Dynamic Constrained Submodular Optimization with Polylogarithmic Update Time

Kiarash Banihashem [*1] Leyla Biabani [*2] Samira Goudarzi [*2] MohammadTaghi Hajiaghayi [*1]
Peyman Jabbarzade [*1] Morteza Monemizadeh [*2]

Abstract

Maximizing a monotone submodular function under cardinality constraint k is a core problem in machine learning and database with many basic applications, including video and data summarization, recommendation systems, feature extraction, exemplar clustering, and coverage problems. We study this classic problem in the fully dynamic model where a stream of insertions and deletions of elements of an underlying ground set is given and the goal is to maintain an approximate solution using a fast update time. A recent paper at NeurIPS'20 by Lattanzi, Mitrovic, Norouzi-Fard, Tarnawski, Zadimoghaddam (Lattanzi et al., 2020a) claims to obtain a dynamic algorithm for this problem with a $(\frac{1}{2} - \epsilon)$ approximation ratio and a query complexity bounded by $\text{poly}(\log(n), \log(k), \epsilon^{-1})$. However, as we explain in this paper, the analysis has some important gaps. Having a dynamic algorithm for the problem with polylogarithmic update time is even more important in light of a recent result by Chen and Peng (Chen & Peng, 2022) at STOC'22 who show a matching lower bound for the problem – any randomized algorithm with a $\frac{1}{2} + \epsilon$ approximation ratio must have an amortized query complexity that is polynomial in n. In this paper, we develop a simpler algorithm for the problem that maintains a $(\frac{1}{2} - \epsilon)$-approximate solution for submodular maximization under cardinality constraint k using a polylogarithmic amortized update time.

[*]Equal contribution [1]Department of Computer Science, University of Maryland, MD, USA [2]Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands. Correspondence to: Kiarash Banihashem <kiarash@umd.edu>, Morteza Monemizadeh <m.monemizadeh@tue.nl>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

The problem of maximizing a monotone submodular function under cardinality constraint k is defined as follows: Let $f : 2^V \to \mathbb{R}^+$ be a non-negative monotone submodular function defined on subsets of a ground set V. Let $k \in \mathbb{N}$ be a parameter. We are asked to return a set X of at most k elements such that $f(X)$ is maximum among all k subsets of V. This problem is at the core of machine learning (Elenberg et al., 2017; Mitrovic et al., 2019; Tohidi et al., 2020), data mining (Wu & Tseng, 2022; Ashkan et al., 2015; Parambath, 2019), and database (Bateni et al., 2019; Salehi et al., 2018) with many basic applications including video and data summarization (Feldman et al., 2018), recommendation systems (Parambath et al., 2018; Ashkan et al., 2015; Parambath, 2019; Benouaret et al., 2019), feature extraction (Bateni et al., 2019), spatial search and map exploration (Wu & Tseng, 2022) exemplar clustering (Salehi et al., 2018; Badanidiyuru et al., 2014), sparse regression (Das & Kempe, 2018; Tsai & Tseng, 2020; Tseng & Mettler, 2017) and coverage problems (Bar-Ilan et al., 2001; Sagnol, 2013), to name a few.

For the submodular maximization problem under cardinality constraint k, the celebrated greedy algorithm due to Fisher, Nemhauser, and Wolsey (Nemhauser et al., 1978) achieves an approximation ratio of $1 - 1/e \approx 0.63$, which is optimal assuming $P \neq NP$. However, this classic algorithm is inefficient when applied to *modern big data* settings, given the unique challenges of working with massive datasets. Motivated by these challenges, in recent years there has been a surge of interest in considering the submodular maximization problem under a variety of computational models such as streaming models (Badanidiyuru et al., 2014; Kazemi et al., 2019) and distributed models (da Ponte Barbosa et al., 2016; Liu & Vondrák, 2019; Kumar et al., 2015; McGregor & Vu, 2019).

Related work. Badanidiyuru, Mirzasoleiman, Karbasi and Krause (Badanidiyuru et al., 2014) were the first to study this problem in the insertion-only streaming model and developed a $(\frac{1}{2} - \epsilon)$-approximate streaming algorithm for this problem using space in $O(k\text{poly}(\log(n), \log(k), \frac{1}{\epsilon}))$.

Recently, Mirzasoleiman, Karbasi, and Krause (Mirzasoleiman et al., 2017) studied this problem in the insertion-deletion streaming model where they obtained a $(\frac{1}{2} - \epsilon)$-approximate algorithm using $O(d^2(k\epsilon^{-1} \log k)^2)$ space and $O(dk\epsilon^{-1} \log k)$ average update time, where d is an upper-bound for the number of deletions that are allowed. A follow-up by Kazemi, Zadimoghaddam, and Karbasi (Kazemi et al., 2018) improved the space complexity and the average update of the first work down to $O(k \log k + d \log^2 k)$ and $O(dk \log^2 k + d \log^3 k)$.

One bottleneck of these two works is the polynomial dependency of their space and time complexities on the number of deletions since it takes too much time to (re)compute the solution after every insertion or deletion. Indeed, if the number of deletions is linear in $n = |V|$ or higher (say, $d = \Omega(n)$), it is better to rerun the offline algorithm (Nemhauser et al., 1978) after every insertion and deletion. Despite their use cases, the above algorithms are unsuitable for many modern applications where data is highly *dynamic*. For these applications such as data subset selection problem (Elhamifar & Kaluza, 2017; Elhamifar, 2019), movie recommendation system (Ohsaka & Matsuoka, 2022; Chen et al., 2017), influence maximization in social networks (Chen et al., 2009; Tong et al., 2017; Zhang et al., 2017), elements are continuously added and deleted, prompting the need for algorithms that can efficiently handle both insertions and deletions at the same time.

Submodular maximization in dynamic model. Motivated by these interests, in this paper, we study the submodular maximization problem under the cardinality constraint k in the *dynamic model* (Bernstein et al., 2021; Hanauer et al., 2022; Bhattacharya et al., 2021; 2020). In this model, we are given a stream of insertions and deletions of elements of the underlying ground set V and we have an oracle access to a function f that returns value $f(A)$ for every subset $A \subseteq V$. The goal is to maintain a good approximate set of at most k elements after every insertion and deletion using a fast query complexity.

Recently, Chen and Peng (Chen & Peng, 2022) show a lower bound for submodular maximization in the dynamic model: any randomized algorithm that achieves $(\frac{1}{2} + \epsilon)$-approximation ratio for dynamic submodular maximization under cardinality constraint k requires amortized query complexity $\frac{n^{\tilde{\Omega}(\epsilon)}}{k^3}$. Therefore, the important question that we seek to answer is the following:

Polylogarithmic question: Is there any dynamic algorithm for submodular maximization that maintains a $(\frac{1}{2} - \epsilon)$-approximate solution under cardinality constraint k with a query complexity $\text{poly}(\log(n), \log(k), 1/\epsilon)$ where n is the size of underlying ground set V?

Very recently, Lattanzi, Mitrovic, Norouzi-Fard, Tarnawski, and Zadimoghaddam (Lattanzi et al., 2020a) and Monemizadeh (Monemizadeh, 2020) tried to answer this question. The first work claimed to develop a dynamic algorithm that maintains an expected amortized $(\frac{1}{2} - \epsilon)$-approximate solution for this problem using an amortized expected query complexity of $\text{poly}(\log(n), \log(k), 1/\epsilon)$. The second paper proposed a randomized dynamic $(\frac{1}{2} - \epsilon)$-approximation algorithm with expected amortized query complexity of $O(k^2 \text{poly}(\log(n), 1/\epsilon))$, so it partially answers our question but not fully. Therefore, only the first paper (Lattanzi et al., 2020a) claims to answer our question. However, as we explain later in this paper, the proof of this dynamic algorithm has important gaps in the analysis.

In this paper, we answer the polylogarithmic question affirmatively; we develop a new simpler algorithm which maintains a $(\frac{1}{2} - \epsilon)$-approximate solution for submodular maximization problem under cardinality constraint k with a poly-logarithmic amortized query complexity.

1.1. Preliminaries

Submodular functions. Given a finite set of elements V, a function $f : 2^V \to \mathbb{R}$ is *submodular* if

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B).$$

for all $A, B \subseteq V$. In addition, function f is called *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B$, and is called *nonnegative* if $f(A) \geq 0$ for all $A \subseteq V$. In this paper, we consider non-negative monotone submodular functions. In addition, we assume that $f(\emptyset) = 0$.[1]

Oracle access. In this paper, we assume that our algorithm has access to an *oracle* that outputs value $f(A)$ when it is queried an arbitrary set A. We measure the *time complexity* of a dynamic algorithm in terms of its *query complexity*, which is the number of queries it makes to the oracle.

Dynamic model. We consider the classical dynamic model, where we are given a sequence of insertions and deletions of elements, and the goal is to maintain an α-approximate solution of size at most k at any time t. We assume without loss of generality that once an element is deleted, it is never re-inserted. Indeed, the reinsertion of an element e can always be simulated by inserting an identical copy of e.

We refer to the sequence of insertions and deletions as the *update stream* and assume that it is chosen by a non-adaptive adversary. The adversary is assumed to have knowledge of our algorithm and can control the sequence of insertions

[1]This assumption is without loss of generality since if $f(\emptyset) > 0$, then defining the function g as $g(V) := f(V) - f(\emptyset)$, g will be a monotone submodular function as well. In addition, optimizing f is equivalent to optimizing g, and any algorithm obtaining approximation ratio $\alpha < 1$ for g would obtain approximation ratio at least α for f as well.

and deletions but does not have access to the random bits our algorithm uses.

Notation. Given integers a, b, we use the notation $[a, b]$ to denote the set $\{a, \dots, b\}$ and $[a]$ to denote $[1, a]$. We use $\mathbb{1}[.]$ to denote the *indicator function*, i.e., $\mathbb{1}[A]$ equals one if A is true and equals zero otherwise.
We use $\mathbb{P}[.]$ and $\mathbb{E}[.]$ to denote the probability and expectation, respectively. For an event A satisfying $\mathbb{P}[A] > 0$, we use $\mathbb{P}[.|A]$ and $\mathbb{E}[.|A]$ to denote the *conditional probability* and *conditional expectation*. In this paper, we frequently condition on the value of a random variable. To avoid confusion, we will often use bold letters to denote random variables and non-bold letters to denote their values, e.g., $\mathbb{E}[\mathbf{X}|\mathbf{Y} = Y]$ denotes the expectation of the random variable $\mathbf{X}$, conditioned on the random variable $\mathbf{Y}$ attaining the value Y.
Given a submodular function f and sets A and B, we will use $f(A|B)$ to denote the value $f(A \cup B) - f(A)$.

2. Polylogarithmic Algorithm

2.1. Prior work

In this section, we highlight three main problems with the analysis of the algorithm in (Lattanzi et al., 2020a). [2]

On a high level, the algorithm obtains a solution Sol as union of sets $S_{i,j}$ such that each $S_{i,j}$ is claimed to satisfy $\mathbb{E}\left[f(S_{i,j}|S_{\text{pred}(i,j)})\right] \geq (1-\epsilon)\tau|S_{i,j}|$, where $\tau = \frac{\text{OPT}}{2k}$, the value OPT is the optimal value of any subset of ground set V of size at most k, and $S_{\text{pred}(i,j)}$ denotes the set of elements sampled before $S_{i,j}$. The algorithm stops when it has sampled k elements, or when it is no longer possible to sample any element with the desired property.

Next, we explain three issues with the algorithm's analysis.

First Issue: Incorrect conditioning in proof. The proof of Theorem 5.1 (that proves the $(1/2 - \epsilon)$-approximation guarantee) does not consider the effect of conditional probability when bounding the submodular value of the samples. Specifically, to prove that the approximation factor of the reported set is $\frac{1}{2} - \epsilon$, the proof considers two cases. The first case is when there are k elements sampled. In this case, it is claimed that the submodular value of output set Sol satisfies $f(\text{Sol}) \geq \frac{\text{OPT}}{2} - \epsilon$ since, in expectation, each $S_{i,j}$ contributed $(1-\epsilon)\tau|S_{i,j}|$. Given the assumption $|\text{Sol}| = k$ however, one needs to analyze the *conditional expectation* of $f(S_{i,j}|S_{\text{pred}(i,j)})$. In other words, just because $\mathbb{E}\left[f(S_{i,j}|S_{\text{pred}(i,j)})\right] \geq (1-\epsilon)\tau|S_{i,j}|$, does not mean that $\mathbb{E}\left[f(S_{i,j}|S_{\text{pred}(i,j)})\Big||\text{Sol}| = k\right] \geq (1-\epsilon)\tau|S_{i,j}|$.

Intuitively, one can expect that whenever samples $S_{i,j}$ have high quality, i.e., $f(S_{i,j}|S_{\text{pred}(i,j)})$ is high, the algorithm would be more likely to terminate with less than k samples since the obtained samples already have high quality and the remaining elements may contribute little to them. This means that the condition $|\text{Sol}| = k$ may introduce a negative bias on the value of $f(S_{i,j}|S_{\text{pred}(i,j)})$.

Second Issue: Lack of analysis for biased samples. Another issue is that it is not clear whether the identity

$$\mathbb{E}\left[f(S_{i,j}|S_{\text{pred}(i,j)})\right] \geq (1-\epsilon)\tau|S_{i,j}| \tag{1}$$

holds in the early step of the analysis, even without the extra condition that $|\text{Sol}| = k$. The analysis considers the value of $S_{i,j}$ and $S_{\text{pred}(i,j)}$ the last time LEVEL-CONSTRUCT(ℓ) was called. It is then claimed that because $S_{i,j}$ was obtained by invoking the peeling algorithm, (1) should hold. Importantly however, the analysis of the peeling algorithm assumes that the chosen elements we obtained by sampling *uniformly at random*. While the claim may hold if the values of $S_{\text{pred}(i,j)}$ and $A_{i,j}$ (the set that $S_{i,j}$ is sampled from) were fixed, here the values $S_{\text{pred}(i,j)}$ can only be calculated randomly, by looking back at the last time LEVEL-CONSTRUCT(ℓ) is called. Since this time itself depends on the value of $S_{i,j}$, this looking back process may introduce bias, and it is not clear how the bias should be handled.

To illustrate the issue, consider the following simple decremental process. Assume that we are given a set V, and we sample k elements from V to obtain a set S. An adversary then deletes an element e from V and, if an element in S is deleted, we take k fresh samples from V. Let (V_1, S_1) and (V_2, S_2) denote the value of (V, S) before and after the deletion respectively and let (V', S') denote the value of (V, S) the last time we sampled k elements. At first glance, it may seem that the elements of S' are k random elements from V', this is not the case. Indeed, if $V' = V_1$, then we know that k fresh samples were *not taken*, which implies $e \notin S'$. Since k samples taken uniformly at random would contain e with positive probability, this shows that the samples are not uniformly at random. [3]

[2]An earlier version of this paper was submitted to the SODA'23 conference in July 2022. In February 2023, the authors of (Lattanzi et al., 2020a) contacted us to mention that one of the authors was a referee for our SODA'23 submission, they agree with a bug and they will have a fix for it in their revised arxiv paper (Lattanzi et al., 2020b).

[3]The bias here is reminiscent of the so-called random incidence paradox, also known as the waiting time paradox and the related inspection paradox (Ross, 2003). Consider a Poisson process that has been running forever and assume that the arrival rate is $\lambda = 4$ per hour. This means that the average length between two successive arrivals is $\frac{1}{\lambda}$. One possible way to estimate this arrival rate is to pick an arbitrary point x, e.g., $x = 0$, and measure the length of time between the first arrival after x and the last arrival before x. However, it can be shown that this results in an estimator with expectation $\frac{2}{\lambda}$ and is therefore biased. To understand why, consider the segmentation of the real line caused by arrivals, i.e., each

The algorithm of (Lattanzi et al., 2020a) has a "resampling" condition similar to what is described above. In each level ℓ, they maintain a series of buckets, where the samples $S_{i,\ell}$ on each bucket are obtained by repeatedly invoking the peeling algorithm and depend on the samples of the previous samples $S_{j,\ell}$ for $j < i$. Once an ϵ-fraction of $S_{i,\ell}$ is deleted, the entire level ℓ is reconstructed.

Third Issue: Expectation bound. [4] Lemma $C.4$ bounds the expectation of the ratio of two expressions, but the applications of this lemma assume that the bound applies to the ratios of the expectations, which is not the same thing.

Given the above issues, a natural question is whether it is even possible to obtain $\frac{1}{2} - \epsilon$ approximation factor with polylogarithmic query complexity. Indeed, as we explained in above, Chen and Peng (Chen & Peng, 2022) show that obtaining $\frac{1}{2} + \epsilon$ approximation requires an query complexity that is polynomial in n (in fact, requires amortized query complexity $\frac{n^{\tilde{\Omega}(\epsilon)}}{k^3}$), suggesting that $\frac{1}{2} - \epsilon$ may be impossible as well. In this paper, we show that this is not the case. We develop a simpler algorithm than that proposed in (Lattanzi et al., 2020a) which maintains an $(\frac{1}{2} - \epsilon)$-approximate solution for submodular maximization under cardinality constraint k with a poly-logarithmic query complexity. We emphasize that while our algorithm is simpler, it requires careful analysis to avoid the aforementioned issues.

2.2. Overview of our algorithm

Offline algorithm. In this section, we first present the offline version of our algorithm. Later, we show how to support insertions and deletions. We first remove all elements that have submodular value less than $\tau = OPT/2k$ and let R_1 be the remaining elements. Let $G_0 = \emptyset$ and $i = 1$. In each iteration, we bucket the elements of R_i in based on their relative marginal gain to G_{i-1}, i.e., $f(e|G_{i-1})$ for $e \in R_i$, such that all the elements in the same bucket have the same value of $f(e|G_{i-1})$ up to a factor of $1 + \epsilon_{\text{buck}}$. We use $R_i^{(b(i))}$ to denote the largest (maximum size) bucket.

Next, for a suitable number m_i, we take a uniformly random subset of size m_i from the largest bucket and we denote it by S_i. We then add S_i to G_{i-1} to form G_i. Next, we remove all elements $f(e|G_i) < \tau$ from R_i to form R_{i+1} using the function:

$$R_{i+1} = \textsc{Filter}(R_i, G_i, \tau) = \begin{cases} \{e \in R_i : f(e|G_i) \geq \tau\}, & \text{if } |G_i| < k. \\ \emptyset, & \text{otherwise.} \end{cases}$$

Later, we repeat the above process by setting $i = i + 1$. The process is continued until there is no element with $f(e|G_i) \geq \tau$ left, or we have chosen k elements. The final value of G_i, denoted by G_T, is reported as the output of the algorithm. After our offline algorithm has executed, sets $R_0, \ldots, R_{T+1}, S_1, \ldots, S_T, G_0, \ldots G_T$ will satisfy the following properties

Properties:
(1) $R_i = \textsc{Filter}(R_{i-1}, G_{i-1}, \tau)$ for $i \in [1, T+1]$.
(2) Each S_i is a uniformly random subset of size m_i from the largest bucket of R_i, denoted by $R_i^{(b(i))}$ and $G_i = \bigcup_{j \leq i} S_j$.

Dynamic algorithm. To handle insertions and deletions, we maintain modifications of the above properties after the updates in the stream. These properties will ensure that our algorithm has an approximation factor of $\frac{1}{2} - \epsilon$ using expected amortized query complexity that is polylogarithmic in n, k.

To handle insertions, for each level i, we maintain a set $\overline{R}_i \supseteq R_i$ that will temporarily hold the elements in a level before they are processed. More formally, each time an element v is inserted, we add v to all $\overline{R}_i$ such that $f(v|G_{i-1}) \geq \tau$. Once the size of $\overline{R}_i$ is at least $\frac{3}{2}|R_i|$, we reconstruct the level i. A formal pseudocode is shown in Algorithm 2.
To handle deletions, we keep track of the deleted elements by adding them to a set D. For each level i, we keep track of the number of elements deleted in the bucket $R_i^{(b(i))}$ that we had chosen to sample S_i from. Once an ϵ-fraction of these elements is deleted, we restart the offline algorithm from level i by invoking RECONSTRUCT(i). A formal pseudocode is provided in Algorithm 3.

We note that our construction is different from that of (Lattanzi et al., 2020a) in a number of ways. Firstly, their algorithm maintains multiple buckets in each level, and each bucket contains its own set of samples $S_{i,\ell}$, where the set $S_{i,\ell}$ is sampled after $S_{i-1,\ell}$. Once an ϵ-fraction of $S_{j,\ell}$ is deleted for some j, the entire bucket ℓ is reconstructed. Therefore, the resampling condition for $S_{i,\ell}$ depends on all $S_{j,\ell}$, including $S_{i,\ell}$ itself and $S_{j,\ell}$ for $j > \ell$ which in turn depend on $S_{i,\ell}$. In contrast, our reconstruction condition for S_i depends only on $S_1, \ldots, S_{i-1}$ and does not depend on S_i or any S_j for $j > i$. As we show in Section A.1 (Lemma 12), this choice allows us to prove that the samples S_i are *always* a uniformly random subset of $R_i^{(b(i))}$.

segment represents the interval between two successive arrivals. Intuitively, since we are choosing x as a random point in the real line, x is more likely to fall in the longer intervals than the short ones. This results in a bias in favor of the longer intervals in the estimator, leading to a larger estimate of $\frac{2}{\lambda}$.

[4] We thank an anonymous reviewer in SODA'23 for pointing out this issue.

Now, we explain how we choose sample size m_i in each level. Our choice here is based on the peeling/threshold-sampling algorithm (Fahrbach et al., 2019), though we require new analysis in the proofs. One possibility is to set $m_i = 1$. This ensures that $f(S_i|G_{i-1}) \geq |S_i| \cdot \tau$ since all the elements in R_i are guaranteed to satisfy $f(e|G_{i-1}) \geq \tau$. The problem with this approach is that the number of levels T can be polynomial in k as we may need to sample every element of a solution set which is of size k in a separate level. This would in turn lead to a polynomial query complexity.

Another extreme is to choose a very large integer, e.g., set $m_i = k$. The problem with this approach however is that the sample S_i may have low relative marginal gain to G_{i-1}, i.e., $f(S_i|G_{i-1})$ may be low. Indeed, once we sample one element s_i from $R_i^{(b(i))}$, we expect the marginal gain of the remaining element to drop since f is submodular.

A good value of m_i needs to balance the above trade-off. Let $\tau^{(i)}$ denote the minimum threshold corresponding to $R_i^{(b(i))}$, i.e., $e \in [\tau^{(i)}, (1+\epsilon)\tau^{(i)})$ for $e \in R_i^{(b(i))}$. Intuitively, we want to choose an m_i such that

1. In expectation, each element in S_i adds at least $(1-\epsilon) \cdot \tau^{(i)}$ to the value of the output, i.e., $f(S_i|G_{i-1}) \geq (1-\epsilon) \cdot |S_i| \cdot \tau^{(i)}$.
2. The value m_i is large enough to ensure that $|R_{i+1}|$ is considerably smaller than $|R_i|$, ensuring that the number of levels is not too large.

The first property is important for ensuring the approximation guarantee of our algorithm, while the second property controls its query complexity.

The main idea is to choose the largest m_i that satisfies a modification of the first property. Given a parameter ϵ, we search for the largest integer m_i such that when sampling m_i elements at random, the *last element* has marginal gain at least $\tau^{(i)}$ with probability at least $1-\epsilon$. Since the last element is likely to be the worst element because of submodularity, this ensures the first property. In addition, since we have chosen the largest integer m_i, a random sample of the remaining elements should contribute $\tau^{(i)}$ with probability at most $1-\epsilon$ since it is effectively the last elements of a sample of size $m_i + 1$. This ensures that an ϵ fraction of the elements in $R_i^{(b(i))}$ will be removed from the $b(i)$-th bucket in the next level, which in turn allows us to bound the number of levels.

To find the largest such m_i, we binary search over the set of all possible values m' for m_i. For each m', we test whether it satisfies the first property above by sampling S' for $O(\frac{1}{\epsilon^2} \cdot \log(k/\epsilon))$ trials and testing whether its last element has marginal gain at least τ in more than $(1-\epsilon)$ fraction of the trials. A standard Chernoff bound then shows that m_i satisfies the mentioned properties with high probability.

Finally, we relax the assumption of known OPT by maintaining multiple parallel runs, indexed by $p \in \mathbb{Z}$. For each run p, we will use the value $\mathrm{OPT}_p = (1+\epsilon_{\mathrm{opt}})^p$ for OPT in the algorithm and only insert elements with $f(e) \in [\epsilon \cdot \frac{\mathrm{OPT}_p}{2k}, \mathrm{OPT}_p]$. We always output the set with maximum value of f across all the runs. This increases the query complexity of our algorithm by a factor of at most $O(\log_{1+\epsilon_{\mathrm{opt}}}(k/\epsilon))$, since each element needs to be inserted into $\lceil \log_{1+\epsilon_{\mathrm{opt}}}(2k/\epsilon) \rceil$ runs, and reduces the approximation guarantee by at most ϵ, since the discarded elements affect the solution by at most $\epsilon \cdot \mathrm{OPT}$.

2.3. Overview of techniques

Approximation factor We start by giving an overview of our proof for the approximation factor of the algorithm. As we show in Section A.1 (Lemmas 5 and 12), the output of our algorithm satisfies the following important properties.

1. For all $i \in [0, T]$, defining $\widehat{R}_i := \overline{R}_i \backslash D$, $\widehat{R}_{i+1} = \textsc{Filter}(\widehat{R}_i, G_i)$. In addition, $\widehat{R}_{T+1} = \emptyset$.
2. Conditioned on the values $S_1, \ldots S_{i-1}$ and m_i, the set S_i is a uniform subset of size m_i from $R_i^{(b(i))}$. In other words,
$$\mathbb{P}[\mathbf{S}_i = S | T \geq i, S_1, \ldots, S_{i-1}, m_i] = \frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right] \;,$$

As mentioned earlier, the integers m_i are chosen such that in expectation, each element of S_i contributes at least $(1-\epsilon)\tau$, i.e., $\mathbb{E}[f(S_i|G_{i-1})] \geq (1-\epsilon) \cdot |S_i| \cdot \tau$. If the sample quality was deterministic, i.e., $f(S_i|G_{i-1})$ was guaranteed to always be at least $(1-\epsilon)\tau$, then we could have used a well-known argument that considers two separate cases based on whether or not G_T has k elements. If it has k elements, then since each sample added, on average, $(1-\epsilon)\tau$ to final f, then the claim would hold. If it does not have k elements, then since $\widehat{R}_{i+1} = \textsc{Filter}(\widehat{R}_i, G_i)$ and $\widehat{R}_{T+1} = \emptyset$, each element e in the optimal set must satisfy $f(e|G_T) < \tau$, which can then be used to show that $f(G_T) \geq f(G_{\mathrm{opt}} \cup G) - k\tau \geq \mathrm{OPT} - k\tau = \frac{\mathrm{OPT}}{2}$.

However, the issue with this type of analysis is since conditioning on the event $|G_T| = k$, would mean that we can no longer guarantee $\mathbb{E}[f(S_i|G_{i-1})] \geq (1-\epsilon)\tau|S_i|$. In other words, just because $\mathbb{E}[f(S_i|G_{i-1})|T \geq i] \geq (1-\epsilon)\tau|S_i|$ holds, it does not mean that $\mathbb{E}\left[f(S_i|G_{i-1}) \middle| T \geq i, |G_T| = k\right] \geq (1-\epsilon)\tau|S_i|$ would hold as well. Indeed, one can expect that if the sample set S_i has a high quality, i.e., $f(S_i|G_{i-1})$ is large;

then the algorithm would be more likely to reach $OPT/2$ with less than k elements. The condition $|G_T| \leq k$, therefore, induces a negative bias on the quality of the samples. Further complicating the analysis is the fact that some of the samples are deleted, and the output of our algorithm is $G_T \backslash D$, not G_T.

To address this issue, we introduce a novel relaxation function that may be of independent interest. By carefully unifying the cases of $|G_T| = k$ and $|G_T| < k$, we first show that our new relaxation function is a lower bound on $f(G_T \backslash D)$. We then bound the contribution of each level to the relaxation function to prove the desired $\frac{\text{OPT}}{2} - \epsilon$ bound. We refer the reader to Section A.3 for more details.

Query complexity. To bound the query complexity of our algorithm, we show that our choice of m_i means the number of levels is polylogarithmic in n, k. The choice ensures that at least $\epsilon \cdot |R_i^{(b(i))}|$ elements in $R_i^{(b(i))}$ will either be moved to a *lower* bucket in R_{i+1}, or they will not appear in R_{i+1} altogether. Since the number of buckets is bounded by $\log_{1+\epsilon_{\text{buck}}}(\frac{\text{OPT}}{\tau}) = \log_{1+\epsilon_{\text{buck}}}(2k)$, each element can only go down at most $\log_{1+\epsilon_{\text{buck}}}(2k)$ times. Our proof formalizes this intuition, though there are some subtleties given that the above guarantees hold only in expectation.

The polylogarithmic bound on the number of levels implies that each call to RECONSTRUCT(i) makes at most $O(|R_i| \cdot \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}))$ queries. Given the choice of reconstruction conditions in our algorithm, when an insertion or deletion triggers a call to RECONSTRUCT(i), we can charge it back to at least $\frac{|R_i|}{\text{poly}(log(k), \frac{1}{\epsilon})}$ elements that triggered it. This implies that each RECONSTRUCT(i) charges back a polylogarithmic number of queries to each element. In addition, each update can only be charged once by each level (the first time the level is reconstructed after this update). Since the number of levels is polylogarithmic, this implies a polylogarithmic bound on the query complexity. We refer the reader to Section A.4 for a more detailed analysis.

Comparison with prior work. We briefly highlight the main differences in the algorithm and analysis that allow us to sidestep the aforementioned issues of the analysis in (Lattanzi et al., 2020a).

The first main difference is the design of multi-level structure and reconstruction condition. Our structure only samples elements once per level, and our reconstruction condition considers the effect of deletions on the set of elements these samples were chosen from, not the actual elements themselves. This ensures that whether or not a level is reconstructed is independent of its samples. This effectively resolves the biasing issue discussed in Section 2.1. A consequence of these changes, however, is that the existing analysis needs to be altered significantly for both the approximation guarantee and query complexity.

The second main difference is the introduction of a relaxation function that always lower bounds the value of the output $f(G_T \backslash D)$. The relaxation function unifies the two cases of $|G_T| = k$ and $|G_T| < k$, allowing us to sidestep the conditioning issue discussed in Section 2.1.

3. Proposed Algorithm

In this section, we present our dynamic algorithm for submodular maximization under cardinality constraint k using polylogarithmic number of oracle queries. We start by presenting an offline algorithm for the problem in Section 3.1. We then show how to extend this to a dynamic setting by considering lazy updates in Sections 3.2, and 3.3. In these sections, we assume access to a parameter OPT, which estimates the value of the optimal solution up to a factor of $1 + \epsilon$. More formally, we assume that $f(G_{\text{opt}}) \leq \text{OPT} \leq (1 + \epsilon_{\text{opt}})f(G_{\text{opt}})$ where G_{opt} denotes the optimal solution. In section 3.5, we show how to remove this restriction by considering parallel runs.

3.1. Offline Algorithm

We now present an offline algorithm for the problem. In this section, we assume that the value of the optimal solution, which we denote by OPT, is known to the algorithm. As we will show in our analysis, OPT does not need to be an exact estimate, and our results still hold as long as it is approximately correct. Later in Section 3.5, we will remove this restriction altogether.

Setting the threshold τ to be $\frac{\text{OPT}}{2k}$ and defining the set G_0 as $\emptyset$, our algorithm starts by removing all elements with submodular value $f(e)$ less than τ and collecting the remaining elements in the set R_1. Starting with $i := 1$, in each iteration, we group the samples in buckets based on their relative marginal gain to G_{i-1}, i.e., $f(e|G_{i-1})$, such that all the elements in the same bucket have the same value of $f(e|G_{i-1})$ up to a factor of $1 + \epsilon_{\text{buck}}$. In other words, all the elements e in the same bucket satisfy $f(e|G_{i-1}) \in [\tau^{(i)}, (1+\epsilon_{\text{buck}}) \cdot \tau^{(i)})$ for some threshold $\tau^{(i)}$. This requires $\log_{1+\epsilon_{\text{buck}}}(2k)$ buckets as

1. We can assume that $f(e|G_{i-1}) \geq \tau$ for all $e \in R_i$. This was the case for $i = 1$, and we will also ensure it to be true for $i \geq 1$ by removing all elements with $f(e|G_{i-1}) < \tau$ when forming R_i.
2. All elements have $f(e|G_{i-1}) \leq f(e) \leq \text{OPT} = 2k \cdot \tau$.

Next, we take a uniformly random subset of size m_i from the largest bucket for a suitable number m_i, forming the samples S_i. We will explain how to choose m_i in Section

3.4. We then add S_i to G_{i-1} to form G_i and remove all elements $f(e|G_i) \leq \tau$ from R_i to form R_{i+1}. Then we repeat the above process with $i \leftarrow i+1$ until there are no elements with $f(e|G_i) \geq \tau$ left, or we have chosen k elements. The final value of G_i, denoted by G_T, is given as the algorithm's output. A formal pseudocode is provided in Algorithm 1. The values $\overline{R}$ and D in the algorithm will be necessary for the next section, and for now, we can think of them as $\overline{R}_i = R_i$ and $D = \emptyset$. As seen in Algorithm 1, the main part of our design, which is the function RECONSTRUCT, is more general than the description above and can, effectively, start midway from a set R_i for $i \geq 1$. This property will play a crucial role in the upcoming sections as we require reconstructing a portion of our data structure to handle insertions and deletions.

Algorithm 1 Offline algorithm

```
 1: procedure INIT($V$, OPT)
 2:     $R_0 \leftarrow V$,   $\tau \leftarrow \frac{\text{OPT}}{2k}$,   $G_0 \leftarrow \emptyset$
 3:     $D \leftarrow \emptyset$,   $\overline{R}_0 \leftarrow R_0$
 4:     $R_1 \leftarrow$ FILTER$(R_0, G_0, \tau)$,   $\overline{R}_1 \leftarrow R_1$
 5:     RECONSTRUCT(1)
 6: procedure RECONSTRUCT($i$)
 7:     $R_i \leftarrow \overline{R}_i \backslash D$,   $\overline{R}_i \leftarrow R_i$
 8:     while $R_i \neq \emptyset$ do
 9:         for $j \in [0, \lfloor \log_{1+\epsilon_{\text{buck}}}(2k) \rfloor]$ do
10:             $R_i^{(j)} \leftarrow \{e \in R_i : \frac{f(e|G_{i-1})}{\tau} \in [(1+\epsilon_{\text{buck}})^j, (1+\epsilon_{\text{buck}})^{j+1})\}$
11:         $b(i) \in \arg\max_j \left|R_i^{(j)}\right|$
12:         $\tau^{(i)} \leftarrow (1+\epsilon_{\text{buck}})^{b(i)} \cdot \tau$
13:         $m_i \leftarrow$ CALCSAMPLECOUNT$(R_i^{(b(i))}, G_{i-1}, \tau^{(i)})$
14:         $S_i \leftarrow$ Uniform subset of size $m_i$ from $R_i^{(b(i))}$
15:         $G_i \leftarrow G_{i-1} \cup S_i$
16:         $R_{i+1} \leftarrow$ FILTER$(R_i, G_i, \tau)$,   $\overline{R}_{i+1} \leftarrow R_{i+1}$
17:         $i \leftarrow i+1$
18:     $T \leftarrow i-1$
19: function FILTER($R'$, $G'$, $\tau'$)
20:     if $|G'| = k$ then
21:         return $\emptyset$
22:     else
23:         return $\{e \in R' : f(e|G') \geq \tau'\}$
```

Next, we show how to handle insertions and deletions in our algorithm.

3.2. Insertion

We take a lazy approach for dealing with insertions; for each level i, we maintain a set $\overline{R}_i \supseteq R_i$ that will temporarily hold the elements in a level, and process these elements once the number of elements is sufficiently large. More formally, each time an element v is inserted, we add v to all $\overline{R}_i$ such that $f(v|G_{i-1}) \geq \tau$. Once the size of $\overline{R}_i$ is at least $\frac{3}{2}$ the size of R_i, we reconstruct level i. A formal pseudocode is shown in Algorithm 2.

Algorithm 2 Insert

```
1: procedure INSERT($v$)
2:     $\overline{R}_0 \leftarrow \overline{R}_0 \cup \{v\}$
3:     for $i \leftarrow 1, \ldots, T+1$ do
4:         if $f(v|G_{i-1}) < \tau$ or $|G_{i-1}| = k$ then
5:             break
6:         $\overline{R}_i \leftarrow \overline{R}_i \cup \{v\}$
7:         if $i = T+1$ or $|\overline{R}_i| \geq \frac{3}{2} \cdot |R_i|$ then
8:             RECONSTRUCT(i)
9:             break
```

3.3. Deletion

To handle deletions, we keep track of the deleted elements by adding them to a set D. For each level i, we keep track of the number of elements deleted in the bucket $R_i^{(b(i))}$ that we had chosen to sample S_i from. Once an ϵ-fraction of these elements is deleted, we restart the offline algorithm from level i by invoking RECONSTRUCT(i). A formal pseudocode is provided in Algorithm 3. Note that level i is reconstructed with $R_i \leftarrow \overline{R}_i \backslash D$ to avoid using deleted values in the reconstruction.

Algorithm 3 Delete

```
1: procedure DELETE($v$)
2:     $D \leftarrow D \cup v$
3:     for $i \leftarrow 1, \ldots, T$ do
4:         if $|D \cap R_i^{(b(i))}| \geq \epsilon_{\text{del}} \cdot |R_i^{(b(i))}|$ then
5:             RECONSTRUCT(i)
6:             break
```

3.4. Choice of sample size

As explained in the intro, the main idea behind our algorithm is to find the largest integer m_i such that if we sample m_i elements uniformly at random, the last sampled element has marginal gain at least $\tau^{(i)}$ with probability at least $1-\epsilon$. To achieve this, we binary search over all possible values m'. For each m', we use repeated trials to estimate the probability that the last element of S_i has marginal gain at least $\tau^{(i)}$. As we show in Section A.2, a standard Chernoff bound implies that using a polylogarithmic number of samples, we can estimate this probability with error at most $\frac{\epsilon_{\text{sam}}}{10}$. A formal pseudocode of the above sketch is provided in Algorithm 4 in which we use notation u.a.r for a set that is sampled *uniformly at random*.

Algorithm 4 CalcSampleCount

```
1: function ReduceMean($R', G', m'$)
2:     for $t \leftarrow 1, \ldots, \left\lceil \frac{4}{{\epsilon_{\text{sam}}}^2} \log\left(\frac{200k^{11}}{\epsilon_{\text{sam}}}\right) \right\rceil$ do
3:         Sample $S'$ of size $m' - 1$ u.a.r. from $R'$
4:         Sample element $s$ from $R' \setminus S'$ u.a.r.
5:         $I_t \leftarrow \mathbb{1}\left[f(s|G' \cup S') \geq \tau'\right]$
6:     return mean of $I_t$
7: function CalcSampleCount($R', G', \tau'$)
8:     $m \leftarrow 1, M \leftarrow \min\{k - |G'|, |R'|\}$
9:     if ReduceMean($R', G', M$) $\geq 1 - \epsilon_{\text{sam}}$ then
10:        return $M$
11:    while $M - m > 1$ do
12:        $m' \leftarrow \lfloor \frac{m+M}{2} \rfloor$
13:        if ReduceMean($R', G', m'$) $\geq 1 - \epsilon_{\text{sam}}$ then
14:            $m \leftarrow m'$
15:        else
16:            $M \leftarrow m'$
17:    return $m$
```

3.5. Unknown OPT

In this section, we relax the assumption of known OPT by maintaining multiple parallel runs, indexed by $p \in \mathbb{Z}$. For each run p, we will use the value $\text{OPT}_p = (1 + \epsilon_{\text{opt}})^p$ for OPT in the algorithm and only insert elements with $f(e) \in [\epsilon \cdot \frac{\text{OPT}_p}{2k}, \text{OPT}_p]$. We always output the set with maximum value of f across all the runs.

3.6. Choice of parameters

For some $\epsilon < \frac{1}{10}$, we set $\epsilon_{\text{sam}} = \epsilon_{\text{buck}} = \epsilon_{\text{opt}} = \epsilon$ and $\epsilon_{\text{del}} = \frac{\epsilon}{20}$.

4. Theoretical Analysis

In this section, we state our main theoretical result.

Theorem 1. *There is an algorithm for dynamic submodular maximization that maintains a set with expected $\frac{1}{2} - \epsilon$ approximation factor that makes at most poly$(\log(n), \log(k), \frac{1}{\epsilon})$ amortized queries in expectation, where n denotes the largest number of elements at any point of the stream.*

To prove this result, we establish two theorems that consider the approximation factor and query complexity of our algorithm, respectively. First, in Section A.3, we prove the following result.

Theorem 2. *The dynamic Algorithm 1 (with Algorithms 2 and 3 for handling updates) maintains an output set $G_T \setminus D$ with expected approximation factor $\frac{1}{2} - O(\epsilon)$ as long as the optimal solution G_{opt} satisfies*

$$f(G_{\text{opt}}) \leq OPT \leq (1 + \epsilon_{\text{opt}}) f(G_{\text{opt}}).$$

Next, in Section A.4, we prove the following result.

Theorem 3. *The dynamic algorithm 1 (with Algorithms 2 and 3 for handling updates) has an expected amortized query complexity that is polynomial in $\log(n), \log(k)$, and $\frac{1}{\epsilon}$.*

We note that the above results imply Theorem 1 since, as mentioned in Section 3.5, maintaining multiple runs adds a $\log_{1+\epsilon_{\text{opt}}}(k/\epsilon)$ term in the query complexity while the approximation ratio reduces by an extra ϵ term.

5. Conclusion

In this paper, we presented the first provably correct polylogarithmic algorithm for the dynamic submodular maximization problem under a cardinality constraint. For future work, it would be interesting to improve the query complexity of this problem or the more general problem of maximizing a submodular function under a matroid constraint for which the current best result is a $(\frac{1}{4} - \epsilon)$ approximation algorithm with $O(k \log(k) \log^3(\frac{k}{\epsilon}))$ query complexity obtained by (Banihashem et al., 2023).

6. Acknowledgements

The work is partially support by DARPA QuICC, NSF AF:Small #2218678, and NSF AF:Small #2114269

References

Ashkan, A., Kveton, B., Berkovsky, S., and Wen, Z. Optimal greedy diversity for recommendation. In Yang, Q. and Wooldridge, M. J. (eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 1742–1748. AAAI Press, 2015. URL `http://ijcai.org/Abstract/15/248`.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming submodular maximization: massive data summarization on the fly. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R. (eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 671–680. ACM, 2014. doi: 10.1145/2623330.2623637. URL `https://doi.org/10.1145/2623330.2623637`.

Banihashem, K., Biabani, L., Goudarzi, S., Hajiaghayi, M., Jabbarzade, P., and Monemizadeh, M. Dynamic algorithms for matroid submodular maximization. *arXiv preprint arXiv:2306.00959*, 2023.

Bar-Ilan, J., Kortsarz, G., and Peleg, D. Generalized submodular cover problems and applications. *Theor.*

Comput. Sci., 250(1-2):179–200, 2001. doi: 10.1016/S0304-3975(99)00130-9. URL `https://doi.org/10.1016/S0304-3975(99)00130-9`.

Bateni, M., Chen, L., Esfandiari, H., Fu, T., Mirrokni, V. S., and Rostamizadeh, A. Categorical feature compression via submodular optimization. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 515–523. PMLR, 2019. URL `http://proceedings.mlr.press/v97/bateni19a.html`.

Benouaret, I., Amer-Yahia, S., and Roy, S. B. An efficient greedy algorithm for sequence recommendation. In Hartmann, S., Küng, J., Chakravarthy, S., Anderst-Kotsis, G., Tjoa, A. M., and Khalil, I. (eds.), *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part I*, volume 11706 of *Lecture Notes in Computer Science*, pp. 314–326. Springer, 2019. doi: 10.1007/978-3-030-27615-7_24. URL `https://doi.org/10.1007/978-3-030-27615-7_24`.

Bernstein, A., Forster, S., and Henzinger, M. A deamortization approach for dynamic spanner and dynamic maximal matching. *ACM Trans. Algorithms*, 17(4):29:1–29:51, 2021. doi: 10.1145/3469833. URL `https://doi.org/10.1145/3469833`.

Bhattacharya, S., Chakrabarty, D., and Henzinger, M. Deterministic dynamic matching in O(1) update time. *Algorithmica*, 82(4):1057–1080, 2020. doi: 10.1007/s00453-019-00630-4. URL `https://doi.org/10.1007/s00453-019-00630-4`.

Bhattacharya, S., Henzinger, M., Nanongkai, D., and Wu, X. Dynamic set cover: Improved amortized and worst-case update time. In Marx, D. (ed.), *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pp. 2537–2549. SIAM, 2021. doi: 10.1137/1.9781611976465.150. URL `https://doi.org/10.1137/1.9781611976465.150`.

Chen, L., Krause, A., and Karbasi, A. Interactive submodular bandit. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 141–152, 2017.

Chen, W., Wang, Y., and Yang, S. Efficient influence maximization in social networks. In IV, J. F. E., Fogelman-Soulié, F., Flach, P. A., and Zaki, M. J. (eds.), *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pp. 199–208. ACM, 2009. doi: 10.1145/1557019.1557047. URL `https://doi.org/10.1145/1557019.1557047`.

Chen, X. and Peng, B. On the complexity of dynamic submodular maximization. In Leonardi, S. and Gupta, A. (eds.), *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pp. 1685–1698. ACM, 2022. doi: 10.1145/3519935.3519951. URL `https://doi.org/10.1145/3519935.3519951`.

da Ponte Barbosa, R., Ene, A., Nguyen, H. L., and Ward, J. A new framework for distributed submodular maximization. In Dinur, I. (ed.), *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 645–654. IEEE Computer Society, 2016. doi: 10.1109/FOCS.2016.74. URL `https://doi.org/10.1109/FOCS.2016.74`.

Das, A. and Kempe, D. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *J. Mach. Learn. Res.*, 19: 3:1–3:34, 2018. URL `http://jmlr.org/papers/v19/16-534.html`.

Elenberg, E. R., Dimakis, A. G., Feldman, M., and Karbasi, A. Streaming weak submodularity: Interpreting neural networks on the fly. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4044–4054, 2017.

Elhamifar, E. Sequential facility location: Approximate submodularity and greedy algorithm. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1784–1793. PMLR, 2019. URL `http://proceedings.mlr.press/v97/elhamifar19a.html`.

Elhamifar, E. and Kaluza, M. C. D. P. Subset selection and summarization in sequential data. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1035–1045, 2017.

Fahrbach, M., Mirrokni, V., and Zadimoghaddam, M. Submodular maximization with nearly optimal approximation, adaptivity and query complexity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 255–273. SIAM, 2019.

Feldman, M., Karbasi, A., and Kazemi, E. Do less, get more: Streaming submodular maximization with subsampling. 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. 730–740, 2018.

Hanauer, K., Henzinger, M., and Schulz, C. Recent advances in fully dynamic graph algorithms (invited talk). In Aspnes, J. and Michail, O. (eds.), *1st Symposium on Algorithmic Foundations of Dynamic Networks, SAND 2022, March 28-30, 2022, Virtual Conference*, volume 221 of *LIPIcs*, pp. 1:1–1:47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.SAND.2022.1. URL `https://doi.org/10.4230/LIPIcs.SAND.2022.1`.

Kazemi, E., Zadimoghaddam, M., and Karbasi, A. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2549–2558. PMLR, 2018. URL `http://proceedings.mlr.press/v80/kazemi18a.html`.

Kazemi, E., Mitrovic, M., Zadimoghaddam, M., Lattanzi, S., and Karbasi, A. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3311–3320. PMLR, 2019. URL `http://proceedings.mlr.press/v97/kazemi19a.html`.

Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. Fast greedy algorithms in mapreduce and streaming. *ACM Trans. Parallel Comput.*, 2(3):14:1–14:22, 2015. doi: 10.1145/2809814. URL `https://doi.org/10.1145/2809814`.

Lattanzi, S., Mitrovic, S., Norouzi-Fard, A., Tarnawski, J., and Zadimoghaddam, M. Fully dynamic algorithm for constrained submodular optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a.

Lattanzi, S., Mitrović, S., Norouzi-Fard, A., Tarnawski, J., and Zadimoghaddam, M. Fully dynamic algorithm for constrained submodular optimization. *arXiv preprint arXiv:2006.04704*, 2020b.

Liu, P. and Vondrák, J. Submodular optimization in the mapreduce model. In Fineman, J. T. and Mitzenmacher, M. (eds.), *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, volume 69 of *OASICS*, pp. 18:1–18:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi: 10.4230/OASIcs.SOSA.2019.18. URL `https://doi.org/10.4230/OASIcs.SOSA.2019.18`.

McGregor, A. and Vu, H. T. Better streaming algorithms for the maximum coverage problem. *Theory Comput. Syst.*, 63(7):1595–1619, 2019. doi: 10.1007/s00224-018-9878-x. URL `https://doi.org/10.1007/s00224-018-9878-x`.

Mirzasoleiman, B., Karbasi, A., and Krause, A. Deletion-robust submodular maximization: Data summarization with ”the right to be forgotten”. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2449–2458. PMLR, 2017. URL `http://proceedings.mlr.press/v70/mirzasoleiman17a.html`.

Mitrovic, M., Kazemi, E., Feldman, M., Krause, A., and Karbasi, A. Adaptive sequence submodularity. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5353–5364, 2019.

Monemizadeh, M. Dynamic submodular maximization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978. doi: 10.1007/BF01588971. URL `https://doi.org/10.1007/BF01588971`.

Ohsaka, N. and Matsuoka, T. Reconfiguration problems on submodular functions. In Candan, K. S., Liu, H., Akoglu, L., Dong, X. L., and Tang, J. (eds.), *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, pp. 764–774. ACM, 2022. doi: 10.1145/3488560.3498382. URL `https://doi.org/10.1145/3488560.3498382`.

Parambath, S. A. P. Re-ranking based diversification: A unifying view. *CoRR*, abs/1906.11285, 2019. URL `http://arxiv.org/abs/1906.11285`.

Parambath, S. A. P., Vijayakumar, N., and Chawla, S. SAGA: A submodular greedy algorithm for group recommendation. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3900–3908. AAAI Press, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16405`.

Ross, S. M. The inspection paradox. *Probability in the Engineering and Informational Sciences*, 17(1):47–51, 2003.

Sagnol, G. Approximation of a maximum-submodular-coverage problem involving spectral functions, with application to experimental designs. *Discret. Appl. Math.*, 161(1-2):258–276, 2013. doi: 10.1016/j.dam.2012.07.016. URL `https://doi.org/10.1016/j.dam.2012.07.016`.

Salehi, M., Karbasi, A., Shen, X., Scheinost, D., and Constable, R. T. An exemplar-based approach to individualized parcellation reveals the need for sex specific functional networks. *NeuroImage*, 170:54–67, 2018. doi: 10.1016/j.neuroimage.2017.08.068. URL `https://doi.org/10.1016/j.neuroimage.2017.08.068`.

Tohidi, E., Amiri, R., Coutino, M., Gesbert, D., Leus, G., and Karbasi, A. Submodularity in action: From machine learning to signal processing applications. *IEEE Signal Process. Mag.*, 37(5):120–133, 2020. doi: 10.1109/MSP.2020.3003836. URL `https://doi.org/10.1109/MSP.2020.3003836`.

Tong, G., Wu, W., Tang, S., and Du, D. Adaptive influence maximization in dynamic social networks. *IEEE/ACM Trans. Netw.*, 25(1):112–125, 2017. doi: 10.1109/TNET.2016.2563397. URL `https://doi.org/10.1109/TNET.2016.2563397`.

Tsai, Y. and Tseng, K. Deep compressed sensing for learning submodular functions. *Sensors*, 20(9):2591, 2020. doi: 10.3390/s20092591. URL `https://doi.org/10.3390/s20092591`.

Tseng, K. and Mettler, B. Near-optimal probabilistic search via submodularity and sparse regression. *Auton. Robots*, 41(1):205–229, 2017. doi: 10.1007/s10514-015-9521-5. URL `https://doi.org/10.1007/s10514-015-9521-5`.

Wu, J. and Tseng, K. Adaptive submodular inverse reinforcement learning for spatial search and map exploration. *Auton. Robots*, 46(2):321–347, 2022. doi: 10.1007/s10514-021-10025-6. URL `https://doi.org/10.1007/s10514-021-10025-6`.

Zhang, Z., Shi, Y., Willson, J., Du, D., and Tong, G. Viral marketing with positive influence. In *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, pp. 1–8. IEEE, 2017. doi: 10.1109/INFOCOM.2017.8057070. URL `https://doi.org/10.1109/INFOCOM.2017.8057070`.

A. Proofs

We establish some properties of our algorithm in Section A.1 and A.2. Throughout the section, we use $\widehat{R}_i$ to denote $\overline{R}_i \backslash D$.

A.1. Invariants

A.1.1. Deterministic Invariants

In this section, we prove some invariants that will be useful in our proofs. We start with the following fact.

Fact 4. *For all $1 \leq i < j$, a call to* Reconstruct(j) *would not affect any of the values $R_i, \overline{R}_i, G_i$. Furthermore, the value of $\widehat{R}_i := \overline{R}_i \backslash D$ is unaffected by a call to* Reconstruct(i).

The fact follows from the pseudocodes given for Reconstruct.

Lemma 5. *For all $i \in [0, T]$, defining $\widehat{R}_i := \overline{R}_i \backslash D$,*

$$\widehat{R}_{i+1} = \text{Filter}(\widehat{R}_i, G_i). \tag{2}$$

Proof. Fix i and consider the last time Reconstruct(j) was called for some $j \leq i$. Our proof consists of 2 parts. We first show that (2) holds right after this call was made (Part 1). We then show that the condition holds after any subsequent insertion and deletion operations (Part 2).

Part 1. Consider the last execution of Reconstruct(j) for some $j \leq i$. The following properties hold after this execution.

1. $R_{i+1} = \text{Filter}(R_i, G_i)$ because of Line 16.
2. $\overline{R}_i = R_i$ and $\overline{R}_{i+1} = R_{i+1}$ because of Line 16 (or line 7 for i, if $j = i$).
3. $R_i \cap D = R_{i+1} \cap D = \emptyset$ because D is removed from R_j in Line 7, and all $R_{j'}$ for $j' > j$ are subsets of R_j because of Line 16.

Therefore,

$$\widehat{R}_{i+1} = R_{i+1} = \text{Filter}(R_i, G_i) = \text{Filter}(\widehat{R}_i, G_i)$$

Part 2 (Update). We assume that the update does not trigger a call to Reconstruct(j) for $j \leq i$ as otherwise, the claim holds by Part 1.
We first consider Insert. Each time a new value v is inserted to $\overline{R}_i$, then it is inserted into $\overline{R}_{i+1}$ if and only if $f(v|G_i)$ is at least τ and $|G_{i-1}| < k$. Therefore, $\widehat{R}_{i+1} = \text{Filter}(\widehat{R}_i, G_i)$ holds after the execution of Line 6 for $i + 1$. By Fact 4, even if the insertion triggers Reconstruct$(i + 1)$, the values of $\widehat{R}_i$ and $\widehat{R}_{i+1}$ would not be affected. Therefore, Insert preserves (2).
As for Delete, when an element is added to D, it is removed from both $\widehat{R}_i$ and $\widehat{R}_{i+1}$, and therefore (2) is preserved. The possible calls to Reconstruct(j) for $j \geq i + 1$ also preserve the property by Fact 4, which finishes the proof. □

Lemma 6. *$R_{T+1} = \emptyset$ and $R_i \neq \emptyset$ for $i \in [1, T]$.*

Proof. The claim holds at the beginning of the stream as $\text{Filter}(\emptyset, \cdot) = \emptyset$.
For each update in the stream, if the update triggers a call to Reconstruct(i) for some $i \leq T + 1$, the property would be preserved since Reconstruct stops building levels when $R_{T+1} = \emptyset$. Otherwise, we observe that since the value of T and R_i can only change during insertion and deletion through invoking Reconstruct, the invariant will be preserved. □

Lemma 7. *For all $i \in [T]$*

$$|R_i^{(b(i))} \cap D| \leq \epsilon_{\text{del}} |R_i^{(b(i))}|.$$

and

$$|\overline{R}_i| \leq \frac{3}{2} |R_i|.$$

Proof. The claim holds initially because $T = 0$, and it holds after each update because of the reconstruction condition of the levels. $\square$

Lemma 8. *For $i \in [T]$, $\overline{R}_i \neq \emptyset$ and $\widehat{R}_i \neq \emptyset$ and $\overline{R}_{T+1} = \widehat{R}_{T+1} = \emptyset$.*

Proof. We use Lemma 6 in both cases. We note that for $T+1$, we have $|\overline{R}_{T+1}| \leq \frac{3}{2}|R_{T+1}| = 0$ given Lemma 7. Therefore, $\overline{R}_{T+1} = \widehat{R}_{T+1} = \emptyset$.

For $i \in [T]$, we observe that if $\widehat{R}_i = \emptyset$, then it follows that $R_i^{(b(i))} \subseteq D$, which implies $|R_i^{(b(i))} \cap D| = |R_i^{(b(i))}|$. According to Lemma 7, this is only possible when $|R_i^{(b(i))}| = 0$. Because $b(i)$ was the largest bucket, this implies that $|R_i| = 0$ which contradicts Lemma 6. $\square$

Lemma 9. *For all $i \in [0, T]$, $\overline{R}_{i+1} \subseteq \overline{R}_i$.*

Proof. The claim holds initially since $T = 0$. It suffices to show that INSERT and DELETE preserve it.

We first observe that each time RECONSTRUCT(j) is called for some $j \leq i$, the property will hold since

$$\overline{R}_{i+1} = R_{i+1} = \text{FILTER}(R_i, G_i) \subseteq R_i = \overline{R}_i$$

It is also clear that calling RECONSTRUCT(j) for $j \geq i+1$ will preserve the property since it does not alter $\overline{R}_i$, while it may (if $j = i+1$) decrease $\overline{R}_{i+1}$. Other than RECONSTRUCT, the only way either $\overline{R}_i$ or $\overline{R}_{i+1}$ are altered is when an element is inserted into the sets during processing INSERT, but this also preserves the property since an element that was not inserted into $\overline{R}_i$, will not be inserted into $\overline{R}_{i+1}$. $\square$

A.1.2. RANDOM INVARIANTS

Next, we will state and prove the uniformity invariant. Before we do this, though, we will need a few definitions.

Definition 10 (History). *We define the* History *of Level i as*

$$H_i := (\overline{R}_0, \overline{R}_1, \ldots, \overline{R}_i, R_0, R_1, \ldots, R_i, S_1, \ldots, S_{i-1}, m_i). \tag{3}$$

Note that S_i is not included in the definition. Analogously, we define the random variable $\mathbf{H}_i$ as

$$\mathbf{H}_i := (\overline{\mathbf{R}}_0, \overline{\mathbf{R}}_1, \ldots, \overline{\mathbf{R}}_i, \mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_i, \mathbf{S}_1, \ldots, \mathbf{S}_{i-1}, \mathbf{m}_i).$$

Remark 11. *In the above definition, the random variable $\mathbf{H}_i$ is defined if and only if $\mathbf{T} \geq i$. Note that $\mathbf{H}_i$ is not defined for $i = T+1$ because there is no value of m_i. This will not be an issue in our analysis since we will work with $\mathbf{H}_i$ only when $\mathbf{T} \geq i$. Most notably, when conditioning on the value of $\mathbf{H}_i$, we will condition on the event $\mathbf{T} \geq i$ as well. Throughout the proofs, we will frequently write σ_i instead of $\mathbf{T} \geq i, \mathbf{H}_i = H_i$ for convenience.*

Our next lemma shows that conditioned on the history of level i, its sample set S_i is a random subset of size m_i from $R_i^{(b(i))}$.

Lemma 12. *For any $i \geq 1$, and any H_i such that $\mathbb{P}[\mathbf{T} \geq i, \mathbf{H}_i = H_i] > 0$,*

$$\mathbb{P}[\mathbf{S}_i = S | \mathbf{T} \geq i, \mathbf{H}_i = H_i] = \frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right]. \tag{4}$$

Before stating the proof, we observe that the property holds after a call to RECONSTRUCT, as we formally show in the following Lemma.

Lemma 13. *Assume we call* RECONSTRUCT(j) *for some $j \leq i$ in a data structure with values $T^-, R_0^-, \ldots,$ satisfying $T^- \geq i$, obtaining the new (random) values $\mathbf{T}, \mathbf{R}_0, \ldots$. Then for all sets S,*

$$\mathbb{P}[\mathbf{S}_i = S | \mathbf{T} \geq i, \mathbf{H}_i = H_i] = \frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right].$$

Proof. We observe that before S_i is sampled in Line 14, all of the values compromising H_i are already determined and will not change after S_i is sampled. Therefore, since the claim holds when S_i is sampled (because of Line 14), it holds afterwards as well. $\square$

Next, we prove Lemma 12 by showing that (4) is preserved after insertions and deletions.

Proof of Lemma 12. We will prove the claim by induction on the time step. At the beginning of the stream, the claim holds trivially since the event $\mathbf{T} \geq i$ is impossible because of $T = 0 < 1 \leq i$.

Assuming that the claim holds for some time step, we will show that it holds in the next time step as well. More formally, for an insertion or deletion of an element v in the stream, let $\mathbf{T}^-, \mathbf{R}_0^-, \ldots$ denote the values of the data structure before the operation and $\mathbf{T}, \mathbf{R}_0, \ldots,$ denote the values afterwards. For a fixed i, we need to show that (4) holds. The main idea behind the proof is to consider two cases based on whether or not CONSTRUCTLEVEL(i) was triggered by the insertion or deletion operation. In the first case, we will show that Lemma 13 implies the claim. In the second case, we will use the induction hypothesis, i.e, the fact that (4) held for the values $\mathbf{T}^-, \mathbf{R}_0^-, \ldots,$ to prove the claim. As we will observe, a crucial aspect of our proof in the second case is that the decision to invoke RECONSTRUCT(i) in INSERT and DELETE procedures is solely determined by the History of level i and *not* influenced by the actual samples S_i.

We now proceed with a formal proof. Let $\mathbf{L}_i$ be a random variable that takes the value 1 if CONSTRUCTLEVEL(i) was called because of the update, and takes the value 0 otherwise. Let the shorthand σ_i denote the event $\mathbf{T} \geq i \wedge \mathbf{H}_i = H_i$. We need to show that $\mathbb{P}[\mathbf{S}_i = S|\sigma_i] = \frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \cdot \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right]$. Conditioning on $\mathbf{L}_i$, we can rewrite $\mathbb{P}[\mathbf{S}_i = S|\sigma_i]$ as

$$\mathbb{P}[\mathbf{S}_i = S|\sigma_i] = \mathbb{E}_{L_i \sim \mathbf{L}_i|\sigma_i}[\mathbb{P}[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = L_i]]. \tag{5}$$

It, therefore, suffices to prove that

$$\mathbb{P}[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = L_i] = \frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \cdot \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right], \tag{6}$$

for both $L_i = 0$ and $L_i = 1$. For $L_i = 1$, the claim holds by Fact 13 since by definition, $L_i = 1$ means RECONSTRUCT(j) was called for some $j \leq i$.

We, therefore, focus on $L_i = 0$. We first observe that $\sigma_i, \mathbf{L}_i = 0$ implies $\mathbf{T}^- \geq i$. This is because $\mathbf{L}_i = 0$ means RECONSTRUCT(j) for any $j \leq i$ was not called. Therefore if $\mathbf{T}^-$ were strictly less than i, then $\mathbf{T}$ would equal $\mathbf{T}^-$ (because RECONSTRUCT can only be called for values upto $T + 1$ and if RECONSTRUCT is not called, then T does not change.) which is not possible since $\mathbf{T} \geq i$.
Since $\mathbf{T}^- \geq i$, we can condition on the value of the *previous history* of level i. More formally, define the random variable $\mathbf{H}_i^-$ as

$$\mathbf{H}_i^- := (\overline{\mathbf{R}}_0^-, \overline{\mathbf{R}}_1^-, \ldots, \overline{\mathbf{R}}_i^-, \mathbf{R}_0^-, \mathbf{R}_1^-, \ldots, \mathbf{R}_i^-, \mathbf{S}_1^-, \ldots, \mathbf{S}_{i-1}^-, \mathbf{m}_i^-).$$

By the law of iterated expectation,

$$\begin{aligned}\mathbb{P}[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0] &= \mathbb{P}\left[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0, \mathbf{T}^- \leq i\right] \\ &= \mathbb{E}_{H_i^- \sim \mathbf{H}_i^-|\sigma_i, \mathbf{L}_i = 0, \mathbf{T}^- \leq i}\left[\mathbb{P}\left[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0, \mathbf{T}^- \leq i, \mathbf{H}_i^- = H_i^-\right]\right]\end{aligned} \tag{7}$$

where the expectation is taken over all H_i^- with positive probability.

We now observe that conditioned on $\mathbf{T}^- \leq i, \mathbf{H}_i^- = H_i^-$, the value of $\mathbf{L}_i$ always equals 0. This is because L_i is a function of $(\overline{R}_0, \ldots \overline{R}_i, R_0, \ldots, R_i, D)$, which is determined by H_i. Note that D is deterministic since it contains all the deleted elements in the stream and is independent of our algorithm. Therefore, since we only consider H_i^- with positive probability, we can drop the conditioning on $\mathbf{L}_i = 0$ in the $\mathbb{P}\left[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0, \mathbf{T}^- \leq i, \mathbf{H}_i^- = H_i^-\right]$ term of (7) since it is redundant.
We can similarly drop σ_i. This is because as $\mathbf{L}_i = 0$, the value of $\mathbf{H}_i$ is deterministic conditioned on $\mathbf{H}_i^- = H_i^-$. Notably, the values of $\mathbf{R}_1, \ldots, \mathbf{R}_i$ are going to be $R_1^-, \ldots, R_i^-$. The same can be said for $\mathbf{S}_1, \ldots, \mathbf{S}_{i-1}$ and $\mathbf{m}_i$. As for $\overline{\mathbf{R}}_1, \ldots, \overline{\mathbf{R}}_i$, even though their value may be different from $\overline{R}_1^-, \ldots, \overline{R}_i^-$, it is *still deterministic* conditioned on $\mathbf{H}_i^- = H_i^-$ as the

decision to add elements in Line 6 is based on the values in H_i^- only. Given the above observation, we can rewrite $\mathbb{P}[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0]$ as

$$\mathbb{P}[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0] = \mathbb{E}_{H_i^- \sim \mathbf{H}_i^- | \sigma_i, \mathbf{L}_i = 0}\left[\mathbb{P}\left[\mathbf{S}_i = S|\mathbf{T}^- \leq i, \mathbf{H}_i^- = H_i^-\right]\right]$$

We can further replace $\mathbf{S}_i = S$ with $\mathbf{S}_i^- = S$ as $\mathbf{L}_i = 0$ implies $\mathbf{S}_i = \mathbf{S}_i^-$. Denoting the largest bucket in R_i^- with $R_i^{-,(b(i))}$, it follows that

$$\begin{aligned}
\mathbb{P}[\mathbf{S}_i = S|\sigma_i, \mathbf{L}_i = 0] &= \mathbb{E}_{H_i^- \sim \mathbf{H}_i^- | \sigma_i, \mathbf{L}_i = 0}\left[\mathbb{P}\left[\mathbf{S}_i^- = S|\mathbf{T}^- \leq i, \mathbf{H}_i^- = H_i^-\right]\right] \\
&\overset{(a)}{=} \mathbb{E}_{H_i^- \sim \mathbf{H}_i^- | \sigma_i, \mathbf{L}_i = 0}\left[\frac{1}{\binom{|R_i^{(-,b(i))}|}{m_i^-}} \cdot \mathbb{1}\left[S \subseteq R_i^{(-,b(i))} \wedge |S| = m_i^-\right]\right] \\
&\overset{(b)}{=} \mathbb{E}_{H_i^- \sim \mathbf{H}_i^- | \sigma_i, \mathbf{L}_i = 0}\left[\frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \cdot \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right]\right] \\
&= \frac{1}{\binom{|R_i^{(b(i))}|}{m_i}} \cdot \mathbb{1}\left[S \subseteq R_i^{(b(i))} \wedge |S| = m_i\right]
\end{aligned}$$

where for (a), we have used the induction assumption, and for (b) we have used the fact that $R_i = R_i^-$ and $m_i = m_i^-$ because of $\mathbf{L}_i = 0$. We have therefore proved (6) for both $L_i = 0$ and $L_i = 1$, which completes the proof given (5). □

A.2. Properties of sample size

In this section, we state the key properties of CALCSAMPLECOUNT that will be useful in our proofs. We begin by defining the notion of Suitable sample count (Definition 14), which captures the properties we require in our proofs. We then show that the output of CALCSAMPLECOUNT is suitable with high probability (Lemma 15).

Definition 14 (Suitable sample count). *Given R', G', τ' such that $R' \neq \emptyset$ and* FILTER$(R', G', \tau') = R'$*, define their* Suitable sample count *$M^*(R', G', \tau')$ as the set of all integers like m' such that $m' \geq 1$, and if we sample a set S' with $m' - 1$ elements from R' and an element s' from $R' \backslash S'$,*

$$\mathbb{E}\left[|\text{FILTER}(R', G' \cup S' \cup s', \tau')|\right] \leq (1 - \epsilon_{\text{sam}}/4)|R'| \tag{8}$$

and

$$\mathbb{P}\left[f(s'|G' \cup S') \geq \tau'\right] \geq (1 - 2\epsilon_{\text{sam}}) \tag{9}$$

Define the suitable sample count of level i as $M_i^* := M^*(R_i^{(b(i))}, G_{i-1}, \tau^{(i)})$.

Lemma 15. *Consider a call to* CALCSAMPLECOUNT(R', G', τ') *with values satisfying* FILTER$(R', G', \tau') = R'$ *and $R' \neq \emptyset$. The number of queries made by* CALCSAMPLECOUNT *is bounded by $O\left(|R'| \cdot \frac{\log(k)}{\epsilon_{\text{sam}}^3}\right)$. Furthermore, the output is a suitable sample count (Definition 14) with probability at least $1 - \frac{\epsilon_{\text{sam}}}{100k^{10}}$.*

Proof. To bound the number of queries, we note that there will be at most $\min\{|R'|, \log(k)\} \leq |R'|$ calls to REDUCEMEAN, and each call makes at most

$$O\left(\frac{4}{\epsilon_{\text{sam}}^2}\log(200k^{10}/\epsilon_{\text{sam}})\right) = O\left(\frac{1}{\epsilon_{\text{sam}}^2} \cdot \left(\log(k) + \log(\frac{1}{\epsilon_{\text{sam}}})\right)\right),$$

queries, which implies the lemma's statement.

We therefore focus on proving that the output is suitable with a probability of at least $1 - \frac{\epsilon_{\text{sam}}}{100k^{10}}$.

For a number m', define the value $I_{m'}$ as

$$\mathbb{P}\left[f(s'|G' \cup S') \geq \tau'\right].$$

where s', S' are sampled as in the Definition 14, i.e., S' has size $m' - 1$ and is sampled from R', and s' is an element sampled from $R' \backslash S'$. Observe that $I_{m'}$ is the expectation of the Bernoulli random variable $\mathbb{1}\left[f(s'|G' \cup S') \geq \tau'\right]$, while

ReduceMean(R', G', m') estimates this mean by repeated sampling. By Hoeffding's inequality, we conclude that for each call to ReduceMean(R', G', m'),

$$\begin{aligned}\mathbb{P}\left[|I_{m'} - \textsc{ReduceMean}(R', G', m')| \geq \epsilon_{\text{sam}}/2\right] &\leq 2e^{-\left\lceil \frac{4}{\epsilon_{\text{sam}}^2} \log(200k^{11}/\epsilon_{\text{sam}})\right\rceil \cdot (\frac{\epsilon_{\text{sam}}}{2})^2} \\ &\leq 2e^{-\frac{4}{\epsilon_{\text{sam}}^2} \log(200k^{11}/\epsilon_{\text{sam}}) \cdot (\frac{\epsilon_{\text{sam}}}{2})^2} \\ &\leq 2e^{-\log(200k^{11}/\epsilon_{\text{sam}})} \\ &\leq \frac{\epsilon_{\text{sam}}}{100k^{11}}\end{aligned}$$

The above formula is an upper bound on the probability that a single call to ReduceMean is off by at most $\epsilon_{\text{sam}}/2$. Since CalcSampleCount makes at most $M \leq k$ calls, a union bound implies that with probability at least $1 - \frac{\epsilon_{\text{sam}}}{100k^{10}}$, ReduceMean is off by at most $\frac{\epsilon_{\text{sam}}}{2}$ for *all* the calls made in CalcSampleCount. In other words,

$$\mathbb{P}\left[\forall m' \in [1, \min\{k - |G'|, |R'|\} : |I_{m'} - \textsc{ReduceMean}(R', G', m')| \geq \frac{\epsilon_{\text{sam}}}{2}\right] \leq \frac{\epsilon_{\text{sam}}}{100k^{10}}. \tag{10}$$

For the rest of the proof, we will show that whenever all of the calls to ReduceMean are off by at most $\frac{\epsilon_{\text{sam}}}{2}$, the output is guaranteed to be a suitable sample count. This implies the Lemma's statement because of (10).

If M is outputted in Line 10 of Algorithm 4, we conclude that $\mathbb{P}[I_M] \geq 1 - \frac{3}{2}\epsilon_{\text{sam}}$, which means M satisfies (9). Note however that M always satisfies (8) since Filter$(R', G' \cup S' \cup s', \tau')$ will always equal $\emptyset$.
We therefore assume that the output was some value $m' < M$ in Line 17 of Algorithm 4. We note that ReduceMean$(m') \geq 1 - \epsilon_{\text{sam}}$. This holds because all of the values that m can take in Algorithm 4 satisfy this and the output is m; the value $m = 1$ satisfies this cause $I_1 = 1$ and the others satisfy this given the condition for setting m. Since ReduceMean(m') was accurate within $\epsilon_{\text{sam}}/2$ and ReduceMean$(m') \geq 1 - \epsilon_{\text{sam}}$, it follows that $I_{m'} \geq 1 - \frac{3}{2}\epsilon_{\text{sam}}$, which implies (9).

It remains to show (8). Sampling S' and s' as before,

$$\begin{aligned}\mathbb{E}_{S',s'}\left[|\textsc{Filter}(R', G' \cup S' \cup s', \tau')|\right] &= \mathbb{E}_{S',s'}\left[\sum_{e \in R'} \mathbb{1}\left[e \in \textsc{Filter}(R', G' \cup S' \cup s', \tau')\right]\right] \\ &\overset{(a)}{=} \mathbb{E}_{S',s'}\left[\sum_{e \in R' \setminus (S' \cup s')} \mathbb{1}\left[e \in \textsc{Filter}(R', G' \cup S' \cup s', \tau')\right]\right]\end{aligned}$$

where for (a), we have used the fact that Filter$(A, B, \tau) \cap B$ is $\emptyset$ for any sets A and B. Letting e' be a random sample from $R' \setminus (S' \cup s')$, this implies that

$$\begin{aligned}\mathbb{E}_{S',s'}\left[|\textsc{Filter}(R', G' \cup S' \cup s, \tau')|\right] &= \mathbb{E}_{S',s'}\left[|R' \setminus (S' \cup s')| \cdot \mathbb{E}_{e'}\left[\mathbb{1}\left[e' \in \textsc{Filter}(R', G' \cup S' \cup s', \tau')\right]\right]\right] \\ &\leq |R'| \cdot \mathbb{E}_{S',s'}\left[\mathbb{E}_{e'}\left[\mathbb{1}\left[e' \in \textsc{Filter}(R', G' \cup S' \cup s', \tau')\right]\right]\right] \\ &= |R'| \cdot \mathbb{P}_{S',s',e'}\left[e' \in \textsc{Filter}(R', G' \cup S' \cup s', \tau')\right] \\ &\overset{(a)}{=} |R'| \cdot \mathbb{P}_{S',s',e'}\left[f(e'|G' \cup S' \cup s') \geq \tau'\right]\end{aligned}$$

where for (a), we have used the fact that since $m' < M \leq k - |G'|$, we have $|G' \cup S' \cup s'| < k$, which implies $e' \in$ Filter$(R', G' \cup S' \cup s', \tau')$ is equivalent to $f(e'|G' \cup S' \cup s') \geq \tau'$. Note however that since $S' \cup s'$ is a random subset of size m' and e' is a random sample of $R' \setminus (S' \cup s')$, we have $\mathbb{P}_{S',s',e'}[f(e|G' \cup S' \cup s') \geq \tau']$ equals $I_{m'+1}$ (note that $m' + 1 \leq M$ because we had assumed earlier that $m' < M$). Since $m' \neq M$, we conclude that ReduceMean$(R', G', m' + 1) \leq 1 - \epsilon_{\text{sam}}$, which implies $I_{m'+1} \leq 1 - \frac{\epsilon_{\text{sam}}}{2}$, which implies (8). □

The above lemma effectively shows that $m_i \in M_i^*$ holds with high probability *right after* Reconstruct(j) is called for some $j \leq i$. To prove our approximation guarantee, we will need a stronger result that shows this holds at an arbitrary point in the stream. Before stating the result, we define the pre-count history of a level as, effectively, its history without m_i.

Definition 16 (Pre-count History). *We define the* Pre-count History *of Level i as*

$$H_i^{-m} := (\overline{R}_0, \overline{R}_1, \ldots, \overline{R}_i, R_0, R_1, \ldots, R_i, S_1, \ldots, S_{i-1}). \tag{11}$$

Note that this is similar to the definition of H_i in (3)*, with the difference that m_i is no longer included. Analogously, we define the random variable $\mathbf{H}_i^{-m}$ as*

$$\mathbf{H}_i^{-m} := (\overline{\mathbf{R}}_0, \overline{\mathbf{R}}_1, \ldots, \overline{\mathbf{R}}_i, \mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_i, \mathbf{S}_1, \ldots, \mathbf{S}_{i-1}).$$

Lemma 17. *At any point in the stream, for any $i \geq 1$,*

$$\mathbb{P}\left[\mathbf{m}_i \notin M_i^* | \mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] \leq \frac{\epsilon_{\text{sam}}}{100k^{10}}$$

Proof. The proof follows in the same manner as Lemma 12.

We prove the claim by induction on the update stream. Initially, the claim holds trivially since $\mathbf{T} \geq i$ is impossible. Assuming the lemma's statement holds before an update, we will show that it holds for the new values as well. Let the superscript $^-$ denote the values before the insertion, e.g., T^- denotes the number of levels before insertion, and T denotes the number of levels after insertion.

As before, let $\mathbf{L}_i$ denote a random variable that is set to 1 if RECONSTRUCT(j) is called for some $j \leq i$ and set to 0 otherwise. We will show that

$$\mathbb{P}\left[\mathbf{m}_i \notin M_i^* | \mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}, \mathbf{L}_i = L_i\right] \leq \frac{\epsilon_{\text{sam}}}{100k^{10}}$$

for both $L_i = 0$ and $L_i = 1$. For $L_i = 1$, the claim follows from Lemma 15. As for $L_i = 0$, we note that this implies $\mathbf{T}^- \geq i$ because if $\mathbf{T}^-$ were $< i$, then $\mathbf{T}$ would have been equal to $\mathbf{T}^-$ because $\mathbf{L}_i = 0$ means that RECONSTRUCT(j) was not invoked for any $j \leq i$. Let $(H_i^{-m})^-$ denote the value of pre-count history before the update. By the law of iterated expectation, it suffices to show

$$\mathbb{P}\left[\mathbf{m}_i \notin M_i^* | \mathbf{T} \geq i, \mathbf{T}^- \geq i, \mathbf{H}_i^{-m} = H_i^{-m}, (\mathbf{H}_i^{-m})^- = (H_i^{-m})^-, \mathbf{L}_i = 0\right] \leq \frac{\epsilon_{\text{sam}}}{100k^{10}} \tag{12}$$

for all $(H_i^{-m})^-$ that have positive probability conditioned on $\mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}, \mathbf{L}_i = 0$. As before, we can drop the $\mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}, \mathbf{L}_i = 0$ from the condition since they are implied by $(\mathbf{H}_i^{-m})^- = (H_i^{-m})^-$. We can further replace $\mathbf{m}_i \notin M_i^*$ with $\mathbf{m}_i^- \notin (M_i^*)^-$, where $(M_i^*)^-$ is the set of suitable sample counts for level i before the update, as determined by $(H_i^{-m})^-$. This is because both $\mathbf{m}_i$ and M_i^* are unchanged through the update (note that M_i^* is a function of R_i, G_{i-1} and $(R_i, G_{i-1}) = (R_i^-, G_{i-1}^-)$). This changes (12) to the induction hypothesis, which finishes the proof. □

A.3. Approximation guarantee

In this section, we prove the approximation guarantee of our algorithm.

Our analysis is based on a novel relaxation function that may be of independent interest. We first introduce the relaxation function f' and show that it is a lower bound on $f(G_T \backslash D)$ (Lemma 18). We then bound the contribution of each level to the relaxation function (Lemmas 19 and 20), to prove the desired $\frac{\text{OPT}}{2} - \epsilon$ bound.

We begin with some definitions. For $1 \leq i \leq T$, define the quantities f_i and d_i as

$$f_i := f(S_i | G_{i-1}), \quad \tau^{(i)} := (1 + \epsilon_{\text{buck}})^{b(i)} \cdot \tau, \quad d_i := |D \cap S_i| \cdot (1 + \epsilon_{\text{buck}}) \cdot \tau^{(i)}.$$

We note that

$$f(G_T) = \sum_{i=1}^{T} (f(G_i) - f(G_{i-1})) = \sum_{i=1}^{T} f(S_i | G_{i-1}) = \sum_{i=1}^{T} f_i. \tag{13}$$

We can also bound $f(G_T \backslash D)$ in terms of $f(G_T)$ and d_i as

$$\begin{aligned} f(G_T \backslash D) = \sum_{i=1}^{T} \left(f(G_i \backslash D) - f(G_{i-1} \backslash D) \right) &= \sum_{i=1}^{T} f(S_i \backslash D | G_{i-1} \backslash D) \\ &\geq \sum_{i=1}^{T} f(S_i \backslash D | G_{i-1}) \\ &\overset{(a)}{\geq} \sum_{i=1}^{T} \left(f(S_i | G_{i-1}) - f(D \cap S_i | G_{i-1}) \right) \\ &\geq \sum_{i=1}^{T} f(S_i | G_{i-1}) - \sum_{i=1}^{T} \sum_{e \in D \cap S_i} f(e | G_{i-1}) \\ &\overset{(b)}{\geq} \sum_{i=1}^{T} f_i - \sum_{i=1}^{T} d_i \end{aligned} \tag{14}$$

where the first three inequalities follow from submodularity (in particular, (a) follows from the fact that $f(A \backslash B) \geq f(A) - f(B)$ for any sets A, B) and (b) follows from the fact that $f(e|G_{i-1}) \leq (1 + \epsilon_{\text{buck}}) \cdot \tau^{(i)}$ for all $e \in S_i$.

For $i \in [1, T+1]$, Define f'_i as

$$f'_i = \begin{cases} \min\{f_i, m_i \cdot \tau^{(i)}\} - d_i & \text{If} \quad i \leq T \\ (1 - 4\epsilon_{\text{sam}}) \left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}} \frac{\text{OPT}}{2} - \sum_{j \leq T} m_j \cdot \tau^{(j)} \right) & \text{If} \quad i = T+1 \end{cases}$$

Our next lemma shows that $\sum_{i=1}^{T+1} f'_i$ is a lower bound for the value of output.

Lemma 18. *Assume that*

$$f(G_{\text{opt}}) \leq OPT \leq (1 + \epsilon_{\text{opt}}) f(G_{\text{opt}}).$$

Then

$$f(G_T \backslash D) \geq \sum_{i=1}^{T+1} f'_i.$$

Proof. We divide the proof into two cases, depending on whether or not $f'_{T+1} > 0$.

Case 1: $f'_{T+1} \leq 0$. In this case, by (14),

$$f(G_T \backslash D) \geq \sum_{i=1}^{T} f_i - \sum_{i=1}^{T} d_i \geq \sum_{i=1}^{T} \left(\min\{f_i, m_i \cdot \tau^{(i)}\} - d_i \right) = \sum_{i=1}^{T} f'_i \geq \sum_{i=1}^{T+1} f'_i \ .$$

Where the last inequality uses the assumption $f'_{T+1} \leq 0$.

Case 2: $f'_{T+1} > 0$. We first claim that $|G_T| < k$:

$$f'_{T+1} > 0 \implies \frac{1 - \epsilon_{\text{opt}}}{1 + \epsilon_{\text{opt}}} \frac{\text{OPT}}{2} - \sum_{j=1}^{T} m_j \cdot \tau^{(j)} > 0. \tag{15}$$

which further implies that

$$k\tau \overset{(a)}{=} \frac{\text{OPT}}{2} \geq \frac{1 - \epsilon_{\text{opt}}}{1 + \epsilon_{\text{opt}}} \frac{\text{OPT}}{2} \overset{(b)}{>} \sum_{j=1}^{T} m_j \cdot \tau^{(j)} \geq \sum_{j=1}^{T} m_j \cdot \tau = |G_T| \cdot \tau.$$

Here, we have used the definition of τ for (a) and (15) for (b). Therefore, $|G_T| < k$ as claimed.

Consider an element $s \in G_{\text{opt}}$. By Lemma 8, $\widehat{R}_{T+1} = \emptyset$ and therefore $s \notin \widehat{R}_{T+1}$. Since $s \in \widehat{R}_0$, there is an index $i \in [0, T]$ such that $s \in \widehat{R}_i$ but $s \notin \widehat{R}_{i+1}$. By Lemma 5, this means that either $f(s|G_i) < \tau$ or $|G_i| = k$. Since $|G_i| \le |G_T| < k$, we conclude that $f(s|G_i) < \tau$, which by submodularity implies $f(s|G_T) < \tau$. Now note that

$$\frac{\text{OPT}}{1+\epsilon_{\text{opt}}} \le f(G_{\text{opt}}) \le f(G_T \cup G_{\text{opt}}) \le f(G_T) + \sum_{s^* \in G_{\text{opt}}} f(s^*|G_T) \le f(G_T) + k \cdot \tau = f(G_T) + \frac{\text{OPT}}{2} \ .$$

Rearranging and using (13), we obtain

$$\sum_{i=1}^{T} f_i = f(G_T) \ge \frac{\text{OPT}}{1+\epsilon_{\text{opt}}} - \frac{\text{OPT}}{2} = \frac{\text{OPT}(2-(1+\epsilon_{\text{opt}}))}{2(1+\epsilon_{\text{opt}})} = \frac{\text{OPT}}{2} \cdot (\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}).$$

Since in (14) we show that $f(G_T \backslash D) \ge \sum_{i=1}^T f_i - \sum_{i=1}^T d_i$, this implies that

$$\begin{aligned}
f(G_T \backslash D) &\ge \frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}} \frac{\text{OPT}}{2} - \sum_{i=1}^{T} d_i \\
&\overset{(a)}{=} \sum_{i=1}^{T} \Big(m_i \cdot \tau^{(i)} - d_i \Big) + \left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}} \frac{\text{OPT}}{2} - \sum_{i=1}^{T} m_i \cdot \tau^{(i)} \right) \\
&\ge \sum_{i=1}^{T} \Big(\min\{f_i, m_i \cdot \tau^{(i)}\} - d_i \Big) + \left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}} \frac{\text{OPT}}{2} - \sum_{i=1}^{T} m_i \cdot \tau^{(i)} \right) \\
&\overset{(b)}{\ge} \sum_{i=1}^{T} \Big(\min\{f_i, m_i \cdot \tau^{(i)}\} - d_i \Big) + (1-4\epsilon_{\text{sam}}) \left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}} \frac{\text{OPT}}{2} - \sum_{i=1}^{T} m_i \cdot \tau^{(i)} \right) \\
&= \sum_{i=1}^{T+1} f_i'
\end{aligned}$$

where for (a) we have just added and subtracted $\sum_{i=1}^T m_i \cdot \tau^{(i)}$, and for (b), we have used the assumption $f'_{T+1} > 0$. □

Lemma 19. *For all i, if $m_i \in M_i^*$, then*

$$\mathbb{E}\left[f_i' \middle| \mathbf{T} \ge i, \mathbf{H}_i = H_i\right] \ge (1-3\epsilon_{\text{sam}}) \cdot m_i \cdot \tau^{(i)}$$

for all H_i such that $\mathbb{P}[\mathbf{T} \ge i, \mathbf{H}_i = H_i] > 0$.

Proof. Since $\mathbb{P}[\mathbf{T} \ge i, \mathbf{H}_i = H_i] > 0$, we conclude that $R_i \neq \emptyset$. By Lemma 12, S_i is a uniform sample of size m_i from $R_i^{b(i)}$. Let $s_{i,1}, \ldots, s_{i,m_i}$ be a random ordering of S_i and let $S_{i,<j}$ and $S_{i,>j}$ denote $\{s_{i,1}, \ldots, s_{i,j-1}\}$ and $\{s_{i,j+1}, \ldots, s_{i,m_i}\}$ respectively. It follows that

$$\begin{aligned}
f_i &= \sum_{j=1}^{m_i} f(s_{i,j}|G_{i-1} \cup S_{i,<j}) \\
&\ge \sum_{j=1}^{m_i} f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j}) \\
&\ge \sum_{j=1}^{m_i} \min\{\tau^{(i)}, f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j})\}
\end{aligned}$$

It is also clear that

$$m_i \cdot \tau^{(i)} \ge \sum_{j=1}^{m_i} \min\{\tau^{(i)}, f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j})\}$$

As before, using the shorthand σ_i to denote $\mathbf{T} \geq i, \mathbf{H}_i = H_i$,

$$\begin{aligned}
\mathbb{E}\left[\min\{f_i, m_i\tau^{(i)}\}\middle|\sigma_i\right] &\geq \sum_{j=1}^{m_i} \mathbb{E}\left[\min\{\tau^{(i)}, f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j})\}\middle|\sigma_i\right] \\
&\geq \sum_{j=1}^{m_i} \mathbb{E}\left[\tau^{(i)}\mathbb{1}\left[f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j}) \geq \tau^{(i)}\right]\middle|\sigma_i\right] \\
&= \sum_{j=1}^{m_i} \tau^{(i)}\mathbb{E}\left[\mathbb{1}\left[f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j}) \geq \tau^{(i)}\right]\middle|\sigma_i\right] \\
&= \sum_{j=1}^{m_i} \tau^{(i)}\mathbb{P}\left[f(s_{i,j}|G_{i-1} \cup S_{i,<j} \cup S_{i,>j}) \geq \tau^{(i)}\middle|\sigma_i\right] \\
&\overset{(a)}{=} m_i \cdot \tau^{(i)} \cdot \mathbb{P}\left[f(s_{i,m_i}|G_{i-1} \cup S_{i,<m_i}) \geq \tau^{(i)}\middle|\sigma_i\right]
\end{aligned}$$

where (a) follows from the fact that $s_{i,1}, \ldots, s_{i,m_i}$ was a random permutation.
We now note that by Lemma 12, $S_{i,<m_i}$ is a random subset of size $m_i - 1$ from $R_i^{(b(i))}$ and s_{i,m_i} is a random element in $R_i^{(b(i))} \backslash S_{i,<m_i}$. Therefore, given the assumption $m_i \in M_i^*$, (9) implies that

$$\mathbb{P}\left[f(s_{i,m_i}|G_{i-1} \cup S_{i,<m_i}) \geq \tau^{(i)}\middle|\sigma_i\right] \geq 1 - 2\epsilon_{\text{sam}}.$$

Therefore,

$$\mathbb{E}\left[\min\{f_i, m_i\tau^{(i)}\}\middle|\sigma_i\right] \geq (1 - 2\epsilon_{\text{sam}}) \cdot m_i \cdot \tau^{(i)}. \tag{16}$$

Furthermore, given the definition of d_i, we have

$$\begin{aligned}
\mathbb{E}\left[d_i|\sigma_i\right] &= (1 + \epsilon_{\text{buck}}) \cdot \tau^{(i)} \cdot \mathbb{E}\left[|S_i \cap D|\middle|\sigma_i\right] \\
&\overset{(a)}{=} (1 + \epsilon_{\text{buck}}) \cdot \tau^{(i)} \cdot \frac{|R_i^{(b(i))} \cap D|}{|R_i^{(b(i))}|} m_i \\
&\overset{(b)}{\leq} (1 + \epsilon_{\text{buck}}) \cdot \tau^{(i)} \cdot m_i \cdot \epsilon_{\text{del}}.
\end{aligned} \tag{17}$$

where (a) follows from the uniform sample assumption of Lemma 12, and (b) follows from the restriction $|R_i^{(b(i))} \cap D| \leq \epsilon_{\text{del}} \cdot |R_i^{(b(i))}|$ given in Lemma 7. We note however that $(1 + \epsilon_{\text{buck}}) \cdot \epsilon_{\text{del}} \leq \epsilon_{\text{sam}}$ because of the way the parameters were set in Section 3.6. Therefore, the claim follows from (16) and (17) by the linearity of expectation:

$$\mathbb{E}\left[\min\{f_i, m_i\tau^{(i)}\} - d_i\middle|\sigma_i\right] \geq (1 - 2\epsilon_{\text{sam}}) \cdot m_i \cdot \tau^{(i)} - \epsilon_{\text{sam}} \cdot \tau^{(i)} \cdot m_i = (1 - 3\epsilon_{\text{sam}}) \cdot m_i \cdot \tau^{(i)}.$$

□

Lemma 20. *For any value $i \geq 1$*[5]*,*

$$\mathbb{E}\left[\sum_{j=i}^{\mathbf{T}+1} f'_j \middle| \mathbf{T} + 1 \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] \geq (1 - 4\epsilon_{\text{sam}}) \left(\frac{1 - \epsilon_{\text{opt}}}{1 + \epsilon_{\text{opt}}} \frac{OPT}{2} - \sum_{j<i} m_j \cdot \tau^{(j)}\right)$$

for all H_i^{-m} such that $\mathbb{P}\left[\mathbf{T} + 1 \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] > 0$.

Proof. We divide the proof into two parts. In the first part, we prove the result when $i = T + 1$. Note that we can determine whether $i = T + 1$ by examining H_i^{-m}, as $i = T + 1$ if and only if $R_i = \emptyset$. In part 2, we extend this result to the general case using induction.

[5]Note that we do not impose the restriction $i \leq T + 1$ when specifying the range for i since, effectively, this is done by conditioning on the event $\mathbf{T} + 1 \geq i$.

Part 1. If $i = T+1$, then by definition of f'_{T+1},

$$\mathbb{E}\left[\sum_{j=i}^{T+1} f'_j \middle| \mathbf{T}+1 \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] = f'_{T+1} = (1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2} - \sum_{j<i} m_j \cdot \tau^{(j)}\right)$$

We, therefore, focus on the case of $i < T+1$.

Part 2 We prove the claim by a backward induction on i. We note that $T \leq k$ because we always have at least one sample in $|S_i|$ as long as $R_i \neq \emptyset$. Therefore, if $i > k+1$, then $\mathbb{P}\left[\mathbf{T}+1 \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] = 0$ and there is nothing to prove. If $i = k+1$, then given $\mathbb{P}\left[\mathbf{T}+1 \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] > 0$ we must have $i = T+1$, and the claim follows from Part 1. Assume the claim holds for $i+1$, we prove it holds for i as well. Since in this part, we assumed $i \neq T+1$ we can conclude

$$\left\{\, \mathbf{T}+1 \geq i, \mathbf{H}_i^{-m} = H_i^{-m} \,\right\} = \left\{\, \mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m} \,\right\}.$$

We first give a sketch of the proof. By Lemma 17, we can expect $m_i \in M_i^*$ with high probability. As long as $m_i \in M_i^*$, by Lemma 19,

$$\mathbb{E}\left[f'_i | \sigma_i\right] > (1-4\epsilon_{\text{sam}}) \cdot m_i \tau^{(i)}.$$

Furthermore, we can utilize the induction hypothesis to conclude that

$$\mathbb{E}\left[\sum_{j \geq i+1} f'_j \middle| \sigma_i\right] \geq (1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2} - \sum_{j<i+1} m_j \cdot \tau^{(j)}\right)$$

Adding the two expressions above results in the desired bound. To prove the lemma's statement, we formalize the above sketch while also carefully considering the case of $m_i \notin M_i^*$.

Let the shorthand σ_i^{-m} denote $\mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}$. By the law of total expectation,

$$\begin{aligned}
\mathbb{E}\left[\sum_{j \geq i} f'_j \middle| \sigma_i^{-m}\right] &= \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}}\left[\mathbb{E}\left[\sum_{j \geq i} f'_j \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right] \\
&= \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}}\left[\mathbb{E}\left[f'_i + \sum_{j \geq i+1} f'_j \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right] \\
&= \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}}\left[\mathbb{E}\left[f'_i \mathbb{1}\left[m_i \in M_i^*\right] + f'_i \mathbb{1}\left[m_i \notin M_i^*\right] + \sum_{j \geq i+1} f'_j \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right] \qquad (18)
\end{aligned}$$

Note that $\left\{\, \sigma_i^{-m}, \mathbf{m}_i = m_i \,\right\}$ is the same as $\left\{\, \mathbf{T} \geq i, \mathbf{H}_i = H_i \,\right\}$. By Lemma 19, if $m_i \in M_i^*$, and $\mathbb{P}\left[\mathbf{m}_i = m_i \mid \sigma_i^{-m}\right] > 0$, then

$$\mathbb{E}\left[f'_i \middle| \mathbf{T} \geq i, \mathbf{H}_i = H_i\right] \geq (1-3\epsilon_{\text{sam}}) \cdot m_i \cdot \tau^{(i)}$$

Therefore,

$$\mathbb{E}\left[f'_i \cdot \mathbb{1}\left[m_i \in M_i^*\right] \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right] \geq (1-3\epsilon_{\text{sam}}) m_i \cdot \tau^{(i)} \cdot \mathbb{1}\left[m_i \in M_i^*\right]$$

Furthermore, if $m_i \notin M_i^*$,

$$\mathbb{E}\left[f'_i \middle| \mathbf{T} \geq i, \mathbf{H}_i = H_i\right] \geq -d_i \geq -(1+\epsilon_{\text{buck}}) \cdot \tau^{(i)} \cdot m_i \geq -2k \cdot \tau^{(i)},$$

where the first inequality follows from the fact that $\min\{f_i, m_i\tau^{(i)}\} \geq 0$ and the last inequality follows from $m_i \leq k$ and $1 + \epsilon_{\text{buck}} \leq 2$. Therefore,

$$\mathbb{E}\left[f'_i \cdot \mathbb{1}\left[m_i \notin M_i^*\right] \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right] \geq -2k \cdot \tau^{(i)} \cdot \mathbb{1}\left[m_i \notin M_i^*\right].$$

Plugging the above inequalities in (18), we obtain

$$\begin{aligned}
&\mathbb{E}\left[\sum_{j\geq i} f'_j \middle| \sigma_i^{-m}\right] \\
&\geq \mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[(1-3\epsilon_{\text{sam}})\cdot m_i\cdot\tau^{(i)}\cdot\mathbb{1}\left[m_i\in M_i^*\right]-2k\cdot\tau^{(i)}\mathbb{1}\left[m_i\notin M_i^*\right]+\mathbb{E}\left[\sum_{j\geq i+1} f'_j\middle|\sigma_i^{-m},\mathbf{m}_i=m_i\right]\right]
\end{aligned}$$

Define B as

$$B := \epsilon_{\text{sam}}\cdot m_i\cdot\tau^{(i)}\cdot\mathbb{1}\left[m_i\in M_i^*\right]-2k\cdot\tau^{(i)}\cdot\mathbb{1}\left[m_i\notin M_i^*\right]-(1-4\epsilon_{\text{sam}})\cdot m_i\cdot\tau^{(i)}\cdot\mathbb{1}\left[m_i\notin M_i^*\right].$$

It follows that

$$\begin{aligned}
\mathbb{E}\left[\sum_{j\geq i} f'_j \middle| \sigma_i^{-m}\right] &\geq \mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[(1-4\epsilon_{\text{sam}})\cdot m_i\cdot\tau^{(i)}+B+\mathbb{E}\left[\sum_{j\geq i+1} f'_j\middle|\sigma_i^{-m},\mathbf{m}_i=m_i\right]\right] \\
&\geq \mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[(1-4\epsilon_{\text{sam}})\cdot m_i\cdot\tau^{(i)}+\mathbb{E}\left[\sum_{j\geq i+1} f'_j\middle|\sigma_i^{-m},\mathbf{m}_i=m_i\right]\right]+\mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[B\right]
\end{aligned}$$

For the first term, we observe that

$$\begin{aligned}
\mathbb{E}\left[\sum_{j\geq i+1} f'_j\middle|\sigma_i^{-m},\mathbf{m}_i=m_i\right] &= \mathbb{E}_{S_i\sim\mathbf{S}_i|\sigma_i^{-m},\mathbf{m}_i=m_i}\left[\mathbb{E}\left[\sum_{j\geq i+1} f'_j\middle|\sigma_i^{-m},\mathbf{m}_i=m_i,\mathbf{S}_i=S_i\right]\right] \\
&\overset{(a)}{\geq} \mathbb{E}_{S_i\sim\mathbf{S}_i|\sigma_i^{-m},\mathbf{m}_i=m_i}\left[(1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2}-\sum_{j<i+1} m_j\cdot\tau^{(j)}\right)\right] \\
&= (1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2}-\sum_{j<i+1} m_j\cdot\tau^{(j)}\right)
\end{aligned}$$

where for (a), we have used the induction hypothesis together with iterated expectation. Formally, assume that in addition to $\left\{\sigma_i^{-m},\mathbf{m}_i=m_i,\mathbf{S}_i=S_i\right\}$, we further condition on the value of $\overline{R}_{i+1}, R_{i+1}$. Then we have conditioned on

$$\left\{\sigma_i^{-m},\mathbf{m}_i=m_i,\mathbf{S}_i=S_i,\overline{\mathbf{R}}_{i+1}=\overline{R}_{i+1},\mathbf{R}_{i+1}=R_{i+1}\right\},$$

which is the same as conditioning on $\left\{\mathbf{T}\geq i, H_{i+1}^{-m}\right\}$, which is the same as conditioning on $\left\{\mathbf{T}+1\geq i+1, H_{i+1}^{-m}\right\}$.

Therefore,

$$\begin{aligned}
&\mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[(1-4\epsilon_{\text{sam}})\cdot m_i\cdot\tau^{(i)}+\mathbb{E}\left[\sum_{j\geq i+1} f'_j\middle|\sigma_i^{-m},\mathbf{m}_i=m_i\right]\right] \\
&\geq \mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[(1-4\epsilon_{\text{sam}})\cdot m_i\cdot\tau^{(i)}+(1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2}-\sum_{j<i+1} m_j\cdot\tau^{(j)}\right)\right] \\
&= \mathbb{E}_{m_i\sim\mathbf{m}_i|\sigma_i^{-m}}\left[(1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2}-\sum_{j<i} m_j\cdot\tau^{(j)}\right)\right] \\
&= (1-4\epsilon_{\text{sam}})\left(\frac{1-\epsilon_{\text{opt}}}{1+\epsilon_{\text{opt}}}\frac{\text{OPT}}{2}-\sum_{j<i} m_j\cdot\tau^{(j)}\right)
\end{aligned}$$

where the first inequality follows from (19), To finish the proof, it suffices to show that

$$\mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}} [B] \geq 0$$

Note however that, since we assumed $\mathbf{T} \geq i$, we can assume that $m_i > 0$ by definition of M_i^*. By definition of B,

$$\begin{aligned} B &= \left(\epsilon_{\text{sam}} \cdot m_i \cdot \tau^{(i)} \cdot \mathbb{1}[m_i \in M_i^*]\right) - \left(2k \cdot \tau^{(i)} \cdot \mathbb{1}[m_i \notin M_i^*]\right) - \left((1 - 4\epsilon_{\text{sam}}) \cdot m_i \cdot \tau^{(i)} \cdot \mathbb{1}[m_i \notin M_i^*]\right) \\ &\geq \tau^{(i)} \cdot \left(\mathbb{1}[m_i \in M_i^*] \cdot (\epsilon_{\text{sam}}) - \mathbb{1}[m_i \notin M_i^*] \cdot (2k + k)\right) \\ &\geq \tau^{(i)} \cdot \left(\mathbb{1}[m_i \in M_i^*] \cdot (\epsilon_{\text{sam}}) - \mathbb{1}[m_i \notin M_i^*] \cdot (3k)\right) \end{aligned}$$

where the first inequality follows from $m_i \leq k$. This further implies

$$\begin{aligned} \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}} \left[B \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right] &\geq \tau^{(i)} \cdot \left(\mathbb{P}\left[\mathbf{m}_i \in M_i^* | \sigma_i^{-m}\right] \cdot \epsilon_{\text{sam}} - 3k\mathbb{P}\left[\mathbf{m}_i \notin M_i^* | \sigma_i^{-m}\right]\right) \\ &\overset{(a)}{\geq} \tau^{(i)} \left(\frac{\epsilon_{\text{sam}}}{2} - \frac{3\epsilon_{\text{sam}}}{100}\right) > 0 \end{aligned}$$

Where (a) follows from Lemma 17 and the fact that $1 - \epsilon_{\text{sam}}/100k^{10} > 1/2$. $\square$

Setting $i = 1$ in the above Lemma completes the proof of Theorem 2 as

$$\mathbb{E}[f(G_T \backslash D)] \overset{(a)}{\geq} \mathbb{E}\left[\sum_{i=1}^{\mathbf{T}+1} f_i'\right] \geq (1 - 4\epsilon_{\text{sam}}) \cdot \frac{1 - \epsilon_{\text{opt}}}{1 + \epsilon_{\text{opt}}} \frac{\text{OPT}}{2}$$

where (a) follows from Lemma 18.

A.4. Query complexity

In this section, we analyze the query complexity of our algorithm. First, In section A.4.1, we analyze the query complexity of each call to RECONSTRUCT as it is the main building block of our algorithm. Next, in section A.4.2, we show how to utilize this result for bounding the expected amortized query complexity of our algorithm. Throughout the section, we use V as the ground set containing all of the elements of the stream.

We start by defining a quantity that will be important in our proofs.

Definition 21 (Potential). *For any $i \geq 1$ and any element $e \in \widehat{R}_i$, we define the element's* potential with respect to level i, *denoted by $P(e, i)$, as the single number satisfying $\frac{f(e|G_{i-1})}{\tau} \in [(1 + \epsilon_{\text{buck}})^{P(e,i)-1}, (1 + \epsilon_{\text{buck}})^{P(e,i)})$. For elements $e \notin \widehat{R}_i$, we define $P(e, i)$ to be zero. Additionally, we define the* potential of level i *as $P_i := \sum_{e \in V} P(e, i) = \sum_{e \in \widehat{R}_i} P(e, i)$. We also define P_i for $i > T + 1$ to be 0.*

We will use $\mathbf{P}_i$ instead of P_i when we want to emphasize the fact that these values are random.

Lemma 22. *The potential P satisfies the following properties for all $i \geq 1$:*

$$\forall e \in \widehat{R}_i : P(e, i) \in \left[1, \log_{1+\epsilon_{\text{buck}}}(4k)\right] \tag{19}$$

$$|\widehat{R}_i| \leq P_i \leq |\widehat{R}_i| \cdot \log_{1+\epsilon_{\text{buck}}}(4k) \tag{20}$$

$$P_{T+1} = 0 \quad \textit{and} \quad \forall i \leq T : P_i > 0 \tag{21}$$

Proof. For the first result, note that since $e \in \widehat{R}_i$, given Lemma 5, we have $e \in \text{FILTER}(\widehat{R}_{i-1}, G_{i-1}, \tau)$, which implies $f(e|G_{i-1}) \geq \tau$. Therefore, $P(e, i) \geq 1$. On the other hand, we note that since $e \in \widehat{R}_i$, we have $e \notin D$. Therefore,

$$2k\tau = \text{OPT} \geq f(G_{\text{opt}}) \geq f(e) \geq f(e|G_{i-1}).$$

It follows that $\frac{f(e|G_{i-1})}{\tau} \leq 2k$. Note however that, by definition,

$$\frac{f(e|G_{i-1})}{\tau} \geq (1 + \epsilon_{\text{buck}})^{P(e,i)-1} \implies P(e, i) \leq \log_{1+\epsilon_{\text{buck}}}\left(\frac{f(e|G_{i-1})}{\tau}\right) + 1.$$

Therefore,

$$P(e,i) \leq \log_{1+\epsilon_{\text{buck}}}(2k) + 1 \leq \log_{1+\epsilon_{\text{buck}}}(2k) + \log_{1+\epsilon_{\text{buck}}}(2) = \log_{1+\epsilon_{\text{buck}}}(4k).$$

The second equation in the claim follows from the first one since $P(e,i) = 0$ for $e \notin \widehat{R}_i$.

The third equation in the claim follows from the second equation and Lemma 8. □

Lemma 23. *For any $i \in [1,T]$, and any $e \in V$, $P(e,i) \geq P(e,i+1)$ and therefore, $P_i \geq P_{i+1}$.*

Proof. If $e \notin \widehat{R}_{i+1}$, the claim holds trivially cause the right-hand side will equal zero.

Otherwise, we note that $e \in \widehat{R}_i$ because $\widehat{R}_{i+1} \subseteq \widehat{R}_i$ given Lemma 5. The claim now follows from the fact that $G_i \subseteq G_{i+1}$. □

A.4.1. RECONSTRUCTION

In this section, we bound the query complexity of invoking RECONSTRUCT. It is clear from Algorithm 1 that RECONSTRUCT works by sampling S_i from R_i, forming $R_{i+1} = \text{FILTER}(R_i, G_i)$, and repeating the process with $i \leftarrow i+1$. We can therefore bound its query complexity by bounding

1. The number of times the while-loop is executed. We will obtain this result in Lemma 26 after bounding the decrease in the potential of each level in Lemma 24

2. The number of queries inside the while-loop. We do this using Lemma 15.

We use the above results to establish a bound on the total number of queries made by RECONSTRUCT in Lemma 27.

As we want to state results that hold for an arbitrary level i, throughout this section, we will assume that we call RECONSTRUCT(j) with a value of $\overline{R}_j$ satisfying FILTER$(R_j, G_{j-1}, \tau) = R_j = \overline{R}_j$. Given Lemma 5, this assumption is going to be valid in our algorithm. We note that since we are considering the data structure right after calling RECONSTRUCT, the values R_i and $\overline{R}_i$ and $\widehat{R}_i$ will be the same for $i \geq j$ after the call.

Lemma 24. *Assume we are given sets R_j, G_{j-1} satisfying* FILTER$(R_j, G_{j-1}, \tau) = R_j$*, and we invoke* RECONSTRUCT(j) *with $(\overline{R}_j, D)$ set to $(R_j, \emptyset)$, obtaining the (random) values $\mathbf{T}, \mathbf{S}_j, \dots \mathbf{S}_{\mathbf{T}}, \mathbf{R}_{j+1}, \dots \mathbf{R}_{\mathbf{T}+1}$. If $\epsilon_{\text{sam}} < \frac{1}{4}$, for each $i \geq j$,*

$$\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] \geq \frac{\epsilon_{\text{sam}}}{8} \cdot |R_i^{(b(i))}|$$

for all H_i^{-m} such that $\mathbb{P}\left[\mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] > 0$, where H_i^{-m} is defined as in Definition 16.

Proof. We first observe that since we are considering these values right after we invoke RECONSTRUCT(j), we have $\widehat{R}_i = R_i$. We first give a sketch of the proof. For each $i \geq j$, by Lemma 15, $\mathbf{m}_i \in M_i^*$ with probability at least $1 - \epsilon_{\text{sam}}/k^{10}$. By definition of M_i^*, $\mathbf{m}_i \in M_i^*$ means that in expectation, at least $\epsilon_{\text{sam}}/4$ fraction of the elements in $R_i^{b(i)}$ satisfy $P_{i+1}(b,e) \leq P_i(b,e) - 1$. As for the case $\mathbf{m}_i \notin M_i^*$, since this only happens with low probability, we can handle it using the bound $P_{i+1} \leq P_i$.

Formally, using the shorthand σ_i^{-m} instead of $\mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}$,

$$\begin{aligned}
&\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}\right] \\
&= \mathbb{P}\left[\mathbf{m}_i \in M_i^* | \sigma_i^{-m}\right] \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] + \mathbb{P}\left[\mathbf{m}_i \notin M_i^* | \sigma_i^{-m}\right] \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \notin M_i^*\right] \\
&\geq \mathbb{P}\left[\mathbf{m}_i \in M_i^* | \sigma_i^{-m}\right] \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right]
\end{aligned}$$

where the inequality follows from the fact that $P_i - P_{i+1}$ is always non-negative given Lemma 23. By Lemma 15, we can further bound this as

$$\begin{aligned}\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}\right] &\geq \mathbb{P}\left[\mathbf{m}_i \in M_i^* | \sigma_i^{-m}\right] \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] \\ &\geq (1 - \frac{\epsilon_{\text{sam}}}{100k^{10}}) \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] \\ &\overset{(a)}{\geq} \frac{1}{2} \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] \\ &= \frac{1}{2} \cdot \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}, \mathbf{m}_i \in M_i^*}\left[\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right] \end{aligned} \tag{22}$$

where for (a), we have used the fact that $\epsilon_{\text{sam}} < \frac{1}{4}$. Finally, we observe that

$$\begin{aligned} P_i - P_{i+1} = \sum_{e \in R_i} P(e,i) - \sum_{e \in R_{i+1}} P(e, i+1) &= \sum_{e \in R_i} P(e,i) - \sum_{e \in R_i} P(e,i+1) \\ &= \sum_{e \in R_i^{(b(i))}} (\, P(e,i) - P(e,i+1)\,) + \sum_{e \in R_i \setminus R_i^{(b(i))}} (\, P(e,i) - P(e,i+1)\,) \\ &\overset{(a)}{\geq} \sum_{e \in R_i^{(b(i))}} (\, P(e,i) - P(e,i+1)\,) \\ &\geq \sum_{e \in R_i^{(b(i))}} \mathbb{1}\left[P(e,i) - P(e,i+1) \geq 1\right] \\ &\overset{(b)}{=} \sum_{e \in R_i^{(b(i))}} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] \\ &\overset{(c)}{=} \left|R_i^{(b(i))}\right| - \left|\textsc{Filter}(R_i^{(b(i))}, G_i, \tau^{(i)})\right| \\ &= \left|R_i^{(b(i))}\right| - \left|\textsc{Filter}(R_i^{(b(i))}, G_{i-1} \cup S_i, \tau^{(i)})\right|. \end{aligned}$$

In the above derivation, the second equality follows from the fact that $P(e, i + 1) = 0$ for $e \notin R_{i+1}$, inequality (a) follows from Lemma 23, (b) follows from the fact that if $|G_i| \geq k$, then $P(e, i + 1) < P(e, i)$ will hold for $e \in R_i$ because $P(e, i) \geq 1$ and $P(e, i+1) = 0$, and if $|G_i| \neq k$, then $P(e, i) < P(e, i+1)$ is equivalent to $f(e|G_i) < \tau^{(i)}$ for $e \in R_i^{(b(i))}$. Finally, (c) follows from the definition of Filter.

Therefore,

$$\begin{aligned}\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}\right] &\geq \frac{1}{2} \cdot \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}, \mathbf{m}_i \in M_i^*}\left[\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right] \\ &\geq \frac{1}{2} \cdot \frac{\epsilon_{\text{sam}}}{4} |R_i^{b(i)}| \end{aligned}$$

where the first inequality follows from (22) and the final inequality follows from (8). □

We now prove the following auxiliary lemma

Lemma 25. *Let $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_n$ be a sequence of integer positive variables such that $\mathbf{X}_i \leq \mathbf{X}_{i-1}$ and*

$$\mathbb{E}\left[\mathbf{X}_i \mid \mathbf{X}_1 = X_1, \ldots, \mathbf{X}_{i-1} = X_{i-1}\right] \leq (1 - \varepsilon')X_{i-1}.$$

Let T denote the first index i such that $X_T = 0$ and assume that $\mathbf{X}_0 = N$ for some fixed integer N. Then $\mathbb{E}[\mathbf{T}] \leq \frac{\log(N)+1}{poly(\varepsilon')}$.

Proof. Let T_i denote the first index such that $X_{T_i} \leq (1 - \varepsilon/2)^i N$. We claim that

$$\mathbb{E}\left[\mathbf{T}_i - \mathbf{T}_{i-1}\right] \leq \text{poly}(\frac{1}{\varepsilon'}). \tag{23}$$

By Markov's inequality,

$$\mathbb{P}\left[\mathbf{X}_i > (1-\varepsilon'/2)X_{i-1} \mid \mathbf{X}_1 = X_1, \ldots, \mathbf{X}_{i-1} = X_{i-1}\right] \leq \frac{1-\varepsilon'}{1-\varepsilon'/2} \leq 1-\varepsilon'/2.$$

Therefore, because the mean of a geometric random variable with parameter p is $\frac{1}{p}$, it follows that in expectation, we need to increase the index i at most $O(\frac{1}{\varepsilon'})$ times before we decrease X_i by a factor of $1-\varepsilon'/2$. Since $X_i \leq X_{i-1}$, this implies (23). Therefore, letting T'_i denote the first index such that $X_{T'_i} \leq \frac{1}{2^i}N$, it follows that

$$\mathbb{E}\left[\mathbf{T}'_i - \mathbf{T}'_{i-1}\right] \leq \log_{\frac{1}{1-\varepsilon'/2}}(2) O(\frac{1}{\varepsilon'}) = \frac{1}{\text{poly}(\varepsilon')}$$

Note however that $T'_{\log(N)+1} = T$ because X_i are integers. Therefore,

$$\mathbb{E}\left[\mathbf{T}\right] \leq \mathbb{E}\left[\mathbf{T}'_{\log(N)+1}\right] = \sum_{i=1}^{\log(N)+1} \mathbb{E}\left[\mathbf{T}'_i - \mathbf{T}'_{i-1}\right] \leq \frac{\log(N)+1}{\text{poly}(\varepsilon')}.$$

□

Lemma 26. *Assume we are given sets R_j, G_{j-1} satisfying* FILTER$(R_j, G_{j-1}, \tau) = R_j$*, and we invoke* RECONSTRUCT(j) *with $(\overline{R}_j, D)$ set to $(R_j, \emptyset)$, obtaining the (random) values $\mathbf{T}, \mathbf{S}_j, \ldots \mathbf{S}_T, \mathbf{R}_{j+1}, \ldots \mathbf{R}_{T+1}$. The expected value of $\mathbf{T} - j$ is bounded by poly$(\log(|R_j|), \log(k), \frac{1}{\varepsilon})$.*

Proof. As before, we note that $\widehat{R}_i = R_i$ for any $i \geq j$ because we are considering the values right after invoking RECONSTRUCT(j).

Recall that by Lemma 22, we have

$$|R_i| \leq P_i \leq |R_i| \cdot \log_{1+\epsilon_{\text{buck}}}(4k)$$

Given Lemma 24, for $i \geq j$,

$$\begin{aligned}\mathbb{E}\left[P_i - \mathbf{P}_{i+1} | \sigma_i^{-m}\right] &\geq \frac{\epsilon_{\text{sam}}}{8}|R_i^{b(i)}| \overset{(a)}{\geq} \frac{\epsilon_{\text{sam}}}{8}\frac{|R_i|}{\lceil \log_{1+\epsilon_{\text{buck}}}(2k)\rceil} \geq \frac{\epsilon_{\text{sam}}}{8}\frac{P_i}{\log^2_{1+\epsilon_{\text{buck}}}(4k)} \\ &= \frac{\epsilon_{\text{sam}}}{8}\frac{P_i}{\log^2(4k)} \cdot \log^2(1+\epsilon_{\text{buck}}) \\ &\overset{(b)}{\geq} \frac{\epsilon_{\text{sam}}}{8}\frac{P_i}{\log^2(4k)} \cdot \frac{{\epsilon_{\text{buck}}}^2}{16} \\ &\geq \frac{\epsilon_{\text{sam}} \cdot {\epsilon_{\text{buck}}}^2}{160\log^2(4k)} \cdot P_i\end{aligned}$$

where for (a), we have used the fact that $R_i^{b(i)}$ was the largest bucket, and for (b), we have used the inequality $\log(1+x) \geq \frac{x}{4}$ for $x < 1$

Setting $\varepsilon' = \frac{\epsilon_{\text{sam}} \cdot {\epsilon_{\text{buck}}}^2}{160\log^2(4k)}$, it follows that

$$\mathbb{E}\left[\mathbf{P}_{i+1} | \sigma_i^{-m}\right] \leq (1-\varepsilon') \cdot P_i,$$

Since the value of $\mathbf{P}_1, \ldots, \mathbf{P}_i$ is deterministic conditioned on σ_i^{-m}, which further implies

$$\mathbb{E}\left[\mathbf{P}_{i+1} | \mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1\right] \leq (1-\varepsilon') \cdot P_i. \tag{24}$$

Formally, if $P_i \neq 0$, we have $\mathbf{T} \geq i$ given Lemma 22. Therefore, by iterated expectation,

$$\begin{aligned}\mathbb{E}\left[\mathbf{P}_{i+1} | \mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1\right] &= \mathbb{E}\left[\mathbf{P}_{i+1} | \mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1, \mathbf{T} \geq i\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{P}_{i+1} | \mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1, \mathbf{T} \geq i, \mathbf{H}_i = H_i\right]\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{P}_{i+1} | \mathbf{T} \geq i, \mathbf{H}_i = H_i\right]\right] \\ &\leq (1-\varepsilon')P_i\end{aligned}$$

where the third equality follows from the fact that $\mathbf{P}_1, \ldots, \mathbf{P}_i$ is deterministic conditioned on $\{\, \mathbf{T} \geq i, \mathbf{H}_i = H_i \,\}$. If $P_i = 0$, then (24) holds trivially because $P_{i+1} \leq P_i = 0$.

Observe that

$$\frac{1}{\varepsilon'} = \text{poly}(\log(k), \frac{1}{\varepsilon})$$

and that

$$P_j \leq \log_{1+\epsilon_{\text{buck}}}(4k)|R_j|.$$

The claim now follows from Lemma 25. □

Lemma 27. *The expected number of queries made by calling* RECONSTRUCT*(i) is* $|\widehat{R}_i| \cdot$ *poly*$(\log(|\widehat{R}_i|), \log(k), \frac{1}{\varepsilon})$*, where* $|\widehat{R}_i|$ *refers to the size of* $\widehat{R}_i$ *after the update.*

Proof. Before stating the proofs, we note that the set $\widehat{R}_i$ itself does not change during RECONSTRUCT(i). Therefore, the value $|\widehat{R}_i|$ is deterministic when conditioned on the value of the data structure before the update.

We now note the algorithm for reconstruction first sets R_i to the new value of $\widehat{R}_i$, and then starts building the levels. Therefore, by Lemma 15, when each iteration of the while-loop in algorithm 1 is executed, the call to CALCSAMPLECOUNT makes at most $O((|\widehat{R}_i|) \cdot \text{poly}(\log(k), \frac{1}{\epsilon}))$ queries. By Lemma 26, in expectation, the while-loop is executed at most $\text{poly}(\log(|\widehat{R}_i|), \log(k), \frac{1}{\varepsilon})$ times. Since FILTER and bucketing make at most $O(|\widehat{R}_i|)$ queries, the claim follows from Lemma 26. □

A.4.2. AMORTIZATION

In this section, we bound the expected amortized query complexity of the insert and delete operations. We begin by bounding the number of levels $\mathbf{T}$ at an arbitrary point in the update stream. We state this result in Lemma 28. Next, in Lemma 29, we show how to bound the expected amortized query complexity of the algorithm by charging back each call of RECONSTRUCT to the updates that triggered it.

Lemma 28. *At any point in the stream, the expected number of levels* $\mathbb{E}[\mathbf{T}]$ *is at most poly*$(\log(n), \log(k), \frac{1}{\varepsilon})$.

Proof. We note that this lemma is different from Lemma 26 because we are claiming that the number of levels is always bounded in expectation, while Lemma 26 can only bound $\mathbb{E}[\mathbf{T} - j]$ right after a call to RECONSTRUCT(j). In particular, this also means that we can no longer assume $\widehat{R}_i = R_i$. The proof follows using a similar technique as Lemma 26. We first claim that a variant of Lemma 24 still holds. More formally, we claim that

$$\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}\right] \geq \frac{\epsilon_{\text{sam}}}{16} \cdot |R_i^{(b(i))}| \tag{25}$$

for any $i \geq 1$. The proof follows with the exact same logic as the proof of Lemma 24. Using the shorthand σ_i^{-m} to denote $\mathbf{T} \geq i, \mathbf{H}_i^{-m} = H_i^{-m}$ (Definition 16), for all $i \geq 1$,

$$\begin{aligned}
\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}\right] &\geq \mathbb{P}\left[\mathbf{m}_i \in M_i^* \middle| \sigma_i^{-m}\right] \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] \\
&\overset{(a)}{\geq} (1 - \frac{\epsilon_{\text{sam}}}{100k^{10}}) \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] \\
&\geq \frac{1}{2} \cdot \mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i \in M_i^*\right] \\
&= \frac{1}{2} \cdot \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}, \mathbf{m}_i \in M_i^*}\left[\mathbb{E}\left[P_i - \mathbf{P}_{i+1} \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right]
\end{aligned} \tag{26}$$

where for (a) we have now used Lemma 17 instead of 15.

Next, we observe that

$$\begin{aligned}
P_i - P_{i+1} &= \sum_{e \in \widehat{R}_i} (P(e,i) - P(e,i+1)) \\
&\geq \sum_{e \in R_i^{(b(i))} \cap \widehat{R}_i} \mathbb{1}\left[P(e,i) - P(e,i+1) \geq 1\right] \\
&\geq \sum_{e \in R_i^{(b(i))} \cap \widehat{R}_i} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] \\
&= \sum_{e \in R_i^{(b(i))}} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] - \sum_{e \in R_i^{(b(i))} \setminus \widehat{R}_i} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] \\
&\overset{(a)}{\geq} \sum_{e \in R_i^{(b(i))}} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] - \sum_{e \in R_i^{(b(i))} \cap D} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] \\
&\geq \sum_{e \in R_i^{(b(i))}} \mathbb{1}\left[f(e|G_i) < \tau^{(i)} \text{ or } |G_i| \geq k\right] - |R_i^{(b(i))} \cap D| \\
&= \left|R_i^{(b(i))}\right| - \left|\textsc{Filter}(R_i^{(b(i))}, G_i, \tau^{(i)})\right| - |R_i^{(b(i))} \cap D| \\
&\overset{(b)}{\geq} \left|R_i^{(b(i))}\right| - \left|\textsc{Filter}(R_i^{(b(i))}, G_i, \tau^{(i)})\right| - \epsilon_{\text{del}} |R_i^{(b(i))}|
\end{aligned}$$

where for (a), we have used the fact that $R_i \setminus \widehat{R}_i \subseteq D$, and for (b), we have used Lemma 7. Therefore,

$$\begin{aligned}
\mathbb{E}\left[P_i - \mathbf{P}_{i+1} | \sigma_i^{-m}\right] + \frac{\epsilon_{\text{del}}}{2} |R_i^{(b(i))}| &\geq \frac{1}{2} \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}, \mathbf{m}_i \in M_i^*} \left[\mathbb{E}\left[P_i - \mathbf{P}_{i+1} | \sigma_i^{-m}, \mathbf{m}_i = m_i\right]\right] + \frac{\epsilon_{\text{del}}}{2} |R_i^{(b(i))}| \\
&\geq \frac{1}{2} \mathbb{E}_{m_i \sim \mathbf{m}_i | \sigma_i^{-m}, \mathbf{m}_i \in M_i^*} \left[\left|R_i^{(b(i))}\right| - \left|\textsc{Filter}(R_i^{(b(i))}, G_i, \tau^{(i)})\right| \middle| \sigma_i^{-m}, \mathbf{m}_i = m_i\right] \\
&\geq \frac{1}{2} \cdot \frac{\epsilon_{\text{sam}}}{4} |R_i^{b(i)}|
\end{aligned}$$

where the first inequality follows from (26), and the final inequality follows from Lemma 12, together with Equation (8).

Since $\epsilon_{\text{del}} \leq \epsilon_{\text{sam}}/16$, it follows that

$$\mathbb{E}\left[P_i - \mathbf{P}_{i+1} | \sigma_i^{-m}\right] \geq \frac{\epsilon_{\text{sam}}}{16} |R_i^{b(i)}|.$$

We can conclude from Lemma 7 that $|\widehat{R}_i| \leq |\overline{R}_i| \leq \frac{3}{2}|R_i| \leq 2|R_i|$. Therefore,

$$\begin{aligned}
\mathbb{E}\left[P_i - \mathbf{P}_{i+1} | \sigma_i^{-m}\right] \geq \frac{\epsilon_{\text{sam}}}{16} |R_i^{b(i)}| \geq \frac{\epsilon_{\text{sam}}}{16} \frac{|R_i|}{\log_{1+\epsilon_{\text{buck}}}(4k)} &\overset{(a)}{\geq} \frac{\epsilon_{\text{sam}}}{32} \frac{|\widehat{R}_i|}{\log_{1+\epsilon_{\text{buck}}}(4k)} \\
&\overset{(b)}{\geq} \frac{\epsilon_{\text{sam}}}{32} \frac{P_i}{\log^2_{1+\epsilon_{\text{buck}}}(4k)} \\
&= \frac{\epsilon_{\text{sam}}}{32} \frac{P_i}{\log^2(4k)} \cdot \log^2(1+\epsilon_{\text{buck}}) \\
&\overset{(c)}{\geq} \frac{\epsilon_{\text{sam}}}{32} \frac{P_i}{\log^2(4k)} \cdot \frac{\epsilon_{\text{buck}}{}^2}{20}
\end{aligned}$$

where for (a), we have used $|\widehat{R}_i| \leq 2|R_i|$, for (b) we have used Lemma 22, and for (c) we have used the inequality $\log(1+x) \geq \frac{x}{4}$ for $x < 1$ and the assumption $\epsilon_{\text{buck}} \leq 1/10$

Defining $\varepsilon' := \frac{\epsilon_{\text{sam}}}{32\log^2(4k)} \cdot \frac{\epsilon_{\text{buck}}{}^2}{20}$, it follows that

$$\mathbb{E}\left[\mathbf{P}_{i+1} | \sigma_i^{-m}\right] \leq (1-\varepsilon') \cdot P_i,$$

As before, this further implies

$$\mathbb{E}\left[\mathbf{P}_{i+1}|\mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1\right] \leq (1-\varepsilon') \cdot P_i.$$

Formally, if $P_i \neq 0$, we have $\mathbf{T} \geq i$ given Lemma 22. Therefore, by iterated expectation,

$$\begin{aligned}\mathbb{E}\left[\mathbf{P}_{i+1}|\mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1\right] &= \mathbb{E}\left[\mathbf{P}_{i+1}|\mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1, \mathbf{T} \geq i\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{P}_{i+1}|\mathbf{P}_i = P_i, \ldots, \mathbf{P}_1 = P_1, \mathbf{T} \geq i, \mathbf{H}_i = H_i\right]\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{P}_{i+1}|\mathbf{T} \geq i, \mathbf{H}_i = H_i\right]\right] \\ &\leq (1-\varepsilon')P_i\end{aligned}$$

as claimed. If $P_i = 0$, then (24) holds trivially because $P_{i+1} \leq P_i = 0$. Note however that

$$\frac{1}{\varepsilon'} = \text{poly}(\log(k), \frac{1}{\epsilon})$$

and

$$P_1 \leq \log_{1+\epsilon_{\text{buck}}}(4k)|\widehat{R}_1|.$$

The claim now follows from Lemma 25.

□

Lemma 29. *The amortized query complexity of our algorithm is at most poly*$(\log(n), \log(k), \frac{1}{\epsilon})$

Proof. For queries that were not caused by RECONSTRUCT, each insertion or deletion can cause at most $O(\mathbf{T})$ queries where $\mathbf{T}$ denotes the number of levels at the time of update, which is bounded by poly$(\log(n), \log(k), \frac{1}{\epsilon})$ given Lemma 28 and the fact that $|\widehat{R}_1| \leq n$.

We will therefore consider the number of queries made by RECONSTRUCT. The main idea is as follows. We charge the cost of each invocation of RECONSTRUCT(i) to the updates that caused it. In other words, we charge the cost to the insertion updates corresponding to $\overline{R}_i \backslash R_i$ if RECONSTRUCT(i) was triggered by an insertion and we charge the cost to the deletion updates corresponding to $R_i^{(b(i))} \cap D$ if RECONSTRUCT(i) was triggered by a deletion. Each time RECONSTRUCT(i) is invoked for some i, the expected number of queries made is $|\widehat{R}_i|$poly$(\log(|\widehat{R}_i|), \log(k), \frac{1}{\epsilon})$. However, this cost is spread across at least $\frac{|\widehat{R}_i|}{\text{poly}(\log(n),\log(k),\frac{1}{\varepsilon})}$ updates because of the reconstruction condition (see Claim 31). Therefore, each update is charged at most poly$(\log(n), \log(k), \frac{1}{\varepsilon})$ for each of the levels it affects. Since the expected number of levels at the time of the update is at most poly$(\log(n), \log(k), \frac{1}{\varepsilon})$ by Lemma 28, the claim follows. Note that here we are using different sources of randomness for bounding the expectation of $\mathbf{T}$ and bounding the number of queries for each reconstruction. The value of $\mathbf{T}$ is determined at the time of the update, while the number of queries is determined at a later time, i.e., when RECONSTRUCT is actually triggered. In our proof, we we first bound $\mathbf{T}$ by relying on the randomness of the algorithm before the update. We then condition on the randomness of the algorithm before the update, and bound the number of queries of each subsequent reconstruction by relying on the randomness of the algorithm after the update.

Formally, for an update u, a level i, and a time t,[6] let $Q_{u,i,t}$ denote the number of queries charged to u by an invocation of RECONSTRUCT(i) at time t. If RECONSTRUCT(i) was not invoked, or RECONSTRUCT(i) does not charge u, then set $Q_{u,i,t} = 0$. We need to show that for each u,

$$\mathbb{E}\left[\sum_{i,t} \mathbf{Q}_{u,i,t}\right] \leq \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}). \tag{27}$$

Let T^u denote the number of levels when u is inserted, we will show that

$$\mathbb{E}\left[\sum_{i,t} \mathbf{Q}_{u,i,t} \mid \mathbf{T}^u = T^u\right] \leq (T^u + 1)\text{poly}(\log(n), \log(k), \frac{1}{\epsilon}). \tag{28}$$

[6]Here "time" refers to an update in the stream. We use the word time to avoid confusion with u, and because the actual nature of the update at time t will not play an important role.

This would imply (27) given Lemma 28. To prove this, we first observe that

$$\sum_{i,t} \mathbb{1}\left[Q_{u,i,t} > 0\right] \le T^u + 1.$$

This is because for any fixed i, u there is at most one invocation of RECONSTRUCT(i) that can charge to u. In particular, u can only be charged by the first invocation of RECONSTRUCT(i) after u, if there have not been any invocations of RECONSTRUCT(j) for $j < i$ between u and this invocation of RECONSTRUCT(i). Since u is never charged by RECONSTRUCT(i) for $i > T^u + 1$, the claim follows.

Note however that

$$\mathbb{E}\left[\mathbf{Q}_{u,i,t} \mid \mathbf{T}^u = T^u\right] = \mathbb{E}\left[\mathbf{Q}_{u,i,t} \mid \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T^u\right] \mathbb{P}\left[\mathbf{Q}_{u,i,t} > 0 \mid \mathbf{T}^u = T^u\right].$$

We claim that

$$\mathbb{E}\left[\mathbf{Q}_{u,i,t} \mid \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T^u\right] \le \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}). \tag{29}$$

If this is shown, then it would imply Equation (28) because

$$\begin{aligned}
&\sum_{i,t} \mathbb{E}\left[\mathbf{Q}_{u,i,t} \mid \mathbf{T}^u = T^u\right] \\
&= \sum_{i,t} \mathbb{E}\left[\mathbf{Q}_{u,i,t} \mid \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T^u\right] \mathbb{P}\left[\mathbf{Q}_{u,i,t} > 0 \mid \mathbf{T}^u = T^u\right] \\
&\le \left(\sum_{i,t} \mathbb{P}\left[\mathbf{Q}_{u,i,t} > 0 \mid \mathbf{T}^u = T^u\right] \right) \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}) \\
&= \left(\sum_{i,t} \mathbb{E}\left[\mathbb{1}\left[\mathbf{Q}_{u,i,t} > 0\right] \mid \mathbf{T}^u = T^u\right] \right) \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}) \\
&= \left(\mathbb{E}\left[\sum_{i,t} \mathbb{1}\left[\mathbf{Q}_{u,i,t} > 0\right] \middle| \mathbf{T}^u = T^u \right] \right) \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}) \\
&\le \left(\mathbb{E}\left[\mathbf{T}^u + 1 \mid \mathbf{T}^u = T^u\right] \right) \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}) \\
&= (T^u + 1)\text{poly}(\log(n), \log(k), \frac{1}{\epsilon})
\end{aligned}$$

We divide the proof of (29) into two parts. Let $\widehat{R}_{i,t}$ denote the value of $\widehat{R}_i$ right after time t.

Claim 30.

$$\mathbb{E}\left[\sum_{u'} \mathbf{Q}_{u',i,t} \middle| \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T, \widehat{\mathbf{R}}_{i,t} = \widehat{R}_{i,t} \right] \le |\widehat{R}_{i,t}| \cdot \textit{poly}(\log(n), \log(k), \frac{1}{\epsilon})$$

for all i, t *such that* $\mathbb{P}\left[\mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T\right] > 0$.

Proof. We assume that $t \ge u$ as otherwise $\mathbf{Q}_{u,i,t} = 0$. Let r_t denote the random bits the algorithm uses in time t. By the law of iterated expectation, it suffices to show that

$$\mathbb{E}\left[\sum_{u'} \mathbf{Q}_{u',i,t} \middle| \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T, \widehat{\mathbf{R}}_{i,t} = \widehat{R}_{i,t}, \mathbf{r}_1 = r_1, \dots, \mathbf{r}_{t-1} = r_{t-1} \right] \le |\widehat{R}_{i,t}| \cdot \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}).$$

for all values of $r_1, \dots, r_{t-1}$ such that

$$\mathbb{P}\left[\mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T, \widehat{\mathbf{R}}_{i,t} = \widehat{R}_{i,t}, \mathbf{r}_1 = r_1, \dots, \mathbf{r}_{t-1} = r_{t-1}\right] > 0.$$

We note however that $\mathbb{1}\left[\mathbf{Q}_{u,i,t} > 0\right], \mathbf{T}^u$ are both a function of $\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}$. Specifically, $\mathbf{T}^u$ is a function of $\mathbf{r}_1, \ldots, \mathbf{r}_{u-1}$ and $\mathbb{1}\left[\mathbf{Q}_{u,i,t} > 0\right]$ is a function of $\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}$ because $\mathbf{Q}_{u,i,t} > 0$ if we invoke RECONSTRUCT(i) at time t, which is determined by $\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}$, and if there have been no reconstructions of level $j \leq i$ between times u and t, which is again determined by $\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}$. Therefore, we can drop $\mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T$ from the conditioning. We note that $\widehat{\mathbf{R}}_{i,t}$ is also deterministic conditioned on $\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}$, and it can also be dropped. Therefore, it suffices to show that

$$\mathbb{E}\left[\sum_{u'} \mathbf{Q}_{u',i,t} \,\middle|\, \mathbf{r}_1 = r_1, \ldots, \mathbf{r}_{t-1} = r_{t-1}\right] \leq |\widehat{R}_{i,t}| \cdot \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}).$$

Note however that $\sum_{u'} Q_{u,i,t}$ is the number of queries made at time t for executing RECONSTRUCT(i) (and is zero if RECONSTRUCT(i) is not invoked). This is independent of $\mathbf{r}_1, \ldots, \mathbf{r}_{t-1}$ and depends only on $\mathbf{r}_t$. Therefore, Lemma 27 completes the proof. □

Claim 31. *If* $|\widehat{R}_{i,t}| > 0$,

$$Q_{u,i,t} \leq \frac{\sum_{u'} Q_{u',i,t}}{|\widehat{R}_{i,t}|} \textit{poly}(\log(n), \log(k), \frac{1}{\epsilon}).$$

Proof. If RECONSTRUCT(i) is not called at time t, then the claim holds trivially. Otherwise, we need to show that the cost of this reconstruction is spread across at least $\frac{|\widehat{R}_{i,t}|}{\text{poly}(\log(k), \frac{1}{\varepsilon})}$ updates. We will drop the subscript t throughout the proof for convenience. Let R_i^- denote the value of R_i before the invocation of RECONSTRUCT(i), and let $(R_i^{(b(i))})^-$ denote the largest bucket in R_i^-. If $|R_i^-| = 0$ then $|\widehat{R}_i| = 1$ and the claim holds trivially as $\sum_{u'} Q_{u',i,t} \geq Q_{u,i,t}$. We therefore assume that $|R_i^-| > 0$.

Note that R_i^- is the same as the value of R_i the previous time that level i was constructed (i.e., RECONSTRUCT(j) was invoked for some $j \leq i$). Given the condition for invoking RECONSTRUCT(i), there have been either $\geq \frac{3}{2}|R_i^-|$ insertions or $\geq \epsilon_{\text{del}}|(R_i^{(b(i))})^-|$ deletions since the last reconstruction. Note that $\epsilon_{\text{del}}|(R_i^{(b(i))})^-| \geq \frac{|R_i^-|}{\text{poly}(\log(k), \frac{1}{\varepsilon})}$ because $(R_i^{(b(i))})^-$ is the largest bucket in R_i^- and there are a total of $\left\lceil \log_{1+\epsilon_{\text{buck}}}(2k) \right\rceil \leq \text{poly}(\log(k), \frac{1}{\varepsilon})$ buckets. Therefore, both when RECONSTRUCT(i) is triggered by an insertion and when it is triggered by a deletion, RECONSTRUCT(i) spreads its cost across at least $\frac{|R_i^-|}{\text{poly}(\log(k), \frac{1}{\varepsilon})}$ updates. Therefore,

$$Q_{u,i,t} \leq \frac{\sum_{u'} Q_{u',i,t}}{|R_i^-|} \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}). \tag{30}$$

Given Lemma 7, right before the update that triggers reconstruction, we have $|\widehat{R}_i| \leq 2|R_i|$. Additionally, the update can increase $|\widehat{R}_i|$ by at most 1. Therefore, $|\widehat{R}_i| \leq 2|R_i^-| + 1$. Therefore, since we assumed $|R_i^-| > 0$, we have $|R_i^-| \geq \frac{|\widehat{R}_i|}{3}$, and the claim follows from Equation (30). □

Given the above claims, if $|\widehat{R}_{i,t}| > 0$,

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{Q}_{u,i,t} \mid \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T^u, \widehat{\mathbf{R}}_{i,t} = \widehat{R}_{i,t}\right] \\
&\leq \frac{\mathbb{E}\left[\sum_{u'} \mathbf{Q}_{u',i,t} \mid \mathbf{Q}_{u,i,t} > 0, \mathbf{T}^u = T^u, \widehat{\mathbf{R}}_{i,t} = \widehat{R}_{i,t}\right]}{|\widehat{R}_{i,t}|} \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}). \\
&\leq \text{poly}(\log(n), \log(k), \frac{1}{\epsilon}),
\end{aligned}$$

This also holds if $|\widehat{R}_{i,t}| = 0$ because in this case, no queries are charged to u. Iterated expectation now implies Equation (29), finishing the proof. □

The above lemma implies Theorem 3 as claimed.

A.5. Proof of Theorem 1

We use the parallel run algorithm outlined in Section 3.5. We will show that this algorithm has expected amortized query complexity of poly$(\log(n), \log(k), \frac{1}{\varepsilon})$ per update, and the expected submodular value of the output is at least $(\frac{1}{2} - O(\epsilon))f(G_{\text{opt}})$. Each time an element is inserted or deleted, the operation affects at most $O($poly$(\log(k), \frac{1}{\varepsilon})$ parallel runs. Given Theorem 3, the expected amortized query complexity of each of these algorithms per update is at most poly$(\log(n), \log(k), \frac{1}{\varepsilon})$. Therefore, the expected amortized query complexity of the algorithm per update is poly$(\log(n), \log(k), \frac{1}{\varepsilon})$ as claimed.

It remains to analyze the approximation guarantee. Consider one of the runs with the parameter OPT_p satisfying $f(G_{\text{opt}}) \leq OPT_p \leq (1+\epsilon)f(G_{\text{opt}})$. Consider a modified algorithm where instead of inserting elements with $f(e) \in [\epsilon OPT_p/2k, OPT_p]$ into run p, we had inserted all elements e such that $f(e) \leq OPT_p$. We claim that this would not have changed the final output of the run p. This is because all elements with $f(e) \leq OPT_p/2k$ are filtered out from R_1 and $\overline{R}_1$. Therefore, the values of R_1 and $\overline{R}_1$ (and therefore the values of $T, R_2, \ldots, R_t, \overline{R}_2, \ldots, \overline{R}_T, S_1, \ldots S_T$) would not change, and so the output would stay the same. For the modified algorithm however, Theorem 2 implies that $f(G_T \backslash D) \geq (\frac{1}{2} - O(\epsilon))f(G_{\text{opt}})$, finishing the proof.