustering problem on each, which can be done efficiently with additive error $\sqrt{d}\,\text{polylog}(n/\delta)$ (see (Ghazi et al., 2020)). This therefore reduces the multiplicative approximation to $w^*(1 + \alpha)$, as in the k-means case.

E.2. Structured Clusters

We start by providing a formal statement for Lemma 4.2:

Lemma E.1. *There exists a family of set $\mathcal{G} = \{G_1, ..., G_m\}$ efficiently computable, with $m = n^{O(d)}$, and with the following properties. First, any point from $B_d(0,1)$ is part of at most $O(\log n)$ sets G_i. Second, each cluster can be transformed to consist of the union of few G_i. Formally, fix any $1 > \alpha > 0$, and let $\mathcal{C} = \{c_1, ..., c_k\}$ be any set of centers. Then, it is possible to compute a partition $\mathcal{A}$ of $B_d(0,1)$ together with an assignment $a : \mathcal{A} \to \mathcal{C}$ of parts to centers, such that (1) each part of $\mathcal{A}$ is a set from $\mathcal{G}$, (2) $|\mathcal{A}| \leq k \cdot \alpha^{-O(d)} \log(n)$, and (3) for any multiset P,*

$$\left| \operatorname{cost}(P, \mathcal{C}) - \sum_{A \in \mathcal{A}} \sum_{p \in P \cap A} \operatorname{dist}(p, a(A)) \right| \leq \alpha \operatorname{cost}(P, \mathcal{C}) + \frac{9}{\alpha}.$$

Proof. To define the sets $G_1, ...G_m$, we make use of a *hierarchical decomposition* $\mathcal{G} = \left\{\mathcal{G}_1, ..., \mathcal{G}_{\lceil \log(n) \rceil}\right\}$ with the following properties:

1. for all $i = 0, ..., \lceil \log(n) \rceil$, $\mathcal{G}_i$ is a partition of $B_d(0,1)$. Each part A of $\mathcal{G}_i$ is called a cell of level i, denoted $\operatorname{level}(A) = i$, and has diameter at most 2^{-i}.
2. The partition $\mathcal{G}_{i+1}$ is a refinement of the partition $\mathcal{G}_i$, namely every part of $\mathcal{G}_{i+1}$ is strictly contained in one part of $\mathcal{G}_i$. $\mathcal{G}_0$ is defined as $\{B_d(0,1)\}$.
3. for any point $p \in B_d(0,1)$, any level i and any $r \geq 1$, the ball $B_d(p, r2^{-i})$ intersects at most $r^{O(d)}$ many cells of level i.

Such a recursive decomposition can be computed using e.g. the net tree of (Har-Peled & Mendel, 2006). this construction ensures $|\mathcal{G}_i| = 2^{-O(id)}$. We let $\mathcal{G} = \cup \mathcal{G}_i$: this has size $n^{O(d)}$, as desired. For the first property, since the depth of the decomposition is $\lceil \log(n) \rceil + 1$ and each level is a partition, each point from $B_d(0,1)$ appears in exactly $\lceil \log(n) \rceil + 1$ many G_i.

We now show the second property. Fix a set of centers $\mathcal{C} = \{c_1, ..., c_k\}$. We show how to construct a partition $\mathcal{A}$ of $B_d(0,1)$ and assign each part to a center as in the lemma.

For a cell A of level $i+1$, we denote by $\textsc{Parent}(A)$ the unique cell of level i containing A. Let $\ell = \lceil 10/\alpha \rceil$. For any cell A of the decomposition, fix an arbitrary point $v_A \in A$. We call the ℓ-neighborhood of A all cells at the same level as A that intersect with the ball $B\left(v_A, \ell 2^{-\operatorname{level}(A)}\right)$. Note that this includes A.

Let $\mathcal{A}$ be the set of cells A such that their ℓ-neighborhood does not contain a center of $\mathcal{C}$, but the ℓ-neighborhood of $\textsc{Parent}(A)$ does contain one. Add furthermore to $\mathcal{A}$ the cells at level $\lceil \log(n) \rceil$ that contain a center in their ℓ-neighborhood. We verify now that $\mathcal{A}$ has the desired properties: first, that its size is bounded; second, it is a partition; and finally, that each cell can be fully assigned to a center while preserving the cost.

Size of $\mathcal{A}$. We first bound the size of the set $\mathcal{A}$ obtained. For this, we will count for each level how many cells have a center in their ℓ-neighborhood, and how many children each cell has. The product of those two quantities is an upper bound on $|\mathcal{A}|$. First, for a fixed level i and center c^*, the decomposition ensures that there are at most $\ell^{O(d)}$ many cells intersecting $B_d(c^*, 2\ell 2^{-i})$. Note that any cell A that contains c^* in its ℓ-neighborhood must intersect $B_d(c^*, 2\ell 2^{-i})$: indeed, if B is the cell containing c^* this means there is a point in B at distance at most $\ell 2^{-i}$ of v_A. Since the diameter of B is at most 2^{-i}, c^* is at distance at most $\ell 2^{-i} + 2^{-i}$ of v_A, and therefore A intersects with the ball $B_d(c^*, 2\ell 2^{-i})$.

Now, using the third property of decomposition, any cell A of level i is the parent of at most $2^{O(d)}$ many cells. Indeed, all those cells are included in A: this means they are contained in a ball of radius 2^{-i}, and since they are at level $i+1$, third property ensures that there are at most $2^{O(d)}$ many of them. Therefore, at any level, at most $k \cdot \ell^{O(d)} = k \cdot \alpha^{-O(d)}$ cells A are added, as $N^\ell(\textsc{Parent}(A))$ must contain one center. Hence, $|\mathcal{A}| \leq k \cdot \alpha^{-O(d)} \log(n)$.

$\mathcal{A}$ is a partition of $B_d(0,1)$. Now, we show that $\mathcal{A}$ is a partition of $B_d(0,1)$. For this, we observe that if a cell A contains a center in its ℓ-neighborhood, then $\textsc{Parent}(A)$ also contains one. First, we argue that the cells in $\mathcal{A}$ cover the whole unit ball. For a point p in the ball, consider the sequence of parts containing it. There are two cases: Either the smallest part at level $\lceil \log(n) \rceil$ contains a center in its ℓ-neighborhood, and, thus, it is added to $\mathcal{A}$; or it does not, and then the largest part of

the sequence that does not contain a center in its ℓ-neighborhood is added to $\mathcal{A}$. Note that such a largest part must exist, as the cell of level 0 contains all centers.

Next we argue that all cells in $\mathcal{A}$ are disjoint. Consider two intersecting cells A and B in $\mathcal{A}$. By property 2 of the decomposition, it must be that one is included in the other: assume wlog that $A \subset B$ (therefore $\text{level}(A) > \text{level}(B)$). Note that if A is on level $\log(n)$ and has a center in its ℓ-neighborhood, then trivially $N^\ell(\textsc{Parent}(A))$ contains a center. Thus, for all $A \in \mathcal{A}$ it holds that $N^\ell(\textsc{Parent}(A))$ contains a center. But $\textsc{Parent}(A) \subseteq B$: therefore, $N^\ell(B)$ contains a center, which implies that $B \notin \mathcal{A}$. Thus, A and B cannot be both in $\mathcal{A}$, which concludes the proof that $\mathcal{A}$ is a partition of $B_d(0,1)$.

Assigning cells to center. We can now define c as follows: for each cell $A \in \mathcal{A}$, we define $a(A) = \text{argmin}_i \, \text{dist}(v_A, c_i)$ to be the closest point from $\mathcal{C}$ to v_A.

We first define an assignment for a cell $A \in \mathcal{A}$ at level $\log(n)$. In this case, triangle inequality ensures that paying the diameter of A for each point allows to serve all points in A with the same center. Formally, let $a(A)$ be the closest center to v_A. For any $p \in A$, we have using Lemma A.1:

$$\begin{aligned}\text{dist}(p, a(A))^2 &\leq (\text{dist}(p, v_A) + \text{dist}(v_A, a(A)))^2 \leq (2\,\text{dist}(p, v_A) + \text{dist}(p, \mathcal{C}))^2 \\ &\leq (1 + \alpha/2)\,\text{dist}(p, \mathcal{C})^2 + (1 + \frac{2}{\alpha}) \cdot 4\,\text{dist}(p, v_A)^2 \\ &\leq (1 + \alpha/2)\,\text{dist}(p, \mathcal{C})^2 + \frac{9}{\alpha} \cdot \frac{1}{n}.\end{aligned}$$

Therefore, summing all points in cells at level $\log(n)$ gives an additive error at most $\frac{9}{\alpha}$.

Now, fix a cell $A \in \mathcal{A}$ at level $> \log(n)$, and a point $p \in A$. Since, by construction of $\mathcal{A}$, the ℓ-neighborhood of A contains at most one center from $\mathcal{C}$, it holds that:

- either p is already assigned to $a(A)$, and then $\text{dist}(p, a(A)) = \text{dist}(p, \mathcal{C})$,
- or the center serving p in $\mathcal{C}$ is outside of the ℓ-neighborhood of A: in particular, $\text{dist}(p, \mathcal{C}) \geq \ell \cdot \text{dist}(p, v_A)$. In that case, we use the modified triangle inequality from Lemma A.1. This yields similarly as above:

$$\begin{aligned}\text{dist}(p, a(A))^2 &\leq (\text{dist}(p, v_A) + \text{dist}(v_A, a(A)))^2 \leq (2\,\text{dist}(p, v_A) + \text{dist}(p, \mathcal{C}))^2 \\ &\leq (1 + \alpha/2)\,\text{dist}(p, \mathcal{C})^2 + (1 + \frac{2}{\alpha}) \cdot 4\,\text{dist}(p, v_A)^2 \\ &\leq (1 + \alpha/2)\,\text{dist}(p, \mathcal{C})^2 + (1 + \frac{2}{\alpha}) \cdot 4 \cdot \frac{1}{\ell^2}\,\text{dist}(p, \mathcal{C})^2 \\ &= (1 + \alpha/2)\,\text{dist}(p, \mathcal{C})^2 + (4 + \frac{8}{\alpha}) \cdot \frac{\alpha^2}{100} \cdot \text{dist}(p, \mathcal{C})^2 \\ &\leq (1 + \alpha)\,\text{dist}(p, \mathcal{C})^2.\end{aligned}$$

Summing over all $p \in P$ and combining with the cells at level $\log(n)$, we conclude that:

$$\sum_{A \in \mathcal{A}} \sum_{p \in P \cap A} \text{dist}(p, a(A))^2 \leq (1 + \alpha)\,\text{cost}(P, \mathcal{C}) + \frac{9}{\alpha}.$$

The other direction is straightforward: since $a(A) \in \mathcal{C}$, we have by definition for any $p \in A$ $\text{dist}(p, \mathcal{C}) \leq \text{dist}(p, a(A))$, and therefore $\text{cost}(P, \mathcal{C}) \leq \sum_{A \in \mathcal{A}} \sum_{p \in P \cap A} \text{dist}(p, a(A))^2$. □

E.3. Main Approximation Result

Lemma E.2. *Fix a privacy model where there exist private generalized bucketized vector summation such that with probability $2/3$ the additive error is $A(m, b, D)$, where m is the number of sets, b the maximal number of set containing any given element, and D the dimension of the image of the f_i. Assume that the error A is non-decreasing.*

Then, there is a private algorithm for k-means of points in $\mathbb{R}^d$ that, with probability $1 - \beta$, computes a solution with multiplicative approximation $w^*(1 + \alpha)$ and additive error $\left(A(m, k^{O_\alpha(1)}b, 1) + A(m, b, d)\right) \cdot k^{O_\alpha(1)}\,\text{polylog}(n)\log(1/\beta)$, with $m = n^{O_\alpha(\log(k))}$ and $b = \log(n)$.

There is also a private algorithm for (k,z)-clustering of points in $\mathbb{R}^d$ that, with probability $2/3$, computes a solution with multiplicative approximation $w^*(2^z+\alpha)$ and additive error $\big(A(m,k^{O_\alpha(1)}b,1)+A(m,b,d)\big)\cdot k^{O_\alpha(1)}\operatorname{polylog}(n)$.

Proof. This theorem combines all previous results.

We first describe and analyze our private algorithm designed for low-dimensions $\hat{d}=O_\alpha(\log k)$. To use Algorithm 1, one needs to estimate the value of each balls. There are at most $m_1=n^{O(\hat{d})}$ of them, and each input point is in $b_1=2^{O(\hat{d})}\log n$ many balls. This can be done with a summation algorithm, with additive error $A(m_1,b_1,1)$ on the values. Therefore, Theorem 3.2 shows that this implementation of Algorithm 1 computes a solution with multiplicative approximation $O(1)$ and additive error $kA(m_1,b_1,1)$.

Then, one can apply Lemma 4.2 to get structured cluster, with same multiplicative approximation, and additive error increased by $O(1/\alpha)$. We have the following guarantee: each cluster is described with $k\alpha^{-\hat{d}}\log(n)$ many sets of $\mathcal{G}$, each element appears in at most $\log n$ sets of $\mathcal{G}$, and $|\mathcal{G}|=n^{O_\alpha(\log k)}$. Therefore, using the summation algorithm, one can compute the size of each cluster up to an additive error $e_1=A(|\mathcal{G}|,\log n,1)\cdot\alpha^{-\hat{d}}\log(n)$. Similarly, one can compute $\textsc{Sum}_i$ for each cluster i up to an additive error $A(|\mathcal{G}|,\log n,d)$. Therefore, Property 2.1 is satisfied with an error $e=A(|\mathcal{G}|,\log n,d)$.

This is enough to boost the approximation ratio using Lemma 2.4, and get a multiplicative approximation w^* in the space $\mathbb{R}^{\hat{d}}$. The additive error increases to $k'A(m_1,b_1,1)+ke_1$, for $k'=k^{O(1)}\log(n)$.

Now, we show how to use this algorithm as a subroutine, to solve the problem in large dimension d. The algorithm starts by applying the dimension reduction of Lemma 2.2, to reduce the dimension to $\hat{d}=O_\alpha(\log k)$. Then, it computes a clustering using the previous argument, and turns the result into a structured partition using Lemma E.1. Using generalized summation algorithms, the algorithm computes n_i, $\textsc{Sum}_i$ and $\textsc{SumNorm}_i$ that satisfy Property 2.1. The additive error in the estimation is $e:=A(|\mathcal{G}|,\log n,1)\cdot\alpha^{-\hat{d}}\operatorname{polylog}(n)$ (note crucially that the exponent of α is still $\hat{d}$, as the clusters are structured in $\mathbb{R}^{\hat{d}}$).

We conclude the theorem for k-means by plugging those estimations in Lemma 2.2 for dimension reduction and Corollary 2.6 to boost the probabilities: with probability $1-\beta$, the total additive error is

$$\begin{aligned}
&\underbrace{k'A(m_1,b_1,1)}_{\text{Compute initial solution}}+\underbrace{A(|\mathcal{G}|,\log n,1)\cdot k\cdot\alpha^{-\hat{d}}\log(n)}_{\text{Boost approx.}}+\underbrace{A(|\mathcal{G}|,\log n,d)\cdot\alpha^{-\hat{d}}\operatorname{polylog}(n)}_{\text{Lift up the results and boost proba.}}\\
&\le\Big(A(n^{O_\alpha(\log k)},k^{O_\alpha(1)}\log(n),1)+A(n^{O_\alpha(\log k)},\log n,1)+A(n^{O_\alpha(\log k)},\log n,d)\Big)\cdot k^{O_\alpha(1)}\operatorname{polylog}(n)\\
&\le\Big(A(n^{O_\alpha(\log k)},k^{O_\alpha(1)}\log n,1)+A(n^{O_\alpha(\log k)},\log n,d)\Big)\cdot k^{O_\alpha(1)}\operatorname{polylog}(n).
\end{aligned}$$

For (k,z)-clustering, the same holds, although we cannot boost the success probability, and the multiplicative approximation is $(2^z+\alpha)w^*$. □

Corollary 4.4. *There are algorithms for k-means with multiplicative approximation $w^*(1+\alpha)$ that achieve with probability at least $1-\beta$:*

- *additive error $\sqrt{nD}\cdot k^{O_\alpha(1)}/\varepsilon\cdot\operatorname{polylog}(n)\cdot\log(1/\beta)$ in the local (ε,δ)-DP model with one round of communication,*
- *additive error $\sqrt{D}\cdot k^{O_\alpha(1)}/\varepsilon\cdot\operatorname{polylog}(nD/\delta)\cdot\log(1/\beta)$ in the shuffle (ε,δ)-DP model with one round of communication,*
- *additive error $\sqrt{D}\cdot k^{O_\alpha(1)}/\varepsilon\cdot\operatorname{polylog}(n)\cdot\log^{1.5}(DT)\log\log(T)\sqrt{\log(1/\delta)}\cdot\log(1/\beta)$ in (ε,δ)-DP under continual observation.*

The same additive guarantee hold for (k,z)-clustering, with multiplicative approximation $w^*(2^z+\alpha)$ and probability $2/3$.

Proof. The guarantee for the local model stem from Lemma 28 in (Chang et al., 2021) combined with Lemma B.2, which gives a generalized bucketized summation algorithm with desired guarantee (even for $\delta=0$): the additive error is with probability $2/3$ $A(m,b,D)=\frac{b\sqrt{nD\log(m)}}{\varepsilon}$. For the (ε,δ)-shuffle model, this is Theorem 33 of (Chang et al., 2021): the additive error is with probability $2/3$ $A(m,b,D)=\frac{b\sqrt{D}}{\varepsilon}\cdot\operatorname{polylog}(\frac{mD}{\delta})$ (with an unspecified polylog).

Under continual observation, we give in Lemma B.3 a generalized counting mechanism that has with probability $2/3$, at all time step simultaneously, an additive error $O\left(b\sqrt{D}\cdot\varepsilon^{-1}\log(DT)\sqrt{\log(mDT/\beta)\log(b\log(T)/\delta)}\right)$. Plugging in the

values of b, m and using Theorem 4.3 concludes. □

F. Missing Proofs of Section 5

F.1. Centralized DP

Lemma F.1. *Algorithm 1 implemented with the exponential mechanism computes a solution to (k, z)-clustering with multiplicative approximation $O(1)$ and additive error $\frac{k\sqrt{d}\,\mathrm{polylog}(n/\delta)}{\varepsilon'}$. Furthermore, the running time is $k^{O(1)} \cdot n \log^2 n$.*

Proof. For the initial choice of the sequence (line 4 of Algorithm 1), the exponential mechanism takes a decision out of $n^{O(d)}$ many balls: therefore, the value of the chosen ball deviates from the optimum by an additive error $O\left(\frac{d\log(n/\beta)}{\varepsilon'}\right)$, with probability $1-\beta$. Each choice made in the while loop line 5 takes a decision out of the children of the current ball, and there are at most $2^{O(d)}$ of them: therefore, the additive error is $O\left(\frac{d\log(1/\beta)}{\varepsilon'}\right)$, with probability $1-\beta$.

In total, $k\log n$ choices are made: therefore, taking $\beta = 1/(3k\log n)$ ensures that all errors are bounded by $O\left(\frac{d\log(n/\beta)}{\varepsilon'}\right)$ with probability $2/3$. Combined with Theorem 3.4, this shows that the additive error of Algorithm 1 is $O\left(\frac{kd\log(n/\beta)}{\varepsilon'}\right)$.

Dimension reduction reduces d to $\hat{d} = O(\log(k))$ while preserving a multiplicative approximation $O(1)$, and the additive error induced by lifting the clustering up in the original space with Lemma 2.2 is $O\left(\frac{\mathrm{polylog}(n)k\sqrt{d}}{\varepsilon'}\right)$. Indeed, one can estimate the size of each cluster and $\sum_{p\in P_i} p$ with the Gaussian mechanism: in the both cases, the ℓ_2 sensitivity is $O(1)$. The Gaussian mechanism ensures that, for each of those estimates, the additive error is bounded by $\sqrt{d}/\varepsilon'\log(1/\beta)$ with probability $1-\beta$. Therefore, a union-bound over all $2k$ estimates shows that Property 2.1 is satisfied with $e = \sqrt{d}\log(k)/\varepsilon'$. This concludes the approximation guarantee.

The running time of a naive implementation of this algorithm is $n^{\hat{d}}$, to run the exponential mechanism. However, note that only $2^{\hat{d}}n\log n$ balls are non-empty (since at each level, each point is in at most $2^{\hat{d}}$ many balls). All empty balls have same value, equal to 0. Therefore, the exponential mechanism can be implemented in time $2^{\hat{d}}n\log n = k^{O(1)}n\log n$. As the algorithm makes $O(k\log n)$ calls to the exponential mechanism, this concludes the complexity analysis. □

To show that the procedure is (ε,δ)-DP, we follow the analysis of (Gupta et al., 2010) for weighted set cover (adapted by (Chaturvedi et al., 2021) for k-means): in our language, they analyzed the exponential mechanism for a process that selects iteratively balls with largest value, and then remove all points of the selected ball. Our process is more intricate, as it combines selecting the heads with constructing the sequence. We manage nonetheless to adapt their proof in the following lemma:

Lemma F.2. *Algorithm 1 when implemented using the exponential mechanism with parameter $\varepsilon' = \frac{\varepsilon}{4\log(n/\delta)}$, is (ε,δ)-DP.*

Proof. We follow the proof of (Gupta et al., 2010) for set cover. Our two step process incurs some complication: analyzing the selection of the first ball of each sequence σ_i is similar to (Gupta et al., 2010), but in our case it is interlaced with the recursive greedy choices, which makes thing more technical.

We consider the outcome of the algorithm to be the k sequences of balls $\sigma_1, ..., \sigma_k$ (instead of merely the k centers), and we will show that this is (ε,δ)-DP. For this, we will fix a given set of sequences $C = (\sigma_1, ..., \sigma_k)$ and will compare the probability that the outcome of $\mathcal{A}$ is C on two neighboring inputs X and X'. Note that fixing a sequence C also fixes the center chosen by the algorithm: the i-th center is the (geometric) center of the last ball in σ_i.

We start with some notations. We write $\mathcal{A}(X)_i^j$ to be the j-th choice made by the algorithm in the i-th sequence, on input X. We denote $\{\sigma, <i, <j\}$ the event $\mathcal{A}(X)_{i'}^{j'} = \sigma_i^j$ for all i', j' such that either $i' < i$ or $i' = i$ and $j' < j$. We write $\{\sigma, \le i\}$ for $\{\sigma, <i, <\infty\}$. Furthermore, for a ball B, we write $C(B)$ the children of B (see Section 3 for the definition of children, forbidden and available)

For any ball $B \in \mathcal{B}$ and dataset X, we write $\mathrm{Value}_i(B, X) = -\infty$ when B is forbidden by some center among the first $i-1$ of C; and $\mathrm{Value}_i(B, X) := \mathrm{Value}(B, X)$ otherwise (when B is still available after choosing the first $i-1$ centers from C).

Let X and X' be two neighboring datasets, with symmetric difference $X \Delta X' = \{p\}$. We aim at bounding $\frac{\Pr[\mathcal{A}(X)=C]}{\Pr[\mathcal{A}(X')=C]}$.

There are two different choices in the algorithm: the choice of the first elements of each sequence, and then the recursive construction of the sequences themselves.

$$\Pr\left[\mathcal{A}(X)_i^1 = \sigma_i^1 \mid \{\sigma, \leq i\}\right] = \frac{\exp(\varepsilon' \operatorname{Value}(\sigma_i^1, X))}{\sum_{B \in \mathcal{B}} \exp(\varepsilon' \operatorname{Value}_i(B, X))},$$
$$\Pr\left[\mathcal{A}(X)_i^j = \sigma_i^j \mid \{\sigma, < i, < j\}\right] = \frac{\exp(\varepsilon' \operatorname{Value}(\sigma_i^j, X))}{\sum_{c \in C(\sigma_i^{j-1})} \exp(\varepsilon' \operatorname{Value}(c, X))}.$$

Note that, if there is i such that σ_i^1 is not available at the i-th step, then $\Pr[\mathcal{A}(X) = C] = \Pr[\mathcal{A}(X') = C] = 0$; and similarly if there are some i, j such that σ_i^j is not a children of σ_i^{j-1}, namely $\sigma_i^j \notin C(\sigma_i^{j-1})$. Therefore, we only need to focus on admissible sequences, where the previous cases cannot happen. This ensures that one input point p can appear only in $\log n$ many σ_i^j and $C(\sigma_i^j)$, one per level of the ball hierarchy. Indeed, if $p \in \sigma_i^j$, then all balls containing p at the level of σ_i^j and the level below are forbidden by the algorithm, and thus in any subsequent admissible sequence p cannot appear in another ball at those levels; since there are $\log n$ levels, p appears in at most $\log n$ balls or children of balls in an admissible sequence of balls.

We now analyse the ratio of the probabilities. We write:

$$\frac{\Pr[\mathcal{A}(X) = C]}{\Pr[\mathcal{A}(X') = C]}$$
$$= \prod_{i=1}^{k} \frac{\exp(\varepsilon' \operatorname{Value}_i(\sigma_i^1, X))}{\exp(\varepsilon' \operatorname{Value}(\sigma_i^1, X'))} \tag{9}$$
$$\cdot \prod_{i=1}^{k} \frac{\sum_B \exp(\varepsilon \operatorname{Value}_i(B, X'))}{\sum_B \exp(\varepsilon \operatorname{Value}_i(B, X))} \tag{10}$$
$$\cdot \prod_{i=1}^{k} \prod_{j=1}^{\operatorname{len}(\sigma_i)} \frac{\exp(\varepsilon' \operatorname{Value}(\sigma_i^j, X))}{\exp(\varepsilon' \operatorname{Value}(\sigma_i^j, X'))} \tag{11}$$
$$\cdot \prod_{i=1}^{k} \prod_{j=1}^{\operatorname{len}(\sigma_i)} \frac{\sum_{B \in C(\sigma_i^{j-1})} \exp(\varepsilon' \operatorname{Value}(B, X'))}{\sum_{B \in C(\sigma_i^{j-1})} \exp(\varepsilon' \operatorname{Value}(B, X))} \tag{12}$$

We start by bounding the two easier Equation (11) and Equation (12). Those are the standard terms from the exponential mechanism: for twe neighboring dataset differing on a single point p, the value of any ball changes by at most Δ, and does so only when p is part of the ball. So, when p is part of the ball σ_i^j, we have: $\frac{\exp(\varepsilon' \operatorname{Value}(\sigma_i^j, X))}{\exp(\varepsilon' \operatorname{Value}(\sigma_i^j, X'))} \leq \exp(\varepsilon' \Delta)$, and when p is part of a ball among $C(\sigma_i^j)$, we have: $\frac{\sum_{B \in C(\sigma_i^{j-1})} \exp(\varepsilon' \operatorname{Value}(B, X'))}{\sum_{B \in C(\sigma_i^{j-1})} \exp(\varepsilon' \operatorname{Value}(B, X))} \leq \exp(\varepsilon' \Delta)$.

As mentioned before, is an admissible sequence p is part of at most $\log n$ σ_i^j, and $\log n$ $C(\sigma_i^j)$: therefore, the products of Equation (11) and Equation (12) is bounded by $\exp(2 \log n \cdot \varepsilon' \Delta)$.

We can now focus on the two more intricate terms, Equation (9) and Equation (10). In the case where $X = X' \cup \{p\}$, then for any ball B and any i, $\operatorname{Value}_i(B, X') \leq \operatorname{Value}_i(B, X)$, and therefore term in Equation (10) is at most one. On the other hand, $\operatorname{Value}(B, X) \leq \operatorname{Value}_i(B, X') + \Delta$ when B contains the point p, and $\operatorname{Value}_i(B, X) = \operatorname{Value}(B, X')$ if B doesn't contain p. Since at most $\log n$ of the σ_i^1 contain p, the term in Equation (9) is at most $\exp(\log n \cdot \varepsilon)$. Therefore,

$$\Pr[\mathcal{A}(X) = C] \leq \exp(\log n \cdot \varepsilon) \Pr[\mathcal{A}(X') = C].$$

The second case, where $X' = X \cup \{p\}$, is more intricate. There, Equation (9) is at most one, and we bound Equation (10) as follows: first, we write for simplicity $\pi_i(B) := \Pr[\mathcal{A}(X)_i^j = x \mid \{\sigma, < i, < j\}] = \frac{\exp(\varepsilon' \operatorname{Value}_i(B, X))}{\sum_{B'} \exp(\varepsilon' \operatorname{Value}_i(B', X'))}$.

$$
\begin{aligned}
&\frac{\sum_{B'} \exp(\varepsilon \operatorname{Value}_i(B', X'))}{\sum_{B'} \exp(\varepsilon' \operatorname{Value}_i(B', X))} \\
=&\sum_{B'} \pi_i(B') \exp\left(\varepsilon'(\operatorname{Value}_i(B', X') - \operatorname{Value}_i(B', X))\right) \\
\leq&\sum_{B'} \pi_i(B')(1 + 2\varepsilon'(\operatorname{Value}_i(B', X') - \operatorname{Value}_i(B', X))/\Delta) \\
\leq&1 + 2\varepsilon' \sum_{B'} \pi_i(B')(\operatorname{Value}_i(B', X') - \operatorname{Value}_i(B', X)) \\
\leq&\exp\left(2\varepsilon' \sum_{B'} \pi_i(B')(\operatorname{Value}_i(B', X') - \operatorname{Value}_i(B', X))\right) \\
\leq&\exp\left(2\varepsilon'\Delta \sum_{B' \text{ s.t. } p \in B'} \pi_i(B')\right).
\end{aligned}
$$

The sum in the exponent corresponds to the probability to chose a ball B' that contains p, after having made the choices $\{\sigma, < i, < j\}$. We write π_i this probability.

Therefore, combining with the bounds on Equation (11) and Equation (12), we have:

$$
\frac{\Pr[\mathcal{A}(x) = C]}{\Pr[\mathcal{A}(X') = C]} \leq \exp\left(2\varepsilon'\Delta \sum_{i=1}^{k} \pi_i\right) \cdot \exp\left(2 \log n \cdot \varepsilon'\Delta\right).
$$

The analysis of (Gupta et al., 2010) works black box from this point. They show that the above equations implies that, for any set of possible outcomes $\mathcal{C}$, it holds that

$$
\Pr[\mathcal{A}(x) \in \mathcal{C}] \leq \exp(2\varepsilon'\Delta \log(1/\delta) + 2 \log n \cdot \varepsilon'\Delta) \Pr[\mathcal{A}(X') = \mathcal{C}] + \delta,
$$

which, with our choice of ε', concludes the proof.

□

F.2. Parallel algorithm for MPC

Lemma 5.1. *Suppose there is a constant $c_{\mathcal{A}}$ such that, for any level ℓ, (1) the distance between two distinct centers of C_ℓ is greater than $3 \cdot 2^{-\ell}$; and (2) for any center $c \in C_\ell$, the value of the ball $B(c, 2^{-\ell})$ is greater (up to an additive error θ) than the value of any ball of $\mathcal{B}_\ell$ available at scale $c_{\mathcal{A}}$ from C_ℓ. Then,* $\operatorname{cost}(P, C) \leq O(1) \cdot \textsc{opt}_{k,z} + O(k\theta)$.

Proof. The proof of this lemma is grounded in the analysis of Algorithm 1, which we conducted in Appendix D. We define $\mathcal{A}(c_{\mathcal{A}}, C)$,$\Gamma, \Gamma_0$, Γ_1, and the B_γ's for $\gamma \in \Gamma_1$ as detailed in Appendix D. Note that $\mathcal{A}(c_{\mathcal{A}}, C)$ is the set of balls of $\mathcal{B}$ available at scale $c_{\mathcal{A}}$ from C. Using Lemma D.2 and Equation (8), we can reduce the task of proving Lemma 5.1 to bounding $\sum_{\gamma \in \Gamma_1} \operatorname{Value} B_\gamma$. Note that the constants appearing in Lemma D.2 depend on $c_{\mathcal{A}}$, but we consider $c_{\mathcal{A}}$ as a fixed constant that does not appear in the $O(\cdot)$ notation.

In the proof of Theorem 3.4, to $\sum_{\gamma \in \Gamma_1} \operatorname{Value} B_\gamma$, the proof defines a function ϕ that maps the center of Γ_1 to balls of $\mathcal{B}$ such that for all $\gamma \in \Gamma_1$:

1. for all $\gamma' \in \Gamma_1$ with $\gamma \neq \gamma'$, $\phi(\gamma) \cap \phi(\gamma') = \emptyset$,
2. $\phi(\gamma)$ is not covered by Γ,
3. The value of B_γ is less than $\operatorname{Value}(\phi(\gamma)) + \theta$.

Given such a mapping, the proof of Theorem 3.4 shows $\sum_{\gamma\in\Gamma_1} \text{Value}\, B_\gamma \le \text{cost}(P,\Gamma)$. The same proof can be applied here: therefore, it is enough to construct the mapping ϕ. For this, we use the following procedure.

At the beginning of the procedure, the set Φ consists of all balls that can be matched to a γ: this is all the balls of the form $B(c,2^{-\ell})$ with $c\in C_\ell$, for all ℓ. Until all $\gamma\in\Gamma_1$ have been matched, repeat the following process. Let $\gamma\in\Gamma_1$ be such that the radius of B_γ is minimal, among the centers of Γ_1 that have not yet been matched. γ is matched to an arbitrary ball $B(c,2^{-\ell})\in\Phi$ (ensuring that the radius of B_γ is also $2^{-\ell}$ and $c\in C_\ell$). All balls intersecting $B(c,2^{-\ell})$ or containing γ are removed from Φ; repeat the procedure for the next γ.

We begin by demonstrating that this procedure is well defined, namely that there is always a ball $B(c,2^{-\ell})$ in Φ. Once a ball $B(c,2^{-\ell})$ is selected, at most 2 balls per level $\ell'\le\ell$ are eliminated. This is due to condition (1) of the lemma, which ensures that the distance between two distinct centers of $C_{\ell'}$ is greater than $3\cdot 2^{-\ell'}$. This condition has two implications: first, the balls $B(c',2^{-\ell'})$ with $c'\in C_{\ell'}$ are disjoint, and therefore at most one contains γ. Second, a single ball $B(c',2^{-\ell'})$ at level ℓ' can intersect $B(c,2^{-\ell})$. Indeed, assuming by contradiction that two balls $B(c'_1,2^{-\ell'})$ and $B(c'_2,2^{-\ell'})$ intersect $B(c,2^{-\ell})$, we have, by the triangle inequality, $\text{dist}(c'_1,c'_2)\le 2\cdot 2^{-\ell'}+2^{-\ell}\le 3\cdot 2^{-\ell'}$ because we assumed $\ell'\le\ell$.

The procedure defines ϕ by increasing order of the radii of the B_γ's. Therefore, when the center γ' with $B_{\gamma'}=B(c',2^{-\ell'})$ is considered in the procedure, all centers γ previously matched had $B_\gamma=B(c,2^{-\ell})$ with $\ell'\le\ell$: the previous discussion implies that each of those removed at most two balls with radius $2^{-\ell'}$ from Φ.

By construction of our algorithm, each C_ℓ contains $2k$ centers: since there are k centers in Γ, the matching ϕ is well defined. We can now verify that this matching satisfies the three desired properties.

1. The procedure guarantees that for all $\gamma,\gamma'\in\Gamma_1$, $\phi(\gamma)\cap\phi(\gamma')=\emptyset$ because when a ball is selected, any ball intersecting it is removed from Φ.

2. Let $\gamma\in\Gamma$. Assume by contradiction that there exists $\gamma'\in\Gamma$ covering $\phi(\gamma)$, and let $B_\gamma=B(z,2^{-\ell})$, $\phi(\gamma)=B(c,2^{-\ell})$ (the procedure ensures that $\phi(\gamma)$ has same radius as B_γ), and $B_{\gamma'}=B(z',2^{-\ell'})$. By definition of covering, $\gamma'\in\phi(\gamma)$. First, in the case where $\ell'>\ell$, γ' is processed before γ in the procedure: all the balls containing γ' have been removed from Φ when $\phi(\gamma')$ is defined, and in particular $\phi(\gamma)$. This yields to a contradiction. In the other case, when $\ell'\le\ell$, we note the following inequality: $\text{dist}(c,\gamma')\le 2^{-\ell}\le 2^{-\ell'}$. Moreover by definition of $B_{\gamma'}$, $\text{dist}(\gamma',z')\le 2^{-\ell'}/2$. Using the triangle inequality we obtain $\text{dist}(c,z')\le 1.5\cdot 2^{-\ell'}\le c_{\mathcal{A}}\cdot 2^{-\ell'}$, this implies that $B_{\gamma'}$ is not available at scale $c_{\mathcal{A}}$ from C, which contradicts the definition of $B_{\gamma'}$. Those two cases concludes that there is no γ' covering $\phi(\gamma)$.

3. By definition, B_γ is in $\mathcal{A}(c_{\mathcal{A}},C)$, and the condition 3. of the lemma states that the value of the ball $\phi(\gamma)=B(c,2^{-\ell})$ is greater (up to an additive error θ) than the value of any ball of $\mathcal{B}_\ell$ available at scale $c_{\mathcal{A}}$ from C_ℓ: this set contains $\mathcal{A}(c_{\mathcal{A}},C)$, and therefore it contains B_γ, and the third condition holds.

This conclude the proof of Lemma 5.1. □

We describe more formally the greedy algorithm described in the main body. We suppose that machines are organized in a tree structure, where the degree of every node is n^κ – so that the depth of the tree is $1/\kappa$. We also suppose the dimension is reduced to $\hat{d}=O(\log(k))$. The algorithm works as in the centralized case: first solve (k,z)-clustering in this projected space, then lift-up the centers in $\mathbb{R}^d$.

Initially, each possible ball is assigned randomly to one of the leaf (this can be implemented with a simple hash-function). Using standard MPC procedure, each machine can compute the value of each non-empty ball assigned to it.

For a leaf M, let $B(M)$ be the set of non-empty balls assigned to M. For each round $i=0,\dots,1/\kappa$, the following process happens. First, each machine M at height i in the tree runs the greedy algorithm with distance parameter $D_i:=2^{-i}\cdot 32^{1/\kappa}$. Let S_M be the $2k$ balls selected by M: M sends S_M to its parent. For each machine at height $i+1$, let $B(M)$ be the set of balls received by M. This marks the end of the round.

First, the privacy of this algorithm follows directly from composition: at each level of the tree, properties of the exponential mechanism (or, more precisely, the proof of Lemma F.2) ensure that process is (ε,δ)-DP, and there are $1/\kappa=O(1)$ many levels.

We verify that this algorithm can indeed be implemented in MPC when machines have memory $k^{O(1)}n^{\kappa}$. Note there are at most $2^{\hat{d}}n = k^{O(1)}n$ non-empty balls at each level: therefore it is possible to represent all non-empty balls in the memory: each of the $n^{1-\kappa}$ machine has memory $k^{O(1)}n^{\kappa}\operatorname{polylog}(n)$. Assigning uniformly at random the balls to leafs ensures that with high probability they are assigned in a balanced way, and that they fit in memory.

To implement the exponential mechanism, the algorithm do not actually have access to all balls assigned to the machine M – only the non-empty ones. Instead, the algorithm uses the exponential mechanism on the following set: all possible balls, with value 0, and all non-empty balls assigned to M, with their true value. This can be efficiently implemented in time $k^{O(1)}n^{\kappa}\operatorname{polylog}(n)$, and ensures that with probability $1-\beta'$, the ball selected by one exponential mechanism have maximum value up to $\theta = O\left(\frac{\hat{d}\log(n/\beta')}{\varepsilon}\right)$.

The exponential mechanism is used $2k$ times on each machine: choosing $\beta' = \frac{\beta}{2kn^{1-\kappa}}$ allows to do a union bound, and shows that all ball selected have maximum value up to $\theta = O\left(\frac{\hat{d}\log(n/\beta)}{\varepsilon}\right)$.

We will condition all our analysis on the fact that the exponential mechanism selects ball with maximal value, up to an additive error θ. To compute θ, note that the exponential mechanism is used $2k$ times in each machine, each time to select one ball out of $n^{\hat{d}}$ many balls. in total it is used $2kn^{1-\kappa}$ times To analyse this greedy algorithm, we note the following crucial claims:

Claim F.3. *Let M be a machine and M' be its parent. Then, the smallest value in $S_{M'}$ is larger (up to θ) than the smallest value in S_M.*

Proof. The greedy process ensures all balls in S_M are at distance at least $2^{-h(M)}c_{\mathcal{A}} \cdot 2^{-\ell}$ of each other, where $h(M)$ is the height of M in the tree. Therefore, by triangle inequality every ball selected in $S_{M'}$ forbids at most one ball of S_M (due to the exponential decrease of the distance parameter): as a consequence, at any choice made by the greedy on M', at least a ball of S_M is still available. Therefore, the value of selected balls in $S_{M'}$ are only larger than the ones in S_M (up to an additive error θ due to the exponential mechanism). □

Claim F.4. *Let B be a ball in $\mathcal{B}(M)$ forbidden by some $B' \in S_M$ at scale $\sum_{i \le h(M)} D_i$. Then,* $\operatorname{Value}(B) \le \operatorname{Value}(B') + h(M)\theta$.

Proof. Let M_i be the machine at height i such that $B \in \mathcal{B}(M_i)$. Define B_i as follows:

1. if $B_{i-1} \in S_{M_i}$, $B_i = B_{i-1}$
2. if B_{i-1} is available at scale D_i from S_{M_i}: let B_i be an arbitrary ball in S_{M_i}
3. otherwise, B_{i-1} is forbidden by some ball in S_{M_i}. Let B_i be the first ball in the greedy process to forbid B_{i-1}.

Now, we claim that $\operatorname{Value}(B_i) \ge \operatorname{Value}(B) - i\theta$. This is obvious in the first case. In the second case, this is directly due to the greedy procedure: for each choice of the greedy on M_i, the ball B_{i-1} is available, and therefore the ball selected has value at least $\operatorname{Value}(B_{i-1}) - \theta$. In the third case, at the moment B_i is chosen by the greedy procedure on M_i, B_{i-1} is still available and therefore $\operatorname{Value}(B_i) \ge \operatorname{Value}(B_{i-1}) - \theta$.

Since $B' = B_{h(m)}$, this concludes the claim. □

we can now show this implementation of the greedy achieve the guarantees required by Lemma 5.1.

Lemma F.5. *Let C_ℓ be the set of balls selected by the merge-and-reduce process described above after $1/\kappa$ rounds. The balls of C_ℓ are at distance at least $3 \cdot 2^{-\ell}$ from each other, and the value of any ball in C_ℓ is larger (up to θ/κ) than the value of any ball available at scale $2^{1/\kappa+1} \cdot 3$.*

Proof. The first part of the statement follows directly from the greedy merging: at the root of the tree, the ball selected are at distance $3 \cdot 2^{-\ell}$ from each other. For the second, we proceed by induction.

We start with few notations: for a machine M, we let $h(M)$ be the height of M in the tree, and $c(M)$ be the set of children of M. We define inductively $\mathcal{B}_M := \cup_{M' \in c(M)} \mathcal{B}_{M'}$ to be the set of balls covered by machine M (for a leaf M $\mathcal{B}_M$ be the

balls assigned to the leaf M by the random assignment). We define S_M be the set of balls selected by the greedy procedure run on M.

Our inductive claim is that the balls selected in S_M have larger values, up to an additive error $\theta h(M)$, than the balls in $\mathcal{B}_M$ available at scale $\sum_{i\leq h(M)} D_i \cdot 2^{-\ell}$ from S_M. When $h(M) = 0$, this property stems directly from the greedy algorithm.

For $h(M) \geq 1$, we proceed as follows: let B be a ball available at scale $\sum_{i\leq h(M)} D_i \cdot 2^{-\ell}$ from S_M. Let M' be the children of M such that $B \in \mathcal{B}_{M'}$. First, in the case where $B \in S_{M'}$, B is part of the greedy process on machine M: since it is available and not selected, it has smaller value than any ball selected in S_M up to an additive error θ.[2] Second, in the case where $B \notin S_{M'}$: either B is forbidden at scale $D_{h(M')}$ by some ball $B' \in S_{M'}$, or B is available at scale $D_{h(M')}$ from $S_{M'}$.

- First, suppose B is forbidden by some $B' \in S_{M'}$ at scale $\sum_{i\leq h(M')} D_i$. It cannot be that $B' \in S_M$, as otherwise B would be forbidden at scale $\sum_{i\leq h(M)} D_i$ from S_M. Therefore, either B' is forbidden at scale $D_{h(M)}$ from S_M, or it is still available at the end of the greedy on M. In the first case, triangle inequality shows that B is at distance at most $\sum_{i\leq h(M)} D_i$ of S_M, which contradicts the initial condition on B. On the second case, if B' is still available at scale $D_{h(M)}$ from S_M, then the greedy ensures that its value is smaller than that of any ball selected in S_M, up to an additive error θ. Hence, combined with Claim F.4, this shows that the value of B is smaller than any ball of S_M, up to an additive error $h(M)\theta$.
- In the case B is available at scale $\sum_{i\leq h(M')} D_i$ from $S_{M'}$, then we know by induction that its value is less than any selected ball in $S_{M'}$, up to an additive $h(M')\theta$. As shown in Claim F.3, this implies its value is less than any selected ball in S_M as well, up to an additive error $h(M')\theta + \theta = h(M)\theta$, which concludes the inductive claim.

Therefore, at height $2^{1/\kappa}$ (i.e., the root of the merge-and-reduce tree), the balls selected have higher values than the one available at scale $\sum_{i\leq 2^{1/\kappa}} D_i 2^{-\ell}$. As $D_i = 2^{1/\kappa - i}$, this is at most $2^{1/\kappa+1} \cdot 2^{-\ell}$, which concludes the lemma. □

Combining those lemmas proves Theorem 5.2:

Proof of Theorem 5.2. Lemma F.5 shows that, at each level, the set C_ℓ computed by the algorithm satisfies the conditions of Lemma 5.1 (with $c_{\mathcal{A}} = 2^{1/\kappa+1} \cdot 3$ and $\theta = O\left(\frac{\hat{d}\log(n/\beta)}{\varepsilon}\right)$): therefore, $C = \cup C_\ell$ verifies $\mathrm{cost}(P, C) \leq O(1)\mathrm{OPT}_{k,z} + O\left(k \cdot \frac{\hat{d}\log(n/\beta)}{\varepsilon}\right)$).

Furthermore, C has size $O(k \log n)$: it can be aggregated in a single machine. The techniques from Appendix C.1 allows then to reduce the number of centers selected to k, without increasing the error.

Finally, to lift-up the solution in the original space, we use Lemma 2.2: using noisy average, we can estimate the mean of each cluster up to an additive error $O\left(\sqrt{d}/\varepsilon\right)$, which yields a solution with cost $O(1)\mathrm{OPT}_{k,z} + O\left(\mathrm{polylog}(n) \cdot \frac{k\sqrt{d}}{\varepsilon}\right)$.

For the second part of the lemma, we need to boost the approximation ratio using Section 2.2. For this, Lemma 2.4 directly applies, if we can lift up the solution to the original space. For k-means, we can simply compute the noisy average of each cluster – which gives an extra additive error $O(k\sqrt{d}\log(1/\delta)/\varepsilon)$, as in the proof of Lemma F.1.

Recovering the optimal $(1, z)$-clustering in each cluster, using low memory, is more intricate. For this, we apply the techniques presented in Theorem E.6 of (Cohen-Addad et al., 2022a), to get an additive error $(2^{\hat{d}} + k\sqrt{d})\,\mathrm{polylog}(n/\delta)/\varepsilon$.

In both cases, this provides a solution with cost $(1+\alpha)w^*\mathrm{OPT}_{k,z} + (k^{O_\alpha(1)} + k\sqrt{d})\,\mathrm{polylog}(n/\delta)/\varepsilon$. □

G. A note on ε-Differential Privacy

Our results partially extend to pure-DP, i.e. when $\delta = 0$. In that case, we can adapt our algorithm for the optimal multiplicative approximation : in most models, there is a general summation algorithm with additive error essentially worsen by a $\sqrt{b}$ factor, compared to (ε, δ)-DP. For centralized DP, the additive error is $b\sqrt{D}\log(m/\beta)/\varepsilon$ – this is a direct extension of histograms. In local DP, the additive error we presented already worked in the case $\delta = 0$ – as the summation result of (Chang et al., 2021) that we used readily works in this case.

For ε-DP under continual observation, we can apply the second part of our Lemma B.3 to get that an additive error worsen by a $\sqrt{d\log(T)}$ – leading to a (k,z)-clustering with additive error $dk^{O_\alpha(1)}\log^3(n)\log^2(T)\log(1/\beta)$.