An Exponential Speedup in Parallel Running Time for Submodular Maximization without Loss in Approximation*

Eric Balkanski
Harvard University
ericbalkanski@g.harvard.edu

Aviad Rubinstein
Harvard University
aviad@seas.harvard.edu

Yaron Singer
Harvard University
yaron@seas.harvard.edu

Abstract

In this paper we study the adaptivity of submodular maximization. Adaptivity quantifies the number of sequential rounds that an algorithm makes when function evaluations can be executed in parallel. Adaptivity is a fundamental concept that is heavily studied across a variety of areas in computer science, largely due to the need for parallelizing computation. For the canonical problem of maximizing a monotone submodular function under a cardinality constraint, it is well known that a simple greedy algorithm achieves a $1 - 1/e$ approximation [NWF78] and that this approximation is optimal for polynomial-time algorithms [NW78]. Somewhat surprisingly, despite extensive efforts on submodular optimization for large-scale datasets, until very recently there was no known algorithm that achieves a constant factor approximation for this problem whose adaptivity is sublinear in the size of the ground set n.

Recent work by [BS18] describes an algorithm that obtains an approximation arbitrarily close to $1/3$ in $\mathcal{O}(\log n)$ adaptive rounds and shows that no algorithm can obtain a constant factor approximation in $\tilde{o}(\log n)$ adaptive rounds. This approach achieves an exponential speedup in adaptivity (and parallel running time) at the expense of approximation quality.

In this paper we describe a novel approach that yields an algorithm whose approximation is arbitrarily close to the optimal $1 - 1/e$ guarantee in $\mathcal{O}(\log n)$ adaptive rounds. *This algorithm therefore achieves an exponential speedup in parallel running time for submodular maximization at the expense of an arbitrarily small loss in approximation quality.* This guarantee is optimal in both approximation and adaptivity, up to lower order terms.

*This research was supported by a Google PhD Fellowship, a Robert N. Noyce Family Faculty Fellowship, a Rabin Postdoctoral Fellowship, NSF grant CAREER CCF 1452961, NSF CCF 1816874, BSF grant 2014389, NSF USICCS proposal 1540428, a Google Research award, and a Facebook research award.

1 Introduction

In this paper we study the adaptivity of submodular maximization. For the canonical problem of maximizing a non-decreasing submodular function under a cardinality constraint it is well known that the celebrated greedy algorithm which iteratively adds elements whose marginal contribution is largest achieves a $1 - 1/e$ approximation [NWF78]. Furthermore, this approximation guarantee is optimal for any algorithm that uses polynomially-many value queries [NW78].

The optimal approximation guarantee of the greedy algorithm comes at a price of high *adaptivity*. Informally, the *adaptivity* of an algorithm is the number of sequential rounds it makes when polynomially-many function evaluations can be executed in parallel in each round. The concept of adaptivity is heavily studied in computer science and optimization as it provides a measure of efficiency of parallel computation (see Section 1.3 for related work on parallel computation). For cardinality constraint k and ground set of size n, the greedy algorithm is k-adaptive since it sequentially adds elements in k rounds. In each round it makes $\mathcal{O}(n)$ function evaluations to identify and include the element with maximal marginal contribution to the set of elements selected in previous rounds. In the worst case $k \in \Omega(n)$ and thus the greedy algorithm is $\Omega(n)$-adaptive and its parallel running time is $\Theta(n)$.

Since submodular optimization is regularly applied on very large datasets, adaptivity is crucial as algorithms with low adaptivity enable dramatic speedups in parallel computing time. Submodular optimization has been studied for well over forty years now, and in the past decade there has been extensive study of submodular maximization for large datasets [BDF$^+$12, KMVV15, MKSK13, BV14, PJG$^+$14, BMKK14, MBK$^+$15, MZ15, MKBK15, BENW15, MBK16, BENW16, EMZ17]. Somewhat surprisingly however, until very recently, there was no known constant-factor approximation algorithm for submodular maximization whose adaptivity is sublinear in n.

In recent work [BS18] introduce an algorithm for maximizing monotone submodular functions under a cardinality constraint that achieves a constant factor approximation arbitrarily close to $1/3$ in $\mathcal{O}(\log n)$ adaptive steps. Furthermore, [BS18] show that no algorithm can achieve a constant factor approximation with $\tilde{o}(\log n)$ rounds.

For constant factor approximations, [BS18] provide an exponential speedup in the parallel runtime for the canonical problem of maximizing submodular functions under a cardinality constraint. This exponential improvement in adaptivity comes at the expense of the approximation quality achievable by $\Omega(n)$-adaptive algorithms (e.g. greedy), and raises two fundamental questions:

- *Is there an algorithm whose adaptivity is sublinear in the size of the ground set that obtains an approximation arbitrarily close to the optimal* $1 - 1/e$ *approximation guarantee?*
- *Given that a constant factor approximation cannot be obtained in* $\tilde{o}(\log n)$ *rounds, what is the best approximation achievable in* $\mathcal{O}(\log n)$ *rounds?*

In this paper we address both questions as summarized by our main result:

Theorem. *For any constant* $\epsilon > 0$*, any non-decreasing submodular function* $f : 2^{[n]} \to \mathbb{R}$ *and* $k \in [n]$*, there is an algorithm that with probability* $1 - o(1)$ *obtains a* $1 - 1/e - \epsilon$ *approximation to* $\max_{S:|S|\leq k} f(S)$ *in* $\mathcal{O}(\log n)$ *adaptive rounds.*

The algorithm gives an exponential speedup in parallel running time for maximizing a submodular function. In particular, our result shows that exponential speedups in parallel computing are possible with arbitrarily small sacrifice in the quality of the approximation achievable in poly-time.

1.1 Technical overview

The main goal of this paper is to achieve the optimal $1 - 1/e$ guarantee in $\mathcal{O}(\log n)$ adaptive steps. The optimal $1 - 1/e$ approximation of the greedy algorithm stems from the guarantee that for any given set S there exists an element whose marginal contribution to S is at least a $1/k$ fraction of the remaining optimal value $\texttt{OPT} - f(S)$. A standard inductive argument then shows that iteratively adding the element whose marginal contribution is maximal results in the $1 - 1/e$ approximation guarantee. To obtain the $1 - 1/e$ guarantee in $r = \mathcal{O}(\log n)$ adaptive steps rather than k, we could mimic this idea if in each adaptive step we could add a *block* of k/r elements whose marginal contribution to the existing solution S is at least a $1/r$ fraction of $\texttt{OPT} - f(S)$.

The entire challenge is in finding such a block of k/r elements in $\mathcal{O}(1)$ adaptive steps. A priori, this is a formidable task when k/r is super-constant. In general, the maximal marginal contribution over all sets of size k/r is as low as $(\texttt{OPT} - f(S))/r$. Finding a block of size t of maximal marginal contribution in polynomial time is as hard as solving the general problem of submodular maximization under cardinality constraint t, which, in general, cannot be approximated within any factor better than $1 - 1/e$ using polynomially-many queries [NW78]. Furthermore, we know it is impossible to approximate within any constant approximation in $o(\log n/\log\log n)$ adaptive rounds [BS18].

Despite this seeming difficulty, we show one can exploit a fundamental property of submodular functions to identify a block of size k/r whose marginal contribution is arbitrarily close to $(\texttt{OPT} - f(S))/r$. In general, we show that for monotone submodular functions, while it is hard to find a set of size k whose value is an arbitrarily good approximation to $\texttt{OPT}$, it is actually possible to find a set of size k/r whose value is arbitrarily close to that of $\texttt{OPT}/r$ in polynomial time for $r = O(\log n)$, even when k/r is super-constant.

In Section 2 we describe an algorithm which progressively adds a subset of size k/r to the existing solution S whose marginal contribution is arbitrarily close to $(\texttt{OPT} - f(S))/r$. To do so, it uses $\mathcal{O}(\log n)$ rounds in each such progression and it is hence $\mathcal{O}(\log^2 n)$-adaptive. At a high level, in each iteration that it adds a block of size k/r, the algorithm carefully and aggressively filters elements in $\mathcal{O}(\log n)$ rounds by considering their marginal contribution to a random set drawn from a distribution that evolves throughout the filtering iterations.

In Section 3, we generalize the algorithm so that, on average, every step of adding a block of k/r elements is done in $\mathcal{O}(1)$ adaptive steps. The main idea is to consider epochs, which consist of sequences of iterations such that, in the worst case, an iteration might still consist of $\mathcal{O}(\log n)$ rounds, but the amortized number of rounds per iteration during an epoch is now constant.

1.2 Paper organization

We first discuss related work, followed by preliminary definitions and notation below. In Section 2 we describe and analyze Iterative-Filtering which obtains an approximation guarantee arbitrarily close to $1 - 1/e$ in $\mathcal{O}(\log^2 n)$ adaptive rounds. In Section 3 we describe and analyze Amortized-Filtering which obtains the same approximation guarantee in $\mathcal{O}(\log n)$ rounds.

1.3 Related work

Parallel computing and depth. Adaptivity is closely related to the concept of *depth* in the PRAM model. The *depth* of a PRAM algorithm is the number of parallel steps it takes on a shared memory machine with any number of processors. That is, it is the longest chain of dependencies of the algorithm, including operations which are not queries. There is a long line of study on the design of low-depth algorithms (e.g. [Ble96, BPT11, BRS89, RV98, BRM98, BST12]). As discussed in further detail in Appendix A.2.1, our positive results extend to the PRAM model and our main algorithm has $\tilde{\mathcal{O}}(\log^2 n \cdot d_f)$ depth, where d_f is the depth required to evaluate the function on a set. While the PRAM model assumes that the input is loaded in memory, we consider the value query model where the algorithm is given oracle access to a function of potentially exponential size.

Adaptivity. The concept of adaptivity is generally well-studied in computer science, largely due to the role it plays in parallel computing, such as in sorting and selection [Val75, Col88, BMW16], communication complexity [PS84, DGS84, NW91], multi-armed bandits [AAAK17], sparse recovery [HNC09, IPW11, HBCN09], and property testing [CG17, BGSMdW12, CST$^+$17]. Beyond being a fundamental concept, adaptivity is important for applications where sequentiality is the main runtime bottleneck. We discuss in detail several such applications of submodular optimization in Appendix B. Somewhat surprisingly, until very recently $\Omega(n)$ was the best known adaptivity required for a constant factor approximation to maximizing a monotone submodular maximization under a cardinality constraint. As discussed above, [BS18] give an algorithm that is $\mathcal{O}(\log n)$-adaptive and achieves an approximation arbitrarily close to $1/3$. They also show that no algorithm can achieve a constant factor approximation with $\tilde{o}(\log n)$ rounds. The approach and algorithms in this paper are different than [BS18] and we provide a detailed comparison in Appendix B.1.

Map-Reduce. There is a long line of work on distributed submodular optimization in the Map-Reduce model [KMVV15, MKSK13, MZ15, MKBK15, BENW15, BENW16, EMZ17]. Map-Reduce is designed to tackle issues related to massive data sets that are too large to either fit or be processed by a single machine. Instead of addressing distributed challenges, adaptivity addresses the issue of sequentiality, where query-evaluation time is the main runtime bottleneck and where these evaluations can be parallelized. The existing Map-Reduce algorithms for submodular optimization have adaptivity that is linear in n in the worst-case. This high adaptivity is caused by the algorithms run on each machine, which are variants of the greedy algorithm and thus have adaptivity at least linear in k. Additional discussion about the Map-Reduce model is provided in Appendix A.2.2.

1.4 Basic definitions and notation

Submodularity. For a given function $f : 2^N \to \mathbb{R}$, the *marginal contribution* of an element $X \subseteq N$ to a set $S \subseteq N$ denoted $f_S(X)$ is defined as $f(S \cup X) - f(S)$. A function $f : 2^N \to \mathbb{R}$ is *submodular* if for any $S \subseteq T \subseteq N$ and any $a \in N \setminus T$ we have that $f_S(a) \geq f_T(a)$.[1] A function is *monotone* or *non-decreasing* if $f(S) \leq f(T)$ for all $S \subseteq T$. A submodular function f is also *subadditive*, meaning $f(S \cup T) \leq f(S) + f(T)$ for all $S \subseteq T$. The size of the ground set is $n = |N|$ and k denotes the cardinality constraint of the given optimization problem $\max_{S:|S|\leq k} f(S)$.

Adaptivity. As standard, we assume access to a *value oracle* of the function s.t. for any $S \subseteq N$ the oracle returns $f(S)$ in $\mathcal{O}(1)$ time. Given a value oracle for f, an algorithm is r-adaptive if every query $f(S)$ for the value of a set S occurs at a round $i \in [r]$ s.t. S is independent of the values $f(S')$ of all other queries at round i, with at most $\text{poly}(n)$ queries at every round. In Appendix A.2.1 we discuss adaptivity and parallel computing.

[1]For readability we abuse notation and write a instead of $\{a\}$ when evaluating a singleton.

2 Iterative-Filtering: An $\mathcal{O}(\log^2 n)$-adaptive Algorithm

In this section, we present the ITERATIVE-FILTERING algorithm which obtains an approximation arbitrarily close to $1 - 1/e$ in $\mathcal{O}(\log^2 n)$ adaptive rounds. At a high level, the algorithm iteratively identifies large blocks of elements of high value and adds them to the solution. There are $\mathcal{O}(\log n)$ such iterations and each iteration requires $\mathcal{O}(\log n)$ adaptive rounds, which amounts to $\mathcal{O}(\log^2 n)$-adaptivity. The analysis in this section will later be used as we generalize this algorithm to one that obtains an approximation arbitrarily close to $1 - 1/e$ in $\mathcal{O}(\log n)$ adaptive rounds.

2.1 Description of the algorithm

The ITERATIVE-FILTERING algorithm consists of r iterations which each add k/r elements to the solution S. To find these elements the algorithm filters out elements from the ground set using the FILTER subroutine and then adds a set of size k/r sampled uniformly at random from the remaining elements. Let $\mathcal{U}(X, t)$ denote the uniform distribution over subsets of X of size t. Throughout the paper we always sample sets of size $t = k/r$ and therefore write $\mathcal{U}(X)$ instead of $\mathcal{U}(X, \frac{k}{r})$ to simplify notation. The ITERATIVE-FILTERING algorithm is described formally above as Algorithm 1.

Algorithm 1 ITERATIVE-FILTERING

Input: constraint k, bound on number of iterations r
  $S \leftarrow \emptyset$
  for r iterations **do**
    $X \leftarrow \text{FILTER}(N, S, r)$
    $S \leftarrow S \cup R$, where $R \sim \mathcal{U}(X)$
  return S

The FILTER subroutine iteratively discards elements until a random set $R \sim \mathcal{U}(X)$ has marginal contribution arbitrarily close to the desired $(\texttt{OPT} - f(S))/r$ value. In each iteration, the elements discarded from the set of surviving elements X are those whose marginal contribution to $R \sim \mathcal{U}(X)$ is low. Intuitively, FILTER terminates quickly since if a random set has low expected marginal contribution, then there are many elements whose marginal contribution to a random set is low and these elements are then discarded. The subroutine FILTER is formally described below.

Algorithm 2 FILTER(X, S, r)

Input: Remaining elements X, current solution S, bound on number of outer-iterations r
  while $\mathbb{E}_{R\sim\mathcal{U}(X)}[f_S(R)] < (1-\epsilon)(\texttt{OPT} - f(S))/r$ **do**
    $X \leftarrow X \setminus \left\{a : \mathbb{E}_{R\sim\mathcal{U}(X)}\left[f_{S\cup(R\setminus\{a\})}(a)\right] < (1+\epsilon/2)(1-\epsilon)(\texttt{OPT} - f(S))/k\right\}$
  return X

Both ITERATIVE-FILTERING and FILTER are idealized versions of the algorithms we implement. This is due to the fact that we do not know the value of the optimal solution OPT and we cannot compute expectations exactly. In practice, we can apply multiple guesses of OPT in parallel and estimate expectations by repeated sampling. For ease of presentation we analyze these idealized versions of the algorithms and defer the presentation and analysis of the full algorithm to Appendix E. In our analysis we assume that in ITERATIVE-FILTERING when $\mathbb{E}_{R\sim\mathcal{U}(X)}[f_S(R)] \geq t$ this implies that a random set $R \sim \mathcal{U}(X)$ respects $f_S(R) \geq t$.[2]

2.2 Analysis

The analysis of Iterative-Filtering relies on two properties of its Filter subroutine: (1) the marginal contribution of the set of elements not discarded in Filter after $\mathcal{O}(r)$ iterations is arbitrarily close to $(\texttt{OPT} - f(S))/r$ and (2) there are at most k/r remaining elements after $\mathcal{O}(r)$ rounds. We assume that $\epsilon > 0$ is a small constant in the analysis.

2.2.1 Bounding the value of elements that survive Filter

We first prove that the marginal contribution of elements returned by Filter to the existing solution S is arbitrarily close to $(\texttt{OPT} - f(S))/r$. We do so by arguing that the set returned by Filter includes a subset of the optimal solution O with such marginal contribution. Let ρ be the number of iterations of the while loop in Filter. For a given iteration $i \in [\rho]$ let R_i be a random set of size $\frac{r}{k}$ drawn uniformly at random from X_i, where X_i are the remaining elements at iteration i. Notice that by monotonicity and submodularity, $f_S(O) \geq \texttt{OPT} - f(S)$. We first show that we can consider the marginal contribution of O not only to S but $S \cup (\cup_{i=1}^{\rho} R_i)$, while suffering an arbitrarily small loss. Considering the marginal contribution over random sets R_i is important to show that some optimal elements of high value must survive all rounds.

Lemma 1. *Let $R_i \sim \mathcal{U}(X)$ be the random set at iteration i of* Filter(N, S, r). *For all $S \subseteq N$ and $r, \rho > 0$, if* Filter(N, S, r) *has not terminated after ρ iterations, then*

$$\mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S\cup\left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \geq \left(1 - \frac{\rho}{r}\right) \cdot (\mathit{OPT} - f(S)).$$

Proof. We exploit the fact that if Filter(N, S, r) has not terminated after ρ iterations, then by the algorithm, the random set $R_i \sim \mathcal{U}(X)$ at iteration i has expected value that is upper bounded as follows:

$$\mathop{\mathbb{E}}_{R_i}\left[f_S(R_i)\right] < \frac{1-\epsilon}{r}(\texttt{OPT} - f(S))$$

for all $i \leq \rho$. Next, by subadditivity, we have $\mathbb{E}_{R_1,\ldots,R_\rho}\left[f_S\left(\left(\cup_{i=1}^{\rho} R_i\right)\right)\right] \leq \sum_{i=1}^{\rho} \mathbb{E}_{R_i}\left[f_S(R_i)\right]$ and, by monotonicity, $\mathbb{E}_{R_1,\ldots,R_\rho}\left[f_S\left(O \cup \left(\cup_{i=1}^{\rho} R_i\right)\right)\right] \geq \texttt{OPT} - f(S)$. Combining the above inequalities, we conclude that

$$\begin{aligned}
\mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S\cup\left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] &= \mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_S\left(O \cup \left(\cup_{i=1}^{\rho} R_i\right)\right)\right] - \mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_S\left(\left(\cup_{i=1}^{\rho} R_i\right)\right)\right] \\
&\geq \texttt{OPT} - f(S) - \sum_{i=1}^{\rho} \mathop{\mathbb{E}}_{R_i}\left[f_S(R_i)\right] \\
&\geq \left(1 - \frac{\rho}{r}\right) \cdot (\texttt{OPT} - f(S)).
\end{aligned}$$

□

Next, we bound the value of elements that survive filtering rounds. To do so, we use Lemma 1 to show that there exists a a subset T of the optimal solution O that survives ρ rounds of filtering and that has marginal contribution to S arbitrarily close to $(\texttt{OPT} - f(S))/r$.

[2]Since we estimate $E_{R\sim\mathcal{U}(X)}[f_S(R)]$ by sampling in the full version of the algorithm, there is at least one sample with value at least the estimated value of $E_{R\sim\mathcal{U}(X)}[f_S(R)]$ that we can take.

Lemma 2. *For all $S \subseteq N$ and $\epsilon > 0$, if $r \geq 20\rho\epsilon^{-1}$, then the elements X_ρ that survive ρ iterations of* Filter(N, S, r) *satisfy*

$$f_S(X_\rho) \geq \frac{1}{r}(1-\epsilon)(\mathit{OPT} - f(S)).$$

Proof. At a high level, the proof first defines a subset T of the optimal solution O. Then, the remaining of the proof consists of two main parts. First, we show that elements in T survive ρ iterations of Filter(N, S, r). Then, we show that $f_S(T) \geq \frac{1}{r}(1-\epsilon)(\texttt{OPT} - f(S))$. We introduce some notation. Let $O = \{o_1, \ldots, o_k\}$ be the optimal elements in some arbitrary order and $O_\ell = \{o_1, \ldots, o_\ell\}$. We define the following marginal contribution Δ_ℓ of each optimal element o_ℓ:

$$\Delta_\ell := \mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S \cup O_{\ell-1} \cup \left(\cup_{i=1}^{\rho} R_i \setminus \{o_\ell\}\right)}(o_\ell)\right].$$

We define T to be the set of optimal elements o_ℓ such that $\Delta_\ell \geq (1-\epsilon/4)\Delta$ where

$$\Delta := \frac{1}{k}\left(1 - \frac{\rho}{r}\right) \cdot (\texttt{OPT} - f(S)).$$

We first argue that elements in T survive ρ iterations of Filter(N, S, r). For element $o_\ell \in T$, we have

$$\Delta_\ell \geq (1-\epsilon/4)\Delta \geq \frac{1}{k}(1-\epsilon/4)\left(1 - \frac{\rho}{r}\right) \cdot (\texttt{OPT} - f(S)) \geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon) \cdot (\texttt{OPT} - f(S))$$

where the last inequality is since since $r \geq 20\rho\epsilon^{-1}$. Thus, at iteration $i \leq \rho$, by submodularity,

$$\mathop{\mathbb{E}}_{R_i}\left[f_{S \cup (R_i \setminus \{o_\ell\})}(o_\ell)\right] \geq \mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S \cup O_{\ell-1} \cup \left(\cup_{i=1}^{\rho} R_i \setminus \{o_\ell\}\right)}(o_\ell)\right] = \Delta_\ell \geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon) \cdot (\texttt{OPT} - f(S))$$

and o_ℓ survives all iterations $i \leq \rho$, for all $o_\ell \in T$.

Next, we argue that $f_S(T) \geq \frac{1}{r}(1-\epsilon)(\texttt{OPT} - f(S))$. Note that

$$\sum_{\ell=1}^{k} \Delta_\ell \geq \mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S \cup \left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \geq \left(1 - \frac{\rho}{r}\right) \cdot (\texttt{OPT} - f(S)) = k\Delta.$$

where the second inequality is by Lemma 1. Next, observe that

$$\sum_{\ell=1}^{k} \Delta_\ell = \sum_{o_\ell \in T} \Delta_\ell + \sum_{j \in O \setminus T} \Delta_\ell \leq \sum_{o_\ell \in T} \Delta_\ell + k(1-\epsilon/4)\Delta.$$

By combining the two inequalities above, we get $\sum_{o_\ell \in T} \Delta_\ell \geq k\epsilon\Delta/4$. Thus, by submodularity,

$$f_S(T) \geq \sum_{o_\ell \in T} f_{S \cup O_{\ell-1}}(o_\ell) \geq \sum_{o_\ell \in T} \mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S \cup O_{\ell-1} \cup \left(\cup_{i=1}^{\rho} R_i \setminus \{o_\ell\}\right)}(o_\ell)\right] = \sum_{o_\ell \in T} \Delta_\ell \geq k\epsilon\Delta/4.$$

We conclude that

$$f_S(X_\rho) \geq f_S(T) \geq k\epsilon\Delta/4 = (\epsilon/4)\left(1 - \frac{\rho}{r}\right) \cdot (\texttt{OPT} - f(S)) \geq \frac{1}{r} \cdot (1-\epsilon) \cdot (\texttt{OPT} - f(S)).$$

where the first inequality is by monotonicity and since $T \subseteq X_\rho$ is a set of surviving elements. $\square$

2.3 The adaptivity of Filter

The second part of the analysis bounds the number of adaptive rounds of the Filter algorithm. A main lemma for this part, Lemma 3, shows that a constant fraction of elements are discarded at every round of filtering. Combined with the previous lemma that bounds the value of remaining elements, Lemma 4 then shows that Filter has at most $\log n$ rounds. The analysis that a constant fraction of elements are discarded at every round is similar as in [BS18] and we defer the proof to the appendix for completeness. Since this is an important lemma, we give a proof sketch nevertheless.

Lemma 3. *Let X_i and X_{i+1} be the surviving elements at the start and end of iteration i of* Filter(N, S, r). *For all $S \subseteq N$ and $r, i, \epsilon > 0$, if* Filter(N, S, r) *does not terminate at iteration i, then*

$$|X_{i+1}| < \frac{|X_i|}{1 + \epsilon/2}.$$

Proof Sketch (full proof in Appendix C). At a high level, since the surviving elements must have high value and a random set has low value, we can then use the thresholds to bound how many such surviving elements there can be while also having a random set of low value. To do so, we focus on the value of $f(R_i \cap X_{i+1})$ of the surviving elements X_{i+1} in a random set $R_i \sim \mathcal{D}_{X_i}$.

First, the proof uses submodularity and the threshold for elements in X_{i+1} to survive from X_i to show that $\mathbb{E}\left[f_S(R_i \cap X_{i+1})\right] \geq |X_{i+1}| \cdot \frac{1}{r|X_i|} \cdot (1 + \epsilon/2)(1 - \epsilon)(\texttt{OPT} - f(S))$. Using monotonicity and the bound on the value of a random set $\mathbb{E}\left[f_S(R_i)\right]$ for Filter to discard additional elements, the proof then shows that $\mathbb{E}\left[f_S(R_i \cap X_{i+1})\right] < \frac{1}{r}(1 - \epsilon)(\texttt{OPT} - f(S))$ and concludes that $|X_{i+1}| \leq |X_i|/(1 + \epsilon/2)$. □

Thus, by the previous lemma, there are at most k/r surviving elements after logarithmically many filtering rounds and by Lemma 2, these remaining elements must have high value. Thus, Filter terminates and we obtain the following main lemma for the number of rounds.

Lemma 4. *For all $S \subseteq N$, if $r \geq 40\epsilon^{-2} \log n$, then* Filter$(N, S, r)$ *terminates after at most $O(\log n)$ iterations.*

Proof. If Filter(N, S, r) has not yet terminated after $2\epsilon^{-1} \log n$ iterations, then, by Lemma 3, at most k/r elements survived these $\rho = 2\epsilon^{-1} \log n$ iterations. By Lemma 2, with $r \geq 20\rho\epsilon^{-1}$, the set X_ρ of elements that survive these $2\epsilon^{-1} \log n$ iterations is such that $f_S(X_\rho) \geq \frac{1}{r} \cdot (1 - \epsilon)(\texttt{OPT} - f(S))$. Since there are at most k/r surviving elements X, $R = X_\rho$ for $R \sim \mathcal{U}(X_\rho)$ and

$$f_S(R) = f_S(X_\rho) \geq \frac{1}{r} \cdot (1 - \epsilon)(\texttt{OPT} - f(S)),$$

and Filter(N, S, r) terminates at this iteration. □

Main result for Iterative-Filtering. We are now ready to prove the main result for Iterative-Filtering. By Lemma 4, at every iteration of Iterative-Filtering, in at most $O(\log n)$ iterations of Filter, the value of the solution S is increased by at least $(1 - \epsilon)(\texttt{OPT} - f(S))/r$ with k/r new elements. The analysis of the $1 - 1/e - \epsilon$ approximation then follows similarly as for the standard analysis of the greedy algorithm. Regarding the total number of rounds, we fix parameter $r = 40\epsilon^{-2} \log n$. there are at most r iterations of Iterative-Filtering, each of which with at most $O(\log n)$ iterations of Filter and the queries at every iteration of Filter are non-adaptive. We defer the proof to Appendix C.

Theorem 1. *For any constant $\epsilon > 0$,* Iterative-Filtering *is a $\mathcal{O}\left(\log^2 n\right)$-adaptive algorithm that obtains a $1 - 1/e - \epsilon$ approximation, with parameter $r = 40\epsilon^{-2} \log n$.*

3 Amortized-Filtering: An $\mathcal{O}(\log n)$-adaptive Algorithm

In this section, we build on the algorithm and analysis from the previous section to obtain the main result of this paper. We present Amortized-Filtering which accelerates Iterative-Filtering by using less filtering rounds while maintaining the same approximation guarantee. In particular, it obtains an approximation arbitrarily close to $1 - 1/e$ in logarithmically-many adaptive rounds.

3.1 Description of the algorithm

Amortized-Filtering iteratively adds a block of k/r elements obtained using the Filter subroutine to the existing solution S, exactly as Iterative-Filtering. The improvement in adaptivity comes from the use of *epochs*. An epoch is a sequence of iterations during which the value of the solution S increases by at most $\epsilon(\texttt{OPT} - f(S))/20$. During an epoch, the algorithm invokes Filter with the surviving elements from the previous iteration of Amortized-Filtering, rather than all elements in the ground set as in Iterative-Filtering. In a new epoch, Filter is then again invoked with the ground set. A formal description of an idealized version is included below.

Algorithm 3 Amortized-Filtering

Input: bound on number of iterations r
  $S \leftarrow \emptyset$
  for $\frac{20}{\epsilon}$ epochs **do**
    $X \leftarrow N, T \leftarrow \emptyset$
    while $f_S(T) < (\epsilon/20)(\texttt{OPT} - f(S))$ and $|S \cup T| < k$ **do**
      $X \leftarrow \text{Filter}(X, S \cup T, r)$
      $T \leftarrow T \cup R$, where $R \sim \mathcal{U}(X)$
    $S \leftarrow S \cup T$
  return S

3.2 Analysis of Amortized-Filtering

As in the previous section, we analyze the idealized version described above and defer the analysis of the full algorithm to the appendix. Our analysis for Amortized-Filtering relies on the properties of every *epoch*. In particular, we first show that during an epoch, the surviving elements X have marginal contribution at least $\epsilon(\texttt{OPT} - f(S))/20$ to $S \cup T$ (Section 3.2.1). Notice that the marginal contribution is with respect to the set $S \cup T$ and the value with respect only to S. We then show that for any epoch, the total number of iterations of Filter during that epoch is $\mathcal{O}(\log n)$ (Section 3.2.2). We emphasize that an iteration of Filter is different than an iteration of the while-loop of Amortized-Filtering, i.e., an epoch consists of multiple iterations of Amortized-Filtering, each of which consists of multiple iterations of Filter. Since there are at most $20\epsilon^{-1}$ epochs, the amortized number of iterations of Filter per iteration of Amortized-Filtering is now constant.

3.2.1 Bounding the value of elements that survive an epoch

For any given epoch, we first bound the marginal contribution of O to $S \cup T$ and the random sets $\{R_i\}_{i=1}^{\rho}$ when there are ρ iterations of filtering during the epoch. Similar to the previous section, we show that the marginal contribution of O to $S \cup T$ and the random sets is arbitrarily close to the desired $\texttt{OPT} - f(S)$ value. The analysis is similar to the analysis of Lemma 1, except for a subtle yet crucial difference. The analysis in this section needs to handle the fact that the solution $S \cup T$ changes during the epoch. To do so we rely on the fact that the increase in the value of $S \cup T$ during an epoch is bounded. Due to space considerations, we defer the proof to Appendix D.

Lemma 5. *For any epoch j and $\epsilon > 0$, let $R_i \sim \mathcal{U}(X)$ be the random set at iteration i of filtering during epoch j. For all $r, \rho > 0$, if epoch j has not ended after ρ iterations of filtering, then*

$$\mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+ \cup (\cup_{i=1}^{\rho} R_i)}(O)\right] \geq \left(1 - \frac{\rho}{r} - \epsilon/20\right) \cdot (\textit{OPT} - f(S_j))$$

where S_j is the set S at epoch j and S_j^+ is the set $S \cup T$ at the last iteration of epoch j.

Next, we bound the value of elements that survive the filtering iterations during an epoch. The proof is similar to that of Lemma 2, modified to handle the fact that the solution S evolves during an epoch. We defer the proof to Appendix D.

Lemma 6. *For any epoch j and $\epsilon > 0$, if $r \geq 20\rho\epsilon^{-1}$, then the elements X_ρ that survive ρ iterations of filtering at epoch j satisfy*

$$f_{S_j^+}(X_\rho) \geq (\epsilon/4)(1-\epsilon)(\textit{OPT} - f(S_j)).$$

where S_j is the set S at epoch j and S_j^+ is the set $S \cup T$ at the last iteration of epoch j.

3.2.2 The adaptivity of an epoch

The next lemma bounds the total number of iterations of filtering per epoch. At a high level, similarly as for Iterative-Filtering, a constant fraction of elements are discarded at each iteration of filtering by Lemma 3 and there are at most k/r surviving elements after logarithmically-many filtering rounds. Then, we use Lemma 6 and the fact that the surviving elements during an epoch have high contribution to show that the epoch terminates.

Lemma 7. *In any epoch of* Amortized-Filtering *and for any $\epsilon \in (0, 1/2)$, if $r \geq 40\epsilon^{-2}\log n$, then there are at most $2\epsilon^{-1}\log n$ iterations of filtering during the epoch.*

Proof. If an epoch j has not yet terminated after $\rho = 2\epsilon^{-1}\log n$ iterations of filtering, then, by Lemma 3, at most k/r elements survived these ρ filtering iterations. We consider the set T obtained after these ρ filtering iterations. By Lemma 6, with $r \geq 20\rho \cdot \epsilon^{-1}$, the set X_ρ of elements that survive these iterations is such that $f_{S \cup T}(X_\rho) \geq (\epsilon/4) \cdot (1-\epsilon)(\texttt{OPT} - f(S))$. Since there are at most k/r surviving elements, $R = X_\rho$ for $R \sim \mathcal{U}(X_\rho)$ and

$$\mathbb{E}[f_{S \cup T}(R)] \geq f_{S \cup T}(X_\rho) \geq (\epsilon/4) \cdot (1-\epsilon)(\texttt{OPT} - f(S)) \geq \frac{1}{r} \cdot (1-\epsilon)(\texttt{OPT} - f(S \cup T))$$

where the last inequality is by monotonicity. Thus, the current call to the Filter subroutine terminates and X_ρ is added to T by the algorithm. Next,

$$f_S(T \cup X_\rho) \geq f_S(X_\rho) \geq f_{S \cup T}(X_\rho) \geq (\epsilon/4) \cdot (1-\epsilon)(\texttt{OPT} - f(S)) \geq (\epsilon/20)(\texttt{OPT} - f(S))$$

the first inequality is by monotonicity and the second by submodularity. Thus, epoch j ends. $\square$

3.3 Main result

We are now ready to prove the main result of the paper which is that analysis of Amortized-Filtering. There are two cases: either the algorithm terminates after r iterations with $|S \cup T| = k$ or it terminates after $20\epsilon^{-1}$ epochs. With $r = \mathcal{O}(\log n)$, there are at most $\mathcal{O}(\log n)$ iterations of adding elements and at most $\mathcal{O}(1)$ epochs with $\mathcal{O}(\log n)$ filtering iterations per epoch. Thus the total number of adaptive rounds is $\mathcal{O}(\log n)$. The proof is deferred to Appendix D

Theorem 2. *For any constant $\epsilon > 0$, when using parameter $r = 40\epsilon^{-2}\log n$,* Amortized-Filtering *obtains a $1 - 1/e - \epsilon$ approximation in $\mathcal{O}(\log n)$-adaptive steps.*

Similarly as for Iterative-Filtering, Amortized-Filtering is an idealized version of the full algorithm since we do not know `OPT` and cannot compute expectations exactly. The full algorithm, which guesses `OPT` and estimates expectations arbitrarily well by sampling in one adaptive round, is formally described and analyzed in Appendix E. The algorithm is randomized due to the sampling at every round and its analysis is nearly identical to that presented in this section while accounting for an additional arbitrarily small errors due to the guessing of `OPT` and the estimates of the expectation. The main result is the following theorem for the full algorithm.

Theorem 3. *For any $\epsilon \in (0, 1/2)$, there exists an algorithm that obtains a $1 - 1/e - \epsilon$ approximation with probability $1 - \delta$ in $\mathcal{O}(\epsilon^{-2}\log n)$ adaptive steps. Its query complexity in each round is $\mathcal{O}\left(nk^2\log^3(n)\epsilon^{-5}\log\left(\frac{n}{\delta}\right)\right)$.*

References

[AAAK17] Arpit Agarwal, Shivani Agarwal, Sepehr Assadi, and Sanjeev Khanna. Learning with limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from pairwise comparisons. In *COLT*, pages 39–75, 2017.

[AM10] Saeed Alaei and Azarakhsh Malekian. Maximizing sequence-submodular functions and its application to online advertising. *arXiv preprint arXiv:1009.4153*, 2010.

[ANRW15] Noga Alon, Noam Nisan, Ran Raz, and Omri Weinstein. Welfare maximization with limited interaction. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1499–1512. IEEE, 2015.

[AWZ08] Akram Aldroubi, Haichao Wang, and Kourosh Zarringhalam. Sequential adaptive compressed sampling via huffman codes. *arXiv preprint arXiv:0810.4916*, 2008.

[BDF+12] Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In *SODA*, pages 1025–1035. Society for Industrial and Applied Mathematics, 2012.

[BENW15] Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *ICML*, pages 1236–1244, 2015.

[BENW16] Rafael da Ponte Barbosa, Alina Ene, Huy L Nguyen, and Justin Ward. A new framework for distributed submodular maximization. In *FOCS*, pages 645–654. Ieee, 2016.

[BGSMdW12] Harry Buhrman, David García-Soriano, Arie Matsliah, and Ronald de Wolf. The non-adaptive query complexity of testing k-parities. *arXiv preprint arXiv:1209.3849*, 2012.

[Ble96] Guy E Blelloch. Programming parallel algorithms. *Communications of the ACM*, 39(3):85–97, 1996.

[BMKK14] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 671–680, 2014.

[BMW16] Mark Braverman, Jieming Mao, and S Matthew Weinberg. Parallel algorithms for select and partition with noisy comparisons. In *STOC*, pages 851–862, 2016.

[BPR+16] Ashwinkumar Badanidiyuru, Christos Papadimitriou, Aviad Rubinstein, Lior Seeman, and Yaron Singer. Locally adaptive optimization: Adaptive seeding for monotone submodular functions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 414–429. Society for Industrial and Applied Mathematics, 2016.

[BPT11] Guy E Blelloch, Richard Peng, and Kanat Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *SPAA*, pages 23–32, 2011.

[BRM98] Guy E Blelloch and Margaret Reid-Miller. Fast set operations using treaps. In *SPAA*, pages 16–26, 1998.

[BRS89] Bonnie Berger, John Rompel, and Peter W Shor. Efficient nc algorithms for set cover with applications to learning and geometry. In *FOCS*, pages 54–59. IEEE, 1989.

[BS18] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *STOC*, 2018.

[BST12] Guy E Blelloch, Harsha Vardhan Simhadri, and Kanat Tangwongsan. Parallel and i/o efficient set covering algorithms. In *SPAA*, pages 82–90. ACM, 2012.

[BV14] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1497–1514, 2014.

[CG17] Clement Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *arXiv preprint arXiv:1702.05678*, 2017.

[Col88] Richard Cole. Parallel merge sort. *SIAM Journal on Computing*, 17(4):770–785, 1988.

[CST$^+$17] Xi Chen, Rocco A Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the query complexity of non-adaptive junta testing. *arXiv preprint arXiv:1704.06314*, 2017.

[CWW10] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

[CWY09] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DGS84] Pavol Duris, Zvi Galil, and Georg Schnitger. Lower bounds on communication complexity. In *STOC*, pages 81–91, 1984.

[DHK$^+$16] Nikhil R Devanur, Zhiyi Huang, Nitish Korula, Vahab S Mirrokni, and Qiqi Yan. Whole-page optimization and submodular welfare maximization with online bidders. *ACM Transactions on Economics and Computation*, 4(3):14, 2016.

[DNO14] Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 233–242. ACM, 2014.

[DR01] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

[EMZ17] Alessandro Epasto, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Bicriteria distributed submodular maximization in a few rounds. In *SPAA*, pages 25–33, 2017.

[FJK10] Joseph D Frazier, Peter K Jimack, and Robert M Kirby. On the use of adjoint-based sensitivity estimates to control local mesh refinement. *Commun Comput Phys*, 7:631–8, 2010.

[GK10] Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *COLT*, pages 333–345, 2010.

[GLL11] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48. ACM, 2011.

[HBCN09] Jarvis D Haupt, Richard G Baraniuk, Rui M Castro, and Robert D Nowak. Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements. In *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, pages 1551–1555. IEEE, 2009.

[HNC09] Jarvis Haupt, Robert Nowak, and Rui Castro. Adaptive sensing for sparse signal recovery. In *Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*, pages 702–707. IEEE, 2009.

[HS15] Thibaut Horel and Yaron Singer. Scalable methods for adaptively seeding a social network. In *Proceedings of the 24th International Conference on World Wide Web*, pages 441–451. International World Wide Web Conferences Steering Committee, 2015.

[IPW11] Piotr Indyk, Eric Price, and David P Woodruff. On the power of adaptivity in sparse recovery. In *FOCS*, pages 285–294. IEEE, 2011.

[JXC08] Shihao Ji, Ya Xue, and Lawrence Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, 2008.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[KMVV15] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing*, 2(3):14, 2015.

[KSV10] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *SODA*, pages 938–948, 2010.

[MBK+15] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1812–1818, 2015.

[MBK16] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, pages 1358–1367, 2016.

[MKBK15] Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In *NIPS*, pages 2881–2889, 2015.

[MKSK13] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*, pages 2049–2057, 2013.

[MNSW95] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. In *STOC*, pages 103–111. ACM, 1995.

[MSW08] Dmitry M Malioutov, Sujay Sanghavi, and Alan S Willsky. Compressed sensing with sequential observations. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3357–3360. IEEE, 2008.

[MZ15] Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *STOC*, pages 153–162, 2015.

[NW78] George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

[NW91] Noam Nisan and Avi Widgerson. Rounds in communication complexity revisited. In *STOC*, pages 419–429, 1991.

[NWF78] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[PJG+14] Xinghao Pan, Stefanie Jegelka, Joseph E Gonzalez, Joseph K Bradley, and Michael I Jordan. Parallel double greedy submodular maximization. In *Advances in Neural Information Processing Systems*, pages 118–126, 2014.

[PS84] Christos H Papadimitriou and Michael Sipser. Communication complexity. *Journal of Computer and System Sciences*, 28(2):260–269, 1984.

[RD02] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[RS06] Sofya Raskhodnikova and Adam Smith. A note on adaptivity in testing properties of bounded degree graphs. *ECCC, TR06-089*, 2006.

[RV98] Sridhar Rajagopalan and Vijay V Vazirani. Primal-dual rnc approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):525–540, 1998.

[SS13] Lior Seeman and Yaron Singer. Adaptive seeding in social networks. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 459–468. IEEE, 2013.

[STK16] Adish Singla, Sebastian Tschiatschek, and Andreas Krause. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. In *AAAI*, pages 2037–2043, 2016.

[STW15] Rocco A Servedio, Li-Yang Tan, and John Wright. Adaptivity helps for testing juntas. In *Proceedings of the 30th Conference on Computational Complexity*, pages 264–279. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[TIWB14] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421, 2014.

[Val75] Leslie G Valiant. Parallelism in comparison problems. *SIAM Journal on Computing*, 4(3):348–355, 1975.

Appendix

A Additional Discussion of Related Work

A.1 Adaptivity

Adaptivity has been heavily studied across a wide spectrum of areas in computer science. These areas include classical problems in theoretical computer science such as sorting and selection (e.g. [Val75, Col88, BMW16]), where adaptivity is known under the term of parallel algorithms, and communication complexity (e.g. [PS84, DGS84, NW91, MNSW95, DNO14, ANRW15]), where the number of rounds measures how much interaction is needed for a communication protocol.

For the multi-armed bandits problem, the relationship of interest is between adaptivity and query complexity, instead of adaptivity and approximation guarantee. Recent work showed that $\Theta(\log^\star n)$ adaptive rounds are necessary and sufficient to obtain the optimal worst case query complexity [AAAK17]. In the bandits setting, adaptivity is necessary to obtain non-trivial query complexity due to the noisy outcomes of the queries. In contrast, queries in submodular optimization are deterministic and adaptivity is necessary to obtain a non trivial approximation since there are at most polynomially many queries per round and the function is of exponential size. Adaptivity is also well-studied for the problems of sparse recovery (e.g. [HNC09, IPW11, HBCN09, JXC08, MSW08, AWZ08]) and property testing (e.g. [CG17, BGSMdW12, CST$^+$17, RS06, STW15]). In these areas, it has been shown that adaptivity allows significant improvements compared to the non-adaptive setting, which is similar to the results shown in this paper for submodular optimization. However, in contrast to all these areas, adaptivity has not been previously studied in the context of submodular optimization.

We note that the term adaptive submodular maximization has been previously used, but in an unrelated setting where the goal is to compute a policy which iteratively picks elements one by one, which, when picked, reveal stochastic feedback about the environment [GK10].

A.2 Related models of parallelism

A.2.1 Parallel computing and depth

Our main result extends to the PRAM model. Let d_f be the depth required to evaluate the function on a set, then there is a $\tilde{\mathcal{O}}(\log^2 n \cdot d_f)$ depth algorithm with $\tilde{\mathcal{O}}(nk^2)$ work whose approximation is arbitrarily close to $1 - 1/e$ for submodular maximization under a cardinality constraint.

The PRAM model is a generalization of the RAM model with parallelization, it is an idealized model of a shared memory machine with any number of processors which can execute instructions in parallel. The *depth* of a PRAM algorithm is the longest chain of dependencies of the algorithm, including operations which are not necessarily queries. Thus, in addition to the number of adaptive rounds of querying, depth also measures the number of adaptive steps of the algorithms which are not queries. The additional factor in the depth compared to the number of adaptive rounds is $d_f \cdot \tilde{O}(\log n)$, where d_f is the depth required to evaluate the function on a set in the PRAM model. The operations that our algorithms performed at every round, which are maximum, summation, set union, and set difference over an input of size at most quasilinear, can all be executed by algorithms with logarithmic depth. A simple divide-and-conquer approach suffices for maximum and summation, while logarithmic depth for set union and set difference can be achieved with treaps [BRM98].

A.2.2 Map-Reduce

The problem of distributed submodular optimization has been extensively studied in the Map-Reduce model in the past decade. This framework is primarily motivated by large scale problems over massive data sets. At a high level, in the Map-Reduce framework [DG08], an algorithm proceeds in multiple Map-Reduce rounds, where each round consists of a first step where the input to the algorithm is partitioned to be independently processed on different machines and of a second step where the outputs of this processing are merged. Notice that the notion of rounds in Map-Reduce is different than for adaptivity, where one round of Map-Reduce usually consists of multiple adaptive rounds. The formal model of [KSV10] for Map-Reduce requires the number of machines and their memory to be sublinear.

This framework for distributing the input to multiple machines with sublinear memory is designed to tackle issues related to massive data sets. Such data sets are too large to either fit or be processed by a single machine and the Map-Reduce framework formally models this need to distribute such inputs to multiple machines.

Instead of addressing distributed challenges, adaptivity addresses the issue of sequentiality, where each query evaluation requires a long time to complete and where these evaluations can be parallelized (see Section B for applications). In other words, while Map-Reduce addresses the *horizontal* challenge of large scale problems, adaptivity addresses an orthogonal *vertical* challenge where long query-evaluation time is causing the main runtime bottleneck.

A long line of work has studied problems related to submodular maximization in Map-Reduce achieving different improvements on parameters such as the number of Map-Reduce rounds, the communication complexity, the approximation ratio, the family of functions, and the family of constraints (e.g. [KMVV15, MKSK13, MZ15, MKBK15, BENW15, BENW16, EMZ17]). To the best of our knowledge, all the existing Map-Reduce algorithms for submodular optimization have adaptivity that is linear in n in the worst-case, which is exponentially larger than the adaptivity of our algorithm. This high adaptivity is caused by the distributed algorithms which are run on each machine. These algorithms are variants of the greedy algorithm and thus have adaptivity at least linear in k. We also note that our algorithm does not (at least trivially) carry over to the Map-Reduce setting.

B Applications

We discuss in detail several applications of submodular optimization where sequentiality is the main runtime bottleneck. In **crowdsourcing and data summarization**, algorithms involve subtasks performed by the crowd. The intervention of humans in the evaluation of queries causes algorithms with a large number of adaptive rounds to be impractical. A crowdsourcing platform consists of posted tasks and crowdworkers who are remunerated for performing these posted tasks. For several submodular optimization problems, such as data summarization, the value of queries can be evaluated on a crowdsourcing platform [TIWB14, STK16, BMW16]. The algorithm must wait to obtain the feedback from the crowdworkers, however an algorithm can ask different crowdworkers to evaluate a large number of queries in parallel.

In **experimental design**, the goal is to pick a collection of entities (e.g. subjects, chemical elements, data points) which obtains the best outcome when combined for an experiment. Experiments can be run in parallel and have a waiting time to observe the outcome [FJK10]. The submodular problem of **influence maximization**, initiated studied by [DR01, RD02, KKT03] has since then been well-studied (e.g. [CWY09, CWW10, GLL11, SS13, HS15, BPR+16]). Influence maximization consists of finding the most influential nodes in a social network to maximize the spread of information in this network. Information does not spread instantly and an algorithm must wait to observe the total number of nodes influenced by some seed set of nodes. In **advertising**, the goal is to select the optimal subset of advertisement slots to objectives such as the click-through-rate or the number of products purchased by customers, which are objectives exhibiting diminishing returns [AM10, DHK+16]. Naturally, a waiting time is incurred to observe the performance of different collections of advertisements.

B.1 Previous work on adaptivity for submodular maximization

The main algorithm in [BS18], Adaptive-Sampling, obtains a constant factor approximation in $O(\log n)$ adaptive rounds. It consists of two primitives, Down-Sampling and Up-Sampling. Down-Sampling is $O(\log n / \log \log n)$-adaptive but only obtains a $O(\log n)$ approximation. On the other hand, Up-Sampling obtains a constant factor approximation but in linearly many rounds. The main algorithm appropriately combines both primitives to obtain a constant factor approximation guarantee in $O(\log n)$ rounds.

The main algorithm in this paper, Amortized-Filtering, mimics the greedy analysis to obtain an approximation arbitrarily close to $1 - 1/e$ by finding a block of size k/r whose marginal contribution is arbitrarily close to $(\texttt{OPT} - f(S))/r$. We first give Iterative-Filtering which finds such a set in $O(\log n)$ rounds by filtering elements at every iteration. We build on that algorithm to obtain Amortized-Filtering, which obtains a $1 - 1/e$ approximation and uses a concept of epoch to obtain an amortized number of rounds that is constant per iteration during an epoch. The analysis for the approximation is thus very different to obtain the $1 - 1/e$ approximation. One similarity is Lemma 3 which shows that a constant fraction of elements can be discarded in one round, similarly as Lemma 1 in [BS18].

C Missing Analysis from Section 2

Lemma 3. *Let X_i and X_{i+1} be the surviving elements at the start and end of iteration i of* Filter(N, S, r). *For all $S \subseteq N$ and $r, i, \epsilon > 0$, if* Filter(N, S, r) *does not terminate at iteration i, then*

$$|X_{i+1}| < \frac{|X_i|}{1 + \epsilon/2}.$$

Proof. At a high level, since the surviving elements must have high value and a random set has low value, we can then use the thresholds to bound how many such surviving elements there can be while also having a random set of low value. To do so, we focus on the value of $f(R_i \cap X_{i+1})$ of the surviving elements X_{i+1} in a random set $R_i \sim \mathcal{D}_{X_i}$.

$$
\begin{aligned}
\mathbb{E}\left[f_S(R_i \cap X_{i+1})\right] &\geq \mathbb{E}\left[\sum_{a \in R_i \cap X_{i+1}} f_{S \cup (R_i \cap X_{i+1} \setminus a)}(a)\right] && \text{submodularity} \\
&\geq \mathbb{E}\left[\sum_{a \in X_{i+1}} \mathbb{1}_{a \in R_i} \cdot f_{S \cup (R_i \setminus a)}(a)\right] && \text{submodularity} \\
&= \sum_{a \in X_{i+1}} \mathbb{E}\left[\mathbb{1}_{a \in R_i} \cdot f_{S \cup (R_i \setminus a)}(a)\right]. \\
&= \sum_{a \in X_{i+1}} \Pr\left[a \in R_i\right] \cdot \mathbb{E}\left[f_{S \cup (R_i \setminus a)}(a) | a \in R_i\right] \\
&\geq \sum_{a \in X_{i+1}} \Pr\left[a \in R_i\right] \cdot \mathbb{E}\left[f_{S \cup (R_i \setminus a)}(a)\right] && \text{submodularity} \\
&\geq \sum_{a \in X_{i+1}} \Pr\left[a \in R_i\right] \cdot \frac{1}{k}\left(1 + \epsilon/2\right)\left(1 - \epsilon\right)\left(\texttt{OPT} - f(S)\right) && \text{algorithm} \\
&= |X_{i+1}| \cdot \frac{k}{r|X_i|} \cdot \frac{1}{k}\left(1 + \epsilon/2\right)\left(1 - \epsilon\right)\left(\texttt{OPT} - f(S)\right) && \text{definition of } \mathcal{U}(X) \\
&= |X_{i+1}| \cdot \frac{1}{r|X_i|} \cdot \left(1 + \epsilon/2\right)\left(1 - \epsilon\right)\left(\texttt{OPT} - f(S)\right).
\end{aligned}
$$

Next, since elements are discarded, a random set must have low value by the algorithm,

$$
\frac{1}{r}\left(1 - \epsilon\right)\left(\texttt{OPT} - f(S)\right) > \mathbb{E}\left[f_S(R_i)\right].
$$

By monotonicity, we get $\mathbb{E}\left[f_S(R_i)\right] \geq \mathbb{E}\left[f_S(R_i \cap X_{i+1})\right]$. Finally, by combining the above inequalities, we conclude that $|X_{i+1}| \leq |X_i|/(1 + \epsilon/2)$. □

Theorem 1. *For any constant $\epsilon > 0$,* Iterative-Filtering *is a $\mathcal{O}\left(\log^2 n\right)$-adaptive algorithm that obtains a $1 - 1/e - \epsilon$ approximation, with parameter $r = 40\epsilon^{-2} \log n$.*

Proof. Let S_i denote the solution S at the ith iteration of Iterative-Filtering. The algorithm increases the value of the solution S by at least $\left(1 - \epsilon\right)\left(\texttt{OPT} - f(S)\right)/r$ at every iteration with k/r new elements. Thus,

$$
f(S_i) \geq f(S_{i-1}) + \frac{1 - \epsilon}{r}\left(\texttt{OPT} - f(S_{i-1})\right).
$$

Next, we show by induction on i that

$$
f(S_i) \geq \left(1 - \left(1 - \frac{1 - \epsilon}{r}\right)^i\right) \texttt{OPT}.
$$

Observe that

$$\begin{aligned}
f(S_i) &\geq f(S_{i-1}) + \frac{1-\epsilon}{r}\left(\texttt{OPT} - f(S_{i-1})\right) \\
&= \frac{1-\epsilon}{r}\texttt{OPT} + \left(1 - \frac{1-\epsilon}{r}\right) f(S_{i-1}) \\
&\geq \frac{1-\epsilon}{r}\texttt{OPT} + \left(1 - \frac{1-\epsilon}{r}\right)\left(1 - \left(1 - \frac{1-\epsilon}{r}\right)^{i-1}\right)\texttt{OPT} \\
&= \left(1 - \left(1 - \frac{1-\epsilon}{r}\right)^{i}\right)\texttt{OPT}
\end{aligned}$$

Thus, with $i = r$ where there has been r iterations of adding k/r elements, we return solution S such that

$$f(S) \geq \left(1 - \left(1 - \frac{1-\epsilon}{r}\right)^{r}\right)\texttt{OPT}$$

and obtain

$$f(S) \geq \left(1 - e^{-(1-\epsilon)}\right)\texttt{OPT} \geq \left(1 - \frac{1+2\epsilon}{e}\right)\texttt{OPT} \geq \left(1 - \frac{1}{e} - \epsilon\right)\texttt{OPT}$$

where the second inequality is since $e^x \leq 1 + 2x$ for $0 < x < 1$. The number of rounds is at most $r\epsilon^{-1}\log n$ since there are r iterations of Iterative-Filtering, each of which with at most $\epsilon^{-1}\log n$ iterations of Filter by Lemma 4, with $r = O(\epsilon^{-1}\log n)$. □

D Missing Analysis from Section 3

We introduce some notation and terminology. We now call *the iteration i of filtering during epoch j* the i^{th} iteration discarding elements inside of Filter since the beginning of epoch j, over the multiple invokations of Filter. An element *survives ρ iterations of* Filter *at epoch j* if it has not been discarded at iteration i of filtering during epoch j, for all $i \leq \rho$. Let S_j denote the solution S at epoch $j \in [20\epsilon^{-1}]$, S_j^+ denote $S_j \cup T$ during the last iteration of Amortized-Filtering at epoch j, i.e., the last T such that $f_S(T) < (\epsilon/20)(\texttt{OPT} - f(S))$, and $S_{j,i}$ denote $S_j \cup T$ at the iteration i of filtering during epoch j. Thus, for all $i_1 < i_2$,

$$S_j \subseteq S_{j,i_1} \subseteq S_{j,i_2} \subseteq S_j^+ \subseteq S_{j+1}$$

and $f(S_j^+) - f(S_j) < (\epsilon/20)(\texttt{OPT} - f(S_j))$..

Lemma 5. *For any epoch j and $\epsilon > 0$, let $R_i \sim \mathcal{U}(X)$ be the random set at iteration i of filtering during epoch j. For all $r, \rho > 0$, if epoch j has not ended after ρ iterations of filtering, then*

$$\mathop{\mathbb{E}}_{R_1,\ldots,R_\rho}\left[f_{S_j^+ \cup \left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \geq \left(1 - \frac{\rho}{r} - \epsilon/20\right) \cdot (\mathit{OPT} - f(S_j))$$

where S_j is the set S at epoch j and S_j^+ is the set $S \cup T$ at the last iteration of epoch j.

Proof. Similarly as for Lemma 1, we exploit the fact that if FILTER(N, S, r) has not terminated after ρ iterations, then by the algorithm, the random set $R_i \sim \mathcal{U}(X)$ at iteration i has low expected value. In addition, we also use the bound on the change in value of S during epoch j:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+\cup\left(\cup_{i=1}^{\rho}R_i\right)}(O)\right] \\
=&\mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(O\cup\left(\cup_{i=1}^{\rho}R_i\right)\right)\right] - \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(\left(\cup_{i=1}^{\rho}R_i\right)\right)\right] \\
\geq&\texttt{OPT} - f(S_j^+) - \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(\left(\cup_{i=1}^{\rho}R_i\right)\right)\right] && \text{monotonicity} \\
\geq&\texttt{OPT} - f(S_j) - (\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(\left(\cup_{i=1}^{\rho}R_i\right)\right)\right] && \text{same epoch} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho}\mathop{\mathbb{E}}_{R_i}\left[f_{S_j^+}\left(R_i\right)\right] && \text{subadditivity} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho}\mathop{\mathbb{E}}_{R_i}\left[f_{S_{j,i}}\left(R_i\right)\right] && \text{submodularity} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho}\frac{1-\epsilon}{r}\left(\texttt{OPT} - f(S_j)\right)] && \text{algorithm} \\
=&\left(1-\epsilon/20-\frac{\rho}{r}\right)\cdot\left(\texttt{OPT} - f(S_j)\right).
\end{aligned}$$

□

Lemma 6. *For any epoch j and $\epsilon > 0$, if $r \geq 20\rho\epsilon^{-1}$, then the elements X_ρ that survive ρ iterations of filtering at epoch j satisfy*

$$f_{S_j^+}(X_\rho) \geq (\epsilon/4)(1-\epsilon)\left(\texttt{OPT} - f(S_j)\right).$$

where S_j is the set S at epoch j and S_j^+ is the set $S \cup T$ at the last iteration of epoch j.

Proof. Let j be any epoch. Similarly as for Lemma 2, the proof defines a subset Q of the optimal solution O and then shows show that elements in Q survive ρ iterations of filtering at epoch j and show that $f_{S_j^+}(Q) \geq (\epsilon/4)(1-\epsilon)\left(\texttt{OPT} - f(S_j)\right)$. We define the following marginal contribution Δ_ℓ of each optimal element o_ℓ:

$$\Delta_\ell := \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+\cup O_{\ell-1}\cup\left(\cup_{i=1}^{\rho}R_i\setminus\{o_\ell\}\right)}(o_\ell)\right].$$

We define Q to be the set of optimal elements o_ℓ such that $\Delta_\ell \geq (1-\epsilon/4)\Delta$ where

$$\Delta := \frac{1}{k}\left(1-\frac{\rho}{r}-\epsilon/20\right)\cdot\left(\texttt{OPT} - f(S_j)\right).$$

We first argue that elements in Q survive ρ iterations of filtering at epoch j. For element $o_\ell \in Q$, we have

$$\Delta_\ell \geq (1-\epsilon/4)\Delta \geq \frac{1}{k}(1-\epsilon/4)\left(1-\frac{\rho}{r}-\epsilon/20\right)\cdot\left(\texttt{OPT} - f(S_j)\right) \geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon)\cdot\left(\texttt{OPT} - f(S_j)\right)$$

where the third inequality is by the condition on r. Thus, at iteration $i \leq \rho$, by submodularity,

$$\mathbb{E}_{R_i}\left[f_{S_{j,i}\cup(R_i\setminus\{o_\ell\})}(o_\ell)\right] \geq \mathbb{E}_{R_1,\ldots,R_\rho}\left[f_{S_j^+\cup O_{\ell-1}\cup\left(\cup_{i=1}^{\rho} R_i\setminus\{o_\ell\}\right)}(o_\ell)\right] = \Delta_\ell \geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon)\cdot(\texttt{OPT} - f(S_j))$$

and o_ℓ survives all iterations $i \leq \rho$, for all $o_\ell \in Q$.

Next, we argue that $f_{S_j^+}(Q) \geq (\epsilon/4)(1-\epsilon)(\texttt{OPT} - f(S_j))$. Note that

$$\sum_{\ell=1}^{k} \Delta_\ell \geq \mathbb{E}_{R_1,\ldots,R_\rho}\left[f_{S_j^+\cup\left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \geq \left(1 - \frac{\rho}{r} - \epsilon/20\right)\cdot(\texttt{OPT} - f(S_j)) = k\Delta.$$

where the second inequality is by Lemma 5. Next, observe that

$$\sum_{\ell=1}^{k} \Delta_\ell = \sum_{o_\ell \in Q} \Delta_\ell + \sum_{j\in O\setminus Q} \Delta_\ell \leq \sum_{o_\ell\in Q} \Delta_\ell + k(1-\epsilon/4)\Delta.$$

By combining the two inequalities above, we get $\sum_{o_\ell\in Q} \Delta_\ell \geq k\epsilon\Delta/4$. Thus, by submodularity,

$$f_{S_j^+}(Q) \geq \sum_{o_\ell\in Q} f_{S_j^+\cup O_{\ell-1}}(o_\ell) \geq \sum_{o_\ell\in Q} \mathbb{E}_{R_1,\ldots,R_\rho}\left[f_{S_j^+\cup O_{\ell-1}\cup\left(\cup_{i=1}^{\rho} R_i\setminus\{o_\ell\}\right)}(o_\ell)\right] = \sum_{o_\ell\in Q} \Delta_\ell \geq k\epsilon\Delta/4.$$

We conclude that

$$f_{S_j^+}(X_\rho) \geq f_{S_j^+}(Q) \geq k\epsilon\Delta/4 = (\epsilon/4)\left(1 - \frac{\rho}{r}\right)\cdot(\texttt{OPT} - f(S_j)) \geq (\epsilon/4)\cdot(1-\epsilon)\cdot(\texttt{OPT} - f(S_j)).$$

where the first inequality is by monotonicity and since $Q \subseteq X_\rho$ is a set of surviving elements. □

Theorem 2. *For any constant $\epsilon > 0$, when using parameter $r = 40\epsilon^{-2}\log n$,* Amortized-Filtering *obtains a $1 - 1/e - \epsilon$ approximation in $\mathcal{O}(\log n)$-adaptive steps.*

Proof. First, consider the case where the algorithm terminates after r iterations of adding elements to S. Let S_i denote the solution S at the ith iteration. Amortized-Filtering increases the value of the solution S by at least $(1-\epsilon)(\texttt{OPT} - f(S))/r$ at every iteration with k/r new elements. Thus,

$$f(S_i) \geq f(S_{i-1}) + \frac{1-\epsilon}{r}(\texttt{OPT} - f(S_{i-1}))$$

and we obtain $f(S) \geq \left(1 - e^{-(1-\epsilon)}\right)\texttt{OPT} \geq \left(1 - e^{-1} - \epsilon\right)\texttt{OPT}$ similarly as for Theorem 1.

Next, consider the case where the algorithm terminated after $(\epsilon/20)^{-1}$ epochs. At every epoch j, the algorithm increases the value of the solution S by $(\epsilon/20)(\texttt{OPT} - f(S_j))$. Thus,

$$f(S_j) \geq f(S_{j-1}) + (\epsilon/20)(\texttt{OPT} - f(S_{j-1})).$$

Similarly as in the first case, we get that after $(\epsilon/20)^{-1}$ epochs, $f(S) \geq (1 - e^{-1})\texttt{OPT}$.

The total number of rounds of adaptivity of Amortized-Filtering is at most $O(\epsilon^{-3}\log n)$ since there are at most $r = 40\epsilon^{-2}\log n$ iterations of adding elements and at most $(\epsilon/20)^{-1}$ epochs with, by Lemma 7, at most $\epsilon^{-1}\log n$ filtering iterations each. The queries at each filtering iteration are independent and can be evaluated in parallel. □

E The Full Algorithm

E.1 Description of the full algorithm

E.1.1 Estimates of expectations in one round via sampling

We show that the expected value of a random set and the expected marginal contribution of elements to a random set can be estimated arbitrarily well in one round, which is needed for the Iterative-Filtering and Amortized-Filtering algorithms. Recall that $\mathcal{U}(X)$ denotes the uniform distribution over subsets of X of size k/r. The values we are interested in estimating are $\mathbb{E}_{R\sim\mathcal{U}(X)}[f_S(R)]$ and $\mathbb{E}_{R\sim\mathcal{U}(X)}\left[f_{S\cup(R\setminus a)}(a)\right]$. We denote the corresponding estimates by $v_S(X)$ and $v_S(X,a)$, which are computed in Algorithms 4 and 5. These algorithms first sample m sets from $\mathcal{U}(X)$, where m is the sample complexity, then query the desired sets to obtain a random realization of $f_S(R)$ and $f_{S\cup(R\setminus a)}(a)$, and finally averages the m random realizations of these values.

Algorithm 4 EstimateSet: Computes estimate $v_S(X)$ of $\mathbb{E}_{R\sim\mathcal{U}(X)}[f_S(R)]$.

Input: set $X \subseteq N$, sample complexity m.

  Query $f(S)$ and $f(S\cup R_i)$ for all samples $R_1,\ldots,R_m \overset{\text{i.i.d.}}{\sim} \mathcal{U}(X)$

  return $\frac{1}{m}\sum_{i=1}^m f_S(R_i)$

Algorithm 5 EstimateMarginal: Computes estimate $v_S(X,a)$ of $\mathbb{E}_{R\sim\mathcal{U}(X)}\left[f_{S\cup(R\setminus a)}(a)\right]$.

Input: set $X \subseteq N$, sample complexity m, element $a \in N$.

  Query $f(S\cup(R_i\cup a))$ and $f(S\cup(R_i\setminus a))$ for all samples $R_1,\ldots,R_m \overset{\text{i.i.d.}}{\sim} \mathcal{U}(X)$

  return $\frac{1}{m}\sum_{i=1}^m f_{S\cup(R_i\setminus a)}(a)$

Using standard concentration bounds, the estimates computed by these algorithms are arbitrarily good for a sufficiently large sample complexity m. We state the version of Hoeffding's inequality which is used to bound the error of these estimates.

Lemma 8 (Hoeffding's inequality). *Let $S_1,\ldots,S_n$ be independent random variables with values in $[0,b]$. Let $S=\frac{1}{m}\sum_{i=1}^m S_i$. Then for any $\epsilon>0$,*

$$\Pr\left[|S-\mathbb{E}[S]|\geq\epsilon\right]\leq 2e^{-2m\epsilon^2/b^2}.$$

We are now ready to show that these estimates are arbitrarily good.

Lemma 9. *Let $m=\frac{1}{2}\left(\frac{\texttt{OPT}}{\epsilon}\right)^2\log\left(\frac{2}{\delta}\right)$, then for all $S,X\subseteq N$, and $a\in N$, with probability $1-\delta$ over the samples $R_1,\ldots,R_m$,*

$$\left|v_S(X,a)-\mathop{\mathbb{E}}_{R\sim\mathcal{U}(X)}\left[f_{S\cup(R\setminus a)|a\in R}(a)\right]\right|\leq\epsilon \qquad \textit{and} \qquad \left|v_S(X)-\mathop{\mathbb{E}}_{R\sim\mathcal{U}(X)}\left[f_S(R)\right]\right|\leq\epsilon.$$

Thus, with $m=n\left(\frac{\texttt{OPT}}{\epsilon}\right)^2\log\left(\frac{2n}{\delta}\right)$ total samples in one round, with probability $1-\delta$, it holds that $v_S(X)$ and $v_S(X,a)$, for all $a\in N$, are ϵ-estimates.

Proof. Note that

$$\mathbb{E}\left[v_S(X)\right] = \underset{R\sim\mathcal{U}(X)}{\mathbb{E}}\left[f_S(R)\right] \qquad \text{and} \qquad \mathbb{E}\left[v_S(X,a)\right] = \underset{R\sim\mathcal{U}(X)}{\mathbb{E}}\left[f_{S\cup(R\setminus a)}(a)\right]$$

Since all queries are of size at most k, their values are all bounded by $\texttt{OPT}$. Thus, by Hoeffding's inequality with $m = \frac{1}{2}\left(\frac{\texttt{OPT}}{\epsilon}\right)^2 \log\left(\frac{2}{\delta}\right)$, we get

$$\Pr\left[\left|v_S(X) - \underset{R\sim\mathcal{U}(X)}{\mathbb{E}}\left[f_S(R)\right]\right| \geq \epsilon\right], \Pr\left[\left|v_S(X,a) - \underset{R\sim\mathcal{U}(X)}{\mathbb{E}}\left[f_{S\cup(R\setminus a)}(a)\right]\right| \geq \epsilon\right] \leq 2e^{-\frac{2m\epsilon^2}{\texttt{OPT}^2}} \leq \delta$$

for $\epsilon > 0$. Thus, with $m = n\left(\frac{\texttt{OPT}}{\epsilon}\right)^2 \log\left(\frac{2n}{\delta}\right)$ total samples in one round, by a union bound over each of the estimates holding with probability $1-\delta/n$ individually, we get that all the estimates hold simultaneously with probability $1-\delta$. $\square$

We can now describe the (almost) full version of the main algorithm which uses these estimates. Note that we can force the algorithm to stop after any round to obtain the desired adaptive complexity with probability 1. In our analysis, the loss from the event that the algorithm is forced to stop when the desired adaptivity is reached is accounted for in the δ probability of failure of the approximation guarantee of the algorithm.

Algorithm 6 AMORTIZED-FILTERING-PROXY

Input: bound on number of iterations r, sample complexity m, proxy $v^\star$

```
S ← ∅
for 20/ε epochs do
  X ← N, T ← ∅
  v_S(X) ← EstimateSet(X, m)
  while v_S(X) < (ε/20)(v* − f(S)) and |S ∪ T| < k do
    for a ∈ X do                                              Non-adaptive loop
      v_S(X, a) ← EstimateMarginal(X \ S, m, a)
    X ← X \ {a : v_S(X, a) < (1 + ε/2)(1 − ε)(v* − f(S))/k}
    T ← T ∪ R, where R ~ U(X)
    v_S(X) ← EstimateSet(X, m)
  S ← S ∪ T
return S
```

E.1.2 Estimates of OPT

The main idea to estimate $\texttt{OPT}$ is to have $O(\epsilon^{-1}\log n)$ values v_i such that one of them is guaranteed to be a $(1-\epsilon)$-approximation to $\texttt{OPT}$. To obtain such values, we use the simple observation that the singleton $a^\star$ with largest value is at least a $1/n$ approximation to $\texttt{OPT}$.

More formally, let $a^\star = \operatorname{argmax}_{a\in N} f(a)$ be the optimal singleton, and $v_i = (1+\epsilon)^i \cdot f(a^\star)$. We argue that there exists some $i \in \left[\epsilon^{-1}\log n\right]$ such that $\texttt{OPT} \leq v_i \leq (1+\epsilon)\cdot\texttt{OPT}$. By submodularity, we get $f(a^\star) \geq \frac{1}{k}\texttt{OPT} \geq \frac{1}{n}\texttt{OPT}$. By monotonicity, we have $f(a^\star) \leq \texttt{OPT}$. Combining these two inequalities, we get $v_0 \leq \texttt{OPT} \leq v_{\epsilon^{-1}\log n}$. By the definition of v_i, we then conclude that there must exists some $i \in \left[\epsilon^{-1}\log n\right]$ such that $\texttt{OPT} \leq v_i \leq (1+\epsilon)\cdot\texttt{OPT}$.

Since the solution obtained for the unknown v_i which approximates $\texttt{OPT}$ well is guaranteed to be a good solution, we run the algorithm in parallel for each of these values and return the solution with largest value. We obtain the full algorithm Amortized-Filtering-Full which we describe next.

Algorithm 7 Amortized-Filtering-Full

Input: bounds on number of outer-iterations r, sample complexity m, and precision ϵ

Query $f(a_1), \ldots, f(a_n)$

$a^\star \leftarrow \text{argmax}_{a_i} f(a_i)$

for $i \in \left\{0, \ldots, \epsilon^{-1} \log n\right\}$ **do** Non-adaptive loop

$v^\star \leftarrow (1+\epsilon)^i \cdot f(a^\star)$

$X_i \leftarrow$ Amortized-Filtering-Proxy$(v^\star)$

return best solution X_i: $\text{argmax}_{X_i : i \in [\epsilon^{-1} \log n]} f(X_i)$

E.2 Analysis of the Amortized-Filtering-Full algorithm

We bound the number of elements removed from X in each round of the full algorithm.

Lemma 10. *Assume* $(1-\epsilon/20)\texttt{OPT} \leq v^\star \leq \texttt{OPT}$ *and* $0 < \epsilon < 1/2$. *For any* S *and* r*, at the iteration* i *of filtering during any epoch* j *of* Amortized-Filtering-Proxy*, with probability* $1-\delta$*, we have*

$$|X_{i+1}| < \frac{1}{1+\epsilon/3}|X_i|.$$

where X_i *and* X_{i+1} *are the set* X *before and after this* i*th iteration and with sample complexity* $m = \mathcal{O}\left(n\left(\frac{k+r}{\epsilon}\right)^2 \log\left(\frac{n}{\delta}\right)\right)$ *at each round.*

Proof. At a high level, since the surviving elements must have high value and a random set has low value, we can then use the thresholds to bound how many such surviving elements there can be while also having a random set of low value. To do so, we focus on the value of $f(R_i \cap X_{i+1})$ of the surviving elements X_{i+1} in a random set $R_i \sim \mathcal{D}_{X_i}$.

$$
\begin{aligned}
&\mathbb{E}\left[f_S(R_i \cap X_{i+1})\right] \\
\geq &\mathbb{E}\left[\sum_{a \in R_i \cap X_{i+1}} f_{S \cup (R_i \cap X_{i+1} \setminus a)}(a)\right] && \text{submodularity} \\
\geq &\mathbb{E}\left[\sum_{a \in X_{i+1}} \mathbb{1}_{a \in R_i} \cdot f_{S \cup (R_i \setminus a)}(a)\right] && \text{submodularity} \\
= &\sum_{a \in X_{i+1}} \mathbb{E}\left[\mathbb{1}_{a \in R_i} \cdot f_{S \cup (R_i \setminus a)}(a)\right]. \\
= &\sum_{a \in X_{i+1}} \Pr\left[a \in R_i\right] \cdot \mathbb{E}\left[f_{S \cup (R_i \setminus a)}(a) | a \in R_i\right] \\
\geq &\sum_{a \in X_{i+1}} \Pr\left[a \in R_i\right] \cdot \mathbb{E}\left[f_{S \cup (R_i \setminus a)}(a)\right] && \text{submodularity} \\
\geq &\sum_{a \in X_{i+1}} \Pr\left[a \in R\right] \cdot \left(v_S(X,a) - \frac{\epsilon}{20k}\left(1+\epsilon/2\right)\left(1-\epsilon\right)\left(v^\star - f(S)\right)\right) && \text{Lemma 9} \\
\geq &\sum_{a \in X_{i+1}} \Pr\left[a \in R_i\right] \cdot \left(\left(1-\epsilon/20\right)\frac{1}{k}\left(1+\epsilon/2\right)\left(1-\epsilon\right)\left(v^\star - f(S)\right)\right) && \text{algorithm} \\
= &|X_{i+1}| \cdot \frac{k}{r|X_i|} \cdot \left(\left(1-\epsilon/20\right)\frac{1}{k}\left(1+\epsilon/2\right)\left(1-\epsilon\right)\left(v^\star - f(S)\right)\right) \\
= &|X_{i+1}| \cdot \frac{1}{r|X_i|}\left(1-\epsilon/20\right) \cdot \left(1+\epsilon/2\right)\left(1-\epsilon\right)\left(v^\star - f(S)\right).
\end{aligned}
$$

Next, by the algorithm and by Lemma 9,

$$
(1+\epsilon/20)\frac{1}{r}\left(1-\epsilon\right)\left(v^\star - f(S)\right) \geq v_S(X) + \frac{\epsilon}{20}\frac{1}{r}\left(1-\epsilon\right)\left(v^\star - f(S)\right) > \mathbb{E}\left[f_S(R_i)\right].
$$

By monotonicity, we get $\mathbb{E}\left[f_S(R_i)\right] \geq \mathbb{E}\left[f_S(R_i \cap X_{i+1})\right]$. Finally, by combining the above inequalities, we conclude that

$$
|X_{i+1}| \leq \frac{1}{(1-\epsilon/20)^2(1+\epsilon/2)}|X_i| \leq \frac{1}{1+\epsilon/3}|X_i|
$$

where, with probability $1-\delta$, all the estimates hold with sample complexity

$$
m = \mathcal{O}\left(n\left(\frac{k+r}{\epsilon}\right)^2 \log\left(\frac{n}{\delta}\right)\right)
$$

per round by Lemma 9 and since $v^\star \geq (1-\epsilon/20)\texttt{OPT}$. □

Lemma 11. *Assume* $(1-\epsilon/20)\texttt{OPT} \leq v^\star \leq \texttt{OPT}$ *and* $0 < \epsilon < 1/2$. *For any epoch* j, *let* $R_i \sim \mathcal{U}(X)$ *be the random set at iteration* i *of filtering during epoch* j. *For all* $r, \rho > 0$, *if epoch* j *of* Amortized-Filtering-Proxy *has not ended after* ρ *iterations of filtering, then, with probability* $1-\delta$,

$$
\mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+ \cup \left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \geq \left(1 - \frac{\rho}{r} - \epsilon/10\right) \cdot \left(\texttt{OPT} - f(S_j)\right)
$$

where S_j and S_j^+ are the set S at start and end of epoch j, with sample complexity

$$m = \mathcal{O}\left(\rho \left(\frac{\rho}{\epsilon}\right)^2 \log\left(\frac{\rho}{\delta}\right)\right)$$

per epoch.

Proof. By the condition to have a filtering iteration, a random set $R \sim \mathcal{D}$ must have low value at each of the FILTER iterations:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+ \cup \left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \\
=&\mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(O \cup \left(\cup_{i=1}^{\rho} R_i\right)\right)\right] - \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(\left(\cup_{i=1}^{\rho} R_i\right)\right)\right] \\
\geq&\texttt{OPT} - f(S_j^+) - \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(\left(\cup_{i=1}^{\rho} R_i\right)\right)\right] && \text{monotonicity} \\
\geq&\texttt{OPT} - f(S_j) - \frac{\epsilon}{20}\left(\texttt{OPT} - f(S_j)\right) - \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+}\left(\left(\cup_{i=1}^{\rho} R_i\right)\right)\right] && \text{same epoch} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho} \mathop{\mathbb{E}}_{R_i}\left[f_{S_j^+}(R_i)\right] && \text{subadditivity} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho} \mathop{\mathbb{E}}_{R_i}\left[f_{S_{j,i}}(R_i)\right] && \text{submodularity} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho} \mathop{\mathbb{E}}_{R_i}\Big[v_{S_{i,j}}(X_i) + \frac{\epsilon}{20\rho}\left(v^\star - f(S_j)\right)\Big] && \text{Lemma 9} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho} \mathop{\mathbb{E}}_{R_i}\Big[v_{S_{i,j}}(X_i) + \frac{\epsilon}{20\rho}\left(\texttt{OPT} - f(S_j)\right)\Big] && v^\star \geq \texttt{OPT} \\
\geq&(1-\epsilon/20)\left(\texttt{OPT} - f(S_j)\right) - \sum_{i=1}^{\rho}\left(\frac{1-\epsilon}{r} + \frac{\epsilon}{20\rho}\right)\left(\texttt{OPT} - f(S_j)\right)] && \text{algorithm} \\
\geq&\left(1 - \epsilon/20 - \frac{\rho}{r} - \epsilon/20\right) \cdot \left(\texttt{OPT} - f(S_j)\right).
\end{aligned}$$

where, with probability $1-\delta$, the estimates hold for all ρ iterations with sample complexity $m = \mathcal{O}\left(\rho\left(\frac{\rho}{\epsilon}\right)^2 \log\left(\frac{\rho}{\delta}\right)\right)$ per round by Lemma 9 and since $v^\star \geq (1-\epsilon/20)\texttt{OPT}$. ☐

Lemma 12. *Assume $(1-\epsilon/20)\texttt{OPT} \leq v^\star \leq \texttt{OPT}$ and $0 < \epsilon < 1/2$. If $r \geq 20\rho\epsilon^{-1}$, then, with probability $1-\delta$, the set X_ρ of elements that survive ρ iterations of filtering at any epoch j of* AMORTIZED-FILTERING-PROXY *satisfies*

$$f_{S_j^+}(X_\rho) \geq (\epsilon/4)(1-\epsilon)\left(v^\star - f(S_j)\right).$$

where S_j and S_j^+ are the set S at the start and end of epoch j and with sample complexity $m = \mathcal{O}\left(\rho\left(\frac{k}{\epsilon}\right)^2 \log\left(\frac{\rho}{\delta}\right)\right)$ per epoch.

Proof. Let j be any epoch. Similarly as for Lemma 2, the proof defines a subset T of the optimal solution O and then shows show that elements in T survive ρ iterations of filtering at epoch j and show that $f_{S_j^+}(T) \geq (\epsilon/4)\,(1-\epsilon)\,(\texttt{OPT} - f(S_j))$. We define the following marginal contribution Δ_ℓ of each optimal element o_ℓ:

$$\Delta_\ell := \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+\cup O_{\ell-1}\cup\left(\cup_{i=1}^{\rho} R_i\setminus\{o_\ell\}\right)}(o_\ell)\right].$$

We define T to be the set of optimal elements o_ℓ such that $\Delta_\ell \geq (1-\epsilon/4)\Delta$ where

$$\Delta := \frac{1}{k}\left(1-\frac{\rho}{r}-\epsilon/10\right)\cdot(\texttt{OPT} - f(S_j)).$$

We first argue that elements in T survive ρ iterations of filtering at epoch j. For element $o_\ell \in T$, we have

$$\begin{aligned}\Delta_\ell \geq (1-\epsilon/4)\Delta &\geq \frac{1}{k}(1-\epsilon/4)\left(1-\frac{\rho}{r}-\epsilon/10\right)\cdot(\texttt{OPT} - f(S_j))\\ &\geq \frac{1}{k}(1-5\epsilon/12)\cdot(\texttt{OPT} - f(S_j))\\ &\geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon)(1+\epsilon/20)\cdot(\texttt{OPT} - f(S_j)).\end{aligned}$$

where the third inequality is by the condition on r. Thus, at iteration $i \leq \rho$, by Lemma 9 and by submodularity,

$$\begin{aligned}v_{S_{j,i}}(X_i) + \frac{\epsilon}{20}\cdot\frac{1}{k}(1+\epsilon/2)(1-\epsilon)\,(\texttt{OPT} - f(S_j)) &\geq \mathop{\mathbb{E}}_{R_i}\left[f_{S_{j,i}\cup(R_i\setminus\{o_\ell\})}(o_\ell)\right]\\ &\geq \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+\cup O_{\ell-1}\cup\left(\cup_{i=1}^{\rho} R_i\setminus\{o_\ell\}\right)}(o_\ell)\right]\\ &= \Delta_\ell\\ &\geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon)(1+\frac{\epsilon}{20})\cdot(\texttt{OPT} - f(S_j)),\end{aligned}$$

where, with probability $1-\delta$, the estimates hold for all ρ iterations with sample complexity $m = \mathcal{O}\left(\rho\left(\frac{k}{\epsilon}\right)^2\log\left(\frac{\rho}{\delta}\right)\right)$. Thus,

$$v_{S_{j,i}}(X_i) \geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon)\cdot(\texttt{OPT} - f(S_j)) \geq \frac{1}{k}(1+\epsilon/2)(1-\epsilon)\cdot(v^\star - f(S_j))$$

and o_ℓ survives all iterations $i \leq \rho$, for all $o_\ell \in T$.

Next, we argue that $f_{S_j^+}(T) \geq (\epsilon/4)\,(1-\epsilon)\,(\texttt{OPT} - f(S_j))$. Note that

$$\sum_{\ell=1}^{k}\Delta_\ell \geq \mathop{\mathbb{E}}_{R_1,\dots,R_\rho}\left[f_{S_j^+\cup\left(\cup_{i=1}^{\rho} R_i\right)}(O)\right] \geq \left(1-\frac{\rho}{r}-\epsilon/10\right)\cdot(\texttt{OPT} - f(S_j)) = k\Delta.$$

where the second inequality is by Lemma 11. Next, observe that

$$\sum_{\ell=1}^{k}\Delta_\ell = \sum_{o_\ell\in T}\Delta_\ell + \sum_{j\in O\setminus T}\Delta_\ell \leq \sum_{o_\ell\in T}\Delta_\ell + k(1-\epsilon/4)\Delta.$$

By combining the two inequalities above, we get $\sum_{o_\ell \in T} \Delta_\ell \geq k\epsilon\Delta/4$. Thus, by submodularity,

$$f_{S_j^+}(T) \geq \sum_{o_\ell \in T} f_{S_j^+ \cup O_{\ell-1}}(o_\ell) \geq \sum_{o_\ell \in T} \mathbb{E}_{R_1,\ldots,R_\rho}\left[f_{S_j^+ \cup O_{\ell-1} \cup \left(\cup_{i=1}^{\rho} R_i \setminus \{o_\ell\}\right)}(o_\ell)\right] = \sum_{o_\ell \in T} \Delta_\ell \geq k\epsilon\Delta/4.$$

We conclude that

$$\begin{aligned} f_{S_j^+}(X_\rho) \geq f_{S_j^+}(T) \geq k\epsilon\Delta/4 = \frac{\epsilon}{4}\left(1 - \frac{\rho}{r} - \epsilon/10\right) \cdot (\texttt{OPT} - f(S_j)) &\geq \frac{\epsilon}{4} \cdot (1-\epsilon) \cdot (\texttt{OPT} - f(S_j)) \\ &\geq \frac{\epsilon}{4} \cdot (1-\epsilon) \cdot (v^\star - f(S_j)) . \end{aligned}$$

where the first inequality is by monotonicity and since $T \subseteq X_\rho$ is a set of surviving elements. □

Lemma 13. *Assume* $(1-\epsilon/20)\texttt{OPT} \leq v^\star \leq \texttt{OPT}$ *and constant* $0 < \epsilon < 1/2$. *For any epoch* j*, with probability* $1-\delta$*, there are at most* $3\epsilon^{-1}\log n$ *iterations of filtering when the number of iterations of* Amortized-Filtering-Proxy *with* $r = 60\epsilon^{-2}\log n$ *and with sample complexity* $m = \mathcal{O}\left(n\left(\frac{k+r}{\epsilon}\right)^2 \log\left(\frac{n}{\delta}\right)\right)$ *at each round.*

Proof. If an epoch j has not yet terminated after $3\epsilon^{-1}\log n$ iterations of filtering, then, by Lemma 10, at most k/r elements survived these $3\epsilon^{-1}\log n$ iterations. By Lemma 12, with the set X_ρ of elements that survive these $3\epsilon^{-1}\log n$ iterations is such that $f_{S_j^+}(X_\rho) \geq (\epsilon/4) \cdot (1-\epsilon)(v^\star - f(S_j))$. Since there are at most k/r surviving elements, $R = X_\rho$ for $R \sim \mathcal{U}(X_\rho)$ and

$$\mathbb{E}\left[f_{S_j^+}(R)\right] \geq f_{S_j^+}(X_\rho) \geq \frac{\epsilon}{4} \cdot (1-\epsilon)(v^\star - f(S_j)) \geq \frac{1}{r} \cdot (1-\epsilon)\left(v^\star - f(S_j^+)\right)$$

where the last inequality is by monotonicity since $S_j \subseteq S_j^+$. Thus the current call to the Filter subroutine terminates and X_ρ is added to S_j^+ by the algorithm. Next,

$$f_{S_j}(S_j^+ \cup X_\rho) \geq f_{S_j}(X_\rho) \geq f_{S_j^+}(X_\rho) \geq \frac{\epsilon}{4} \cdot (1-\epsilon)(\texttt{OPT} - f(S_j)) \geq \frac{\epsilon}{20}(v^\star - f(S_j))$$

where the first inequality is by monotonicity and the second by submodularity. Thus, epoch j ends. □

We are now ready to prove the main result for Amortized-Filtering-Proxy.

Lemma 14. *Assume* $(1-\epsilon/20)\texttt{OPT} \leq v^\star \leq \texttt{OPT}$. *The* Amortized-Filtering-Proxy *algorithm is a* $\mathcal{O}\left(\epsilon^{-2}\log n\right)$*-adaptive algorithm that obtains, with probability* $1-\delta$*, a* $1-1/e-\epsilon$ *approximation, with* $r = 60\epsilon^{-2}\log n$*. Its sample complexity is* $m = \mathcal{O}\left(n(k+\log n)^2 \epsilon^{-4}\log\left(\frac{n}{\delta}\right)\right)$ *at each round.*

Proof. First, consider the case where the algorithm terminates after r iterations of adding elements to S. Let S_i denote the solution S at the ith iteration of Amortized-Filtering-Proxy. The algorithm increases the value of the solution S by at least $(1-\epsilon)(v^\star - f(S))/r$ at every iteration with k/r new elements. Thus,

$$f(S_i) \geq f(S_{i-1}) + \frac{1-\epsilon}{r}(v^\star - f(S_{i-1}))$$

and we obtain

$$f(S) \geq \left(1 - e^{-(1-\epsilon)}\right) v^\star \geq \left(1 - e^{-(1-\epsilon)} - \epsilon/20\right) \texttt{OPT} \geq \left(1 - \frac{1+2\epsilon}{e} - \epsilon/20\right) \texttt{OPT} \geq \left(1 - \frac{1}{e} - \epsilon\right) \texttt{OPT}$$

similarly as for Theorem 1.

Next, consider the case where the algorithm terminated after $(\epsilon/20)^{-1}$ epochs. At every epoch j, the algorithm increases the value of the solution S by $(\epsilon/20)(v^\star - f(S_j))$. Thus,

$$f(S_j) \geq f(S_{j-1}) + \frac{\epsilon}{20} \cdot (v^\star - f(S_{j-1}))\,.$$

Similarly as in the first case, we get that after $(\epsilon/20)^{-1}$ epochs, $f(S) \geq (1 - e^{-1})v^\star \geq (1 - e^{-1} - \epsilon/20)v^\star$.

The number of rounds is at most $60\epsilon^{-2}\log n + 60\epsilon^{-2}\log n$ since there are at most $r = 60\epsilon^{-2}\log n$ iterations of adding elements and at most $(\epsilon/20)^{-1}$ epochs, each of which with at most $3\epsilon^{-1}\log n$ filtering iterations by Lemma 13. $\square$

Theorem 3. *For any $\epsilon \in (0, 1/2)$, there exists an algorithm that obtains a $1 - 1/e - \epsilon$ approximation with probability $1 - \delta$ in $\mathcal{O}(\epsilon^{-2}\log n)$ adaptive steps. Its query complexity in each round is $\mathcal{O}\left(nk^2\log^3(n)\epsilon^{-5}\log\left(\frac{n}{\delta}\right)\right)$.*

Proof. With $20\epsilon^{-1}\log n$ different guesses $v^\star$ of $\texttt{OPT}$, there is at least one $v^\star$ in Amortized-Filtering-Full that is such that $(1 - \epsilon/20)\texttt{OPT} \leq v^\star \leq \texttt{OPT}$. The solution to Amortized-Filtering-Proxy$(v^\star)$ with such a $v^\star$ is then, with probability $1-\delta$, a $1 - 1/e - \epsilon$ approximation with sample complexity $m = \mathcal{O}\left(n\left(k + \log n\right)^2 \epsilon^{-4}\log\left(\frac{n}{\delta}\right)\right)$ at each round and with adaptivity $\mathcal{O}\left(\epsilon^{-2}\log n\right)$ by Lemma 14. Since Amortized-Filtering-Full picks the best solution returned by all instances of Amortized-Filtering-Proxy, it also obtains with probability $1-\delta$, a $1-1/e-\epsilon$ approximation.

Finally, since there are $20\epsilon^{-1}\log n$ non-adaptive instances of Amortized-Filtering-Proxy, each with adaptivity $\mathcal{O}\left(\epsilon^{-2}\log n\right)$, the total number of adaptive rounds of Amortized-Filtering-Full is $\mathcal{O}\left(\epsilon^{-2}\log n\right)$. The total query complexity per round over all guesses is

$$m = \mathcal{O}\left(nk^2\log^3(n)\epsilon^{-5}\log\left(\frac{n}{\delta}\right)\right).$$

$\square$