The Adaptive Complexity of Maximizing a Submodular Function

Eric Balkanski* Yaron Singer†

Abstract

In this paper we study the adaptive complexity of submodular optimization. Informally, the adaptive complexity of a problem is the minimal number of sequential rounds required to achieve a constant factor approximation when polynomially-many queries can be executed in parallel at each round. Adaptivity is a fundamental concept that is heavily studied in computer science, largely due to the need for parallelizing computation. Somewhat surprisingly, very little is known about adaptivity in submodular optimization. For the canonical problem of maximizing a monotone submodular function under a cardinality constraint, to the best of our knowledge, all that is known to date is that the adaptive complexity is between 1 and $\Omega(n)$.

Our main result in this paper is a tight characterization showing that the adaptive complexity of maximizing a monotone submodular function under a cardinality constraint is $\tilde{\Theta}(\log n)$:

- We describe an algorithm which requires $\mathcal{O}(\log n)$ sequential rounds and achieves an approximation that is arbitrarily close to $1/3$;
- We show that no algorithm can achieve an approximation better than $\mathcal{O}(\frac{1}{\log n})$ with fewer than $\mathcal{O}(\frac{\log n}{\log \log n})$ rounds.

Thus, when allowing for parallelization, our algorithm achieves a constant factor approximation exponentially faster than any known existing algorithm for submodular maximization.

Importantly, the approximation algorithm is achieved via *adaptive sampling* and complements a recent line of work on optimization of functions learned from data. In many cases we do not know the functions we optimize and learn them from labeled samples. Recent results show that no algorithm can obtain a constant factor approximation guarantee using polynomially-many labeled samples as in the `PAC` and `PMAC` models, drawn from any distribution [BRS17, BS17a]. Since learning with non-adaptive samples over any distribution results in a sharp impossibility, we consider learning with adaptive samples where the learner obtains $\text{poly}(n)$ samples drawn from a distribution of her choice in every round. Our result implies that in the realizable case, where there is a true underlying function generating the data, $\tilde{\Theta}(\log n)$ batches of adaptive samples are *necessary and sufficient* to approximately "learn to optimize" a monotone submodular function under a cardinality constraint.

*School of Engineering and Applied Sciences, Harvard University. ericbalkanski@g.harvard.edu. This research was supported by a Google PhD Fellowship, by NSF grant CCF-1301976, CAREER CCF-1452961, and by BSF grant 2014389.

†School of Engineering and Applied Sciences, Harvard University. yaron@seas.harvard.edu. This research was supported by NSF grant CCF-1301976, CAREER CCF-1452961, and by BSF grant 2014389.

1 Introduction

In this paper we study the adaptive complexity of maximizing a submodular function. For the past several decades submodularity has been heavily studied in theoretical computer science, machine learning, and operations research. This is largely due to the fact that submodular functions capture a broad range of applications in diverse domains and are amendable to optimization.

In many cases we do not know the objective function we optimize and instead learn it from data. In the standard notions of learnability for submodular functions such as PAC [Val84] and its generalization PMAC [BH11], the input is a collection of sampled sets and their function values, and the goal is to produce a surrogate that mimics the behavior of the function on samples drawn from the same distribution (e.g. [BH11, FK14, BDF+12, BCIW12, FV13, FV15, Bal15, NPS15]).

In order to investigate the approximation guarantees achievable when a function is learned in the PAC and PMAC models, a recent line of work has been devoted to *optimization from samples* [BRS17]. In this framework the input is a collection of samples and the goal is to find a solution that approximates the optimum. The main result shows that for the canonical problem of maximizing a submodular function under a cardinality constraint, no algorithm can obtain a constant factor approximation guarantee given access to polynomially-many samples drawn from any distribution [BRS17]. This result holds even when the functions are coverage functions which are heavily used in applications and are PMAC-learnable [FK14, BDF+12]. Similar impossibility results hold for submodular minimization [BS17a] and convex optimization [BS17b], even when the objective functions are PAC-learnable. Thus, it is generally impossible to obtain reasonable approximation guarantees for optimization problems that are in P and APX when the objective is learned with polynomially-many samples, even when it is PMAC or PAC learnable.

1.1 Adaptivity

The inapproximability for optimization from samples is a consequence of *non-adaptivity*: any algorithm that only has access to samples of function values cannot make adaptive queries and this restriction inhibits reasonable approximation guarantees. Informally, the *adaptivity* of an algorithm can be quantified in terms of the number of sequential rounds of queries it makes, where every round allows for polynomially-many parallel queries.

Definition. *Given an oracle f, an algorithm is r-**adaptive** if every query q to the oracle f occurs at a* round $i \in [r]$ *such that q is independent of the answers $f(q')$ to all other queries q' at round i.*

Adaptivity is a fundamental concept that is studied across a wide spectrum of areas in computer science (see discussion in Appendix A.1). In the context of submodular optimization, the oracle for a function $f : 2^N \to \mathbb{R}$ is a value oracle that receives a set $S \subseteq N$ and returns its value $f(S)$. An algorithm is then r-adaptive if every query S to the oracle occurs at a round $i \in [r]$ such that S is independent of the values $f(S')$ of all other queries S' at round i. Somewhat surprisingly, the concept of adaptivity which quantifies complexity in a parallel computing model has not been explored for submodular optimization. There is vast literature on submodular optimization in the Map-Reduce model which addresses the challenges associated with processing data that exceeds memory capacity (e.g. [CKT10, KMVV15, MKSK13, MZ15, MKBK15, BENW15, BENW16, EMZ17]), but these algorithms are inherently sequential and $\Omega(n)$-adaptive in the worst case (see discussion in Appendix A.2), where n is the size of the ground set N.

1.2 The adaptivity landscape of submodular optimization

The *adaptive complexity* of an optimization problem is the minimum number of rounds r such that there exists an r-adaptive algorithm which achieves a *constant factor* approximation with $\text{poly}(n)$ queries made in every round. For unconstrained submodular maximization the adaptive complexity is trivially 0 as a random subset is a $1/4$ approximation to the optimal solution [FMV11]. For the canonical problem of maximizing a monotone submodular function under a cardinality constraint k, however, very little is known. The adaptive complexity must be strictly larger than 1 since the main impossibility result for optimization from samples implies that no constant factor approximation is achievable with non-adaptive queries [BRS17]. On the other hand, the celebrated greedy algorithm which achieves the optimal $1-1/e$ approximation guarantee by iteratively adding the element with largest marginal contribution is trivially k-adaptive. In the worst case, $k \in \Omega(n)$. All constant factor approximation algorithms we are aware of for maximizing a submodular function under a cardinality constraint are at best k-adaptive. So all we know is that the adaptive complexity is between 1 and $\Omega(n)$.

What is the adaptive complexity of maximizing a submodular function?

Adaptivity is not only a fundamental theoretical concept but it also has important practical consequences. There is a wide variety of applications of submodular maximization where function evaluations are easily parallelized but each evaluation requires a long time to complete. In crowdsourcing for example, function evaluations depend on responses from human agents and highly sequential algorithms are impractical. Data summarization, experimental design, influence maximization, marketing, survey design, and biological simulations are all examples where the adaptive complexity of optimization largely determines the runtime bottleneck of the optimization algorithm (see Appendix B for a detailed discussion of these applications).

1.3 Main result

Our main result is that the adaptive complexity of submodular maximization is $\tilde{\Theta}(\log n)$. This provides a characterization that is tight up to low-order terms, and an exponential improvement in the adaptivity over any known constant factor approximation algorithm for maximizing a monotone submodular function. Our characterization is composed of two major results. The first is an algorithm whose adaptivity is $\mathcal{O}(\log n)$ and obtains an approximation arbitrarily close to $1/3$.

Theorem. *For the problem of monotone submodular maximization under a cardinality constraint and any constant $\epsilon > 0$, there exists an $\mathcal{O}(\log n)$-adaptive algorithm which obtains, with probability $1 - o(1)$, a $(1/3 - \epsilon)$-approximation.*

We complement the upper bound by showing that the adaptive complexity of submodular maximization is *at least* quasi-logarithmic by showing that no $\tilde{\Omega}(\log n)$-adaptive algorithm can obtain an approximation strictly better than $\frac{1}{\log n}$.

Theorem. *For the problem of monotone submodular maximization under a cardinality constraint, there is no $\left(\frac{\log n}{12 \log\log n}\right)$-adaptive algorithm that obtains, with probability $\omega\left(\frac{1}{n}\right)$, a $\frac{1}{\log n}$-approximation.*

In fact, we show the following more general impossibility result: for any $r \leq \log n$, there is no r-adaptive algorithm that obtains, with probability $\omega\left(\frac{1}{n}\right)$, an $n^{-\frac{1}{2r+2}} \cdot (r+3) \log^2 n$ approximation.

	Best known adaptivity		Adaptivity in this paper
upper bound	$\Omega(n)$	[NWF78]	$\mathcal{O}(\log n)$
lower bound	1	[BRS17]	$\tilde{\Omega}(\log n)$

Table 1: A comparison of results for the number of rounds required to obtain a constant factor approximation.

1.4 Adaptive sampling: a coupling of learning and optimization

Our motivation is to understand what are the necessary and sufficient conditions from a learnability model that yield desirable approximation guarantees for optimization. Since sharp impossibility results arise from learning with non-adaptive samples over any distribution, we turned to an *adaptive sampling* model [Tho90]. In adaptive sampling, the learner obtains poly(n) samples drawn from a distribution of her choice in every round. Our $(1/3 - \epsilon)$-approximation $\mathcal{O}(\log n)$-adaptive algorithm is achieved by adaptive sampling. Our hardness result holds for queries and hence also for adaptive sampling. This implies that in the realizable case, where there is a true underlying function generating the data, $\tilde{\Theta}(\log n)$ batches of adaptive samples are *necessary and sufficient* to approximately "learn to optimize" a monotone submodular function under a cardinality constraint.

1.5 Technical overview

The algorithm. The main building block of the adaptive sampling algorithm is the construction, at every round r, of a meaningful distribution $\mathcal{D}_r$ with elements having marginal probabilities of being drawn $p_1, \ldots, p_n$. We begin by presenting two simple primitives: *down-sampling* (Section 2.1) and *up-sampling* (Section 2.2). In every round, down-sampling identifies elements $a_i \in N$ whose expected marginal contribution to a random set drawn according to $\mathcal{D}_r$ is sufficiently low and sets $p_i = 0$ for all future rounds. This approach achieves logarithmic adaptivity, but its approximation guarantee is $1/\log n$. The second approach, up-sampling, sets $p_i = 1$, for all future rounds, for all elements in the sample with highest value at that round. It achieves a constant approximation but at the cost of a linear adaptivity. Our main algorithm, Adaptive-Sampling (Section 2.3) achieves logarithmic adaptivity and constant factor approximation by shaping $\mathcal{D}_r$ via up-sampling at rounds where a random set has high value and down-sampling otherwise. The analysis then heavily exploits submodularity in non-trivial ways to bound the marginal contribution of elements to a random set drawn from $\mathcal{D}_r$, which evolves in every round.

Hardness. To bound the number of rounds necessary to obtain a certain approximation guarantee, we analyze the information that an algorithm can learn in one round that may depend on queries from previous rounds. Reasoning about these dependencies between rounds is the main challenge. To do so, we reduce the problem of finding an r-adaptive algorithm to the problem of finding an $r+1$-adaptive algorithm over a family of functions with additional information. This approach is related to the round elimination technique used in communication complexity (e.g. [MNSW95]).

1.6 Paper organization

We begin by presenting the algorithm and its analysis in Section 2. Section 3 is devoted to the hardness result. Adaptivity in CS is discussed in Appendix A.1 and comparisons with the Map-Reduce and PRAM models are in Appendix A.2. Finally, applications of adaptivity are in Appendix B.

2 The Adaptive Complexity of Submodular Maximization is $\mathcal{O}(\log n)$

In this section, we show that the adaptive complexity of maximizing a monotone submodular function under a cardinality constraint is $\mathcal{O}(\log n)$ via the ADAPTIVE-SAMPLING algorithm, which has logarithmic adaptivity and obtains an approximation arbitrarily close to $1/3$. This algorithm uses two simple, yet powerful, adaptive sampling techniques as primitives. The first is *down-sampling* which in each round maintains a uniform distribution over high-valued elements by iteratively discarding elements with low marginal contribution to a random set. The second primitive is *up-sampling* which at every round identifies the elements with highest value and includes them in all future samples. Neither of these primitives achieves a constant factor approximation in $\mathcal{O}(\log n)$ rounds, but an appropriate combination of them does.

2.1 Down-sampling

The down-sampling algorithm is $O(\log n)$-adaptive but its approximation guarantee is $\Omega(\frac{1}{\log n})$. We describe the algorithm and analyze its properties which will later be used in the analysis of ADAPTIVE-SAMPLING. In every round, as long as the expected value of a random subset of size k of the surviving elements is not an α-approximation of the value of the optimal solution $\texttt{OPT}$, the down-sampling algorithm discards all elements whose expected marginal contribution to a random set is below a fixed threshold Δ. A formal description is included below.

Algorithm 1 DOWN-SAMPLING, discards a large number of elements at every round by sampling.

Input: approximation α and threshold parameter Δ
  Initialize $S \leftarrow N$, $\mathcal{D}$ as the uniform distribution over sets of size k
  while $|S| > k$ and $\mathbb{E}_{R\sim\mathcal{D}}[f(R)] < \alpha\texttt{OPT}$ **do**
    $S \leftarrow S \setminus \left\{a \ : \ \mathbb{E}_{R\sim\mathcal{D}}\left[f_{R\setminus\{a\}}(a)\right] < \Delta\right\}$
    Update $\mathcal{D}$ to be uniform over subsets of S of size k
  return $R \sim \mathcal{D}$

Algorithm 1 is an idealized description of the down-sampling algorithm. In practice, we cannot evaluate the exact expected value of a random set and we do not know $\texttt{OPT}$. Instead, we sample random sets from $\mathcal{D}$ at every round to estimate the expectations and guess $\texttt{OPT}$. For ease of notation and presentation, we analyze this idealized version of the algorithm, discuss the extension to the full algorithm in Section 2.4, and formally describe the full algorithm in Appendix C. This idealized version also has a nice interpretation via the *multi-linear extension* of submodular functions as a search of a continuous point $\mathbf{x} \in [0,1]^n$ which, at every iteration, is projected to a lower dimension on the boundary of the polytope of feasible points (see Appendix D.1 for details).

Analysis of down-sampling. The analysis of down-sampling largely relies on Lemma 1 and Lemma 2, which respectively bound the number of elements discarded at every round and the loss in the approximation due to these discarded elements. We discuss these lemmas in the following subsections. Recall that a function $f : 2^N \to \mathbb{R}^+$ is *submodular* if for every $S \subseteq T \subseteq N$ and $a \notin T$ we have that $f_S(a) \geq f_T(a)$, where $f_A(b)$ denotes the marginal contribution $f_A(b) = f(A \cup \{b\}) - f(A)$ of $b \in N$ to $A \subseteq N$. Such a function is *monotone* if $f(S) \leq f(T)$ for all $S \subseteq T$. Finally, it is *subadditive* if $f(A \cup B) \leq f(A) + f(B)$ for all $A, B \subseteq N$, which is satisfied by submodular functions.

2.1.1 The adaptivity of down-sampling

One crucial property of the down-sampling algorithm is that it is $\mathcal{O}(\log n)$-adaptive. This is largely due to the fact that in every round a significant fraction of the remaining elements are discarded. Throughout the paper we use $\mathcal{U}(S,t)$ to denote the uniform distribution over subsets of S of size t.

Lemma 1. *Let $f : 2^N \to \mathbb{R}$ be a monotone submodular function. For all $S \subseteq N$, $t \in [n]$, and $\Delta > 0$, let $\mathcal{D} = \mathcal{U}(S,t)$ and the discarded elements be $S^- = \left\{a : \mathbb{E}_{R\sim\mathcal{D}}\left[f_{R\setminus\{a\}}(a)\right] < \Delta\right\}$. Then:*

$$\left|S \setminus S^-\right| \leq \frac{\mathbb{E}_{R\sim\mathcal{D}}[f(R)]}{t\cdot\Delta}\cdot|S|.$$

Proof Sketch (full proof in Appendix D.2.2). We first lower bound the expected value of a random set $\mathbb{E}_{R\sim\mathcal{D}}[f(R)]$ by the sum of the expected marginal contribution of the remaining elements $\sum_{a\in S\setminus S^-}\Pr[a\in R]\cdot\mathbb{E}_{R\sim\mathcal{D}}\left[f_{R\setminus\{a\}}(a)\right]$, using submodularity. Then, we use the fact that the expected marginal contribution of surviving elements is at least Δ to obtain the desired bound. □

Notice that when $\Delta = c\cdot\frac{\alpha\texttt{OPT}}{k}$ for some $c > 1$, the lemma implies that if $\mathbb{E}_{R\sim\mathcal{D}}[f(R)] < \alpha\texttt{OPT}$, then the number of elements remaining is reduced by a factor of at least c at every round.

2.1.2 The approximation guarantee of down-sampling

The down-sampling algorithm is an $\Omega(\frac{1}{\log n})$ approximation (Corollary 1). To analyze the value of sets that survive down-sampling, we show that the value $f(O\cap S^-)$ of discarded optimal elements is small. Thus, the optimal elements that are not discarded conserve a large fraction of $\texttt{OPT}$.

Lemma 2. *Let $f : 2^N \to \mathbb{R}$ be monotone submodular with optimal solution O and $\mathcal{D} = \mathcal{U}(S,t)$, for any $S \subseteq N$ and $t \in [n]$. The loss from discarding elements $S^- := \left\{a \in S : \mathbb{E}_{R\sim D}\left[f_{R\setminus\{a\}}(a)\right] < \Delta\right\}$ is approximately bounded by the value of $R \sim \mathcal{D}$:*

$$f(O\cap S^-) \leq |O\cap S^-|\Delta + \mathop{\mathbb{E}}_{R\sim\mathcal{D}}\left[f(R)\right].$$

Proof. The value of $O\cap S^-$ is upper bounded using the threshold Δ for elements to be in S^-,

$$f(O\cap S^-) - \mathbb{E}\left[f(R)\right] \leq \mathbb{E}\left[f_R(O\cap S^-)\right] \leq \mathbb{E}\left[\sum_{a\in O\cap S^-} f_R(a)\right] \leq \sum_{a\in O\cap S^-}\mathbb{E}\left[f_R(a)\right] \leq |O\cap S^-|\cdot\Delta$$

where the first inequality is by monotonicity, the second by submodularity, the third by linearity of expectation, and the last by submodularity and definition of S^- □

At this point, we can prove the following corollary about the down-sampling algorithm.

Corollary 1. Down-Sampling *with $\Delta = \frac{\texttt{OPT}}{4k}$ and $\alpha = \frac{1}{\log n}$ is $\mathcal{O}(\frac{\log n}{\log\log n})$-adaptive and obtains, in expectation, a $\left(\frac{1}{\log n}\right)$-approximation.*

Proof Sketch (full proof in Appendix D.2.2). The adaptivity follows from the fact that the number of remaining elements is reduced by a $\log n$ factor at every round by Lemma 1. Next, for the approximation guarantee, we first bound the value of remaining elements S by $\texttt{OPT} - f\left(O\cap\left(\cup_{i=1}^r S_i^-\right)\right)$, where S_i^- is the set of discarded elements at round i, by monotonicity and subadditivity. Then, we bound $f(O\cap S_i^-)$ using Lemma 2 and obtain the approximation guarantee. □

It is important to note that $\Omega(\frac{1}{\log n})$ is the best approximation the down-sampling algorithm can achieve, regardless of the number of rounds. There is a delicate tradeoff between the approximation obtained when the algorithm terminates due to $\mathbb{E}[f(R)] \geq \alpha\texttt{OPT}$ and the one when $|S| \leq k$, s.t. more rounds do not improve the approximation guarantee. We further discuss this in Appendix D.2.1.

2.2 Up-sampling

A second component of the main algorithm is *up-sampling*. Instead of discarding elements, the up-sampling algorithm adds elements which are included in all future samples. At each round, the sample containing the k/r new elements with highest value is added to the current solution X.

Algorithm 2 UP-SAMPLING, adds a large number of elements at every round by sampling.

Input: Sample complexity m and number of rounds r
  Initialize $X \leftarrow \emptyset$
  for r rounds **do**
    Update $\mathcal{D}$ to be uniform over subsets of $N \setminus X$ of size k/r
    $X \leftarrow X \cup \text{argmax}_{R_i}\{f(X \cup R_i) : R_i \sim \mathcal{D}\}_{i=1}^{m}$
  return X

Note that when $r = k$ this method is the celebrated greedy algorithm. In contrast to down-sampling, which obtains a logarithmic number of rounds and approximation, up-sampling is inherently sequential and only obtains an $O(r/k)$ approximation. The proof is deferred to Appendix D.3.

Proposition 2. *For any constant $c \leq k/r$,* UP-SAMPLING *is an r-adaptive algorithm and obtains, w.p. $1-o(1)$, a $\left(1-\frac{1}{e}\right)\frac{c \cdot r}{k+c}$ approximation, with sample complexity $m = cn^{2+c}\log n$ at every round.*

2.3 Adaptive-sampling: $\mathcal{O}(\log n)$-adaptivity and constant factor approximation

We build upon down and up-sampling to obtain the main algorithm, ADAPTIVE-SAMPLING. The algorithm maintains two sets, S for down-sampling and X for up-sampling. If a random subset has high expected value, then a sample of high value is added to the up-sampling set X. Otherwise, low-value elements can be discarded from the down-sampling solution S. A crucial subtlety is that this algorithm samples sets of size k/r not only for up-sampling but also for down-sampling (rather than k). The description below is an idealized version of the algorithm.

Algorithm 3 ADAPTIVE-SAMPLING: down-samples or up-samples depending on context.

Input: approximation α, threshold Δ, sample complexity m, bound on up-sampling rounds r
  Initialize $X \leftarrow \emptyset, S \leftarrow N$
  while $|X| < k$ **and** $|X \cup S| > k$ **do**
    Update $\mathcal{D}$ to be uniform over subsets of $S \setminus X$ of size k/r
    if $\mathbb{E}_{R\sim\mathcal{D}}\left[f_X(R)\right] \geq (\alpha/r)\texttt{OPT}$ **then**
      $X \leftarrow X \cup \text{argmax}_{R_i}\{f(X \cup R_i) : R_i \sim \mathcal{D}\}_{i=1}^{m}$
    else
      $S \leftarrow S \setminus \left\{a \ : \ \mathbb{E}_{R\sim\mathcal{D}}\left[f_{X\cup R\setminus\{a\}}(a)\right] < \Delta\right\}$
  return X if $|X| = k$, or $X \cup S$ otherwise

2.3.1 The adaptivity of Adaptive-Sampling is $\mathcal{O}(\log n)$

The adaptivity of Adaptive-Sampling is the sum of the number of up-sampling rounds and of the number of down-sampling rounds, which we denote by r_u and r_d respectively.

Lemma 3. Adaptive-Sampling *is* $(r_u + r_d)$*-adaptive with* $r_u + r_d \leq r + \log_c n$ *when* $\Delta = c \cdot \frac{\alpha \mathtt{OPT}}{k}$.

Proof Sketch (full proof in Appendix D.4). The number of up-sampling rounds r_u is bounded by r since there are k/r elements added to X at every such round. The number of down-sampling rounds r_d is bounded similarly as for the down-sampling algorithm, using Lemma 1 with the function $f_X(\cdot)$ and the fact that $\mathbb{E}_{R\sim\mathcal{D}}[f_X(R)]$ has low value at a down-sampling round. □

2.3.2 Adaptive-Sampling is a constant factor approximation

We now analyze the approximation guarantee of Adaptive-Sampling.

Lemma 4. Adaptive-Sampling *obtains, w.p.* $1-\delta$, *a* $\left(\frac{1}{3}-\epsilon\right)$*-approximation and has sample complexity* $m = \left(\frac{r}{\epsilon}\right)^2 \log\left(\frac{2r}{\delta}\right)$ *at every round, with* $\alpha = \frac{1}{3}$, $r = \frac{3}{\epsilon} \cdot \log_{1+\epsilon/2} n$, $\Delta = \left(1+\frac{\epsilon}{2}\right)\frac{\alpha \mathtt{OPT}}{k}$.

Proof. We begin with the case where the algorithm returns $S \cup X$, which is the main component of the proof, and then we consider the case where it returns X. At a high level, the first part in analyzing $S \cup X$ consists of bounding $f(S \cup X)$ in terms of the loss from optimal elements $f(O \setminus S)$. Then, we use Lemma 2 to bound the loss from these elements at every round. A main theme of this proof is that we need to simultaneously deal with the up-sampling solution X while analyzing the loss from $O \setminus S$.

Let $O = \{o_1, \ldots, o_k\}$ be the optimal solution indexed by an arbitrary order and X_i and S_i^- be the sets X and $S^- = \{a \ : \ \mathbb{E}_{R\sim\mathcal{D}}\left[f_{X\cup R\setminus\{a\}}(a)\right] < \Delta\}$ at the ith round of down-sampling, $i \in [r_d]$. First, by monotonicity, subadditivity, and again monotonicity, we get

$$f(S \cup X) \geq f(O) - f(O \setminus (S \cup X)) \geq \mathtt{OPT} - f(O \setminus S).$$

The remaining of the proof bounds the loss $f(O \setminus S)$ from optimal elements that were discarded from S. Next, we bound $f(O \setminus S)$. The elements in $O \setminus S$ are elements that have been discarded from S, so $O \setminus S = \cup_{i=1}^{r_d}\left(S_i^- \cap O\right)$, and we get

$$f(O \setminus S)) = f\left(\cup_{i=1}^{r_d}\left(S_i^- \cap O\right)\right) \leq f_X\left(\cup_{i=1}^{r_d}\left(S_i^- \cap O\right)\right) + f(X) \leq \sum_{i=1}^{r_d} f_X\left(O \cap S_i^-\right) + f(S \cup X).$$

where the first inequality is by monotonicity and the second by subadditivity. Next,

$$f_X\left(O \cap S_i^-\right) \leq f_{X_i}\left(O \cap S_i^-\right) \leq |O \cap S_i^-| \cdot \Delta + \mathbb{E}\left[f_{X_i}(R)\right].$$

where the first inequality is by submodularity and the second is by Lemma 2 and since $f_{X_i}(\cdot)$ is a submodular function. Thus,

$$\begin{aligned}
f(O \setminus S) - f(S \cup X) &\leq \sum_{i=1}^{r_d}\left(|O \cap S_i^-| \cdot \Delta + \mathbb{E}\left[f_{X_i}(R)\right]\right) \\
&\leq |O \cap (\cup_{i=1}^{r_d} S_i^-)| \cdot \Delta + \left(\alpha \cdot \frac{r_d}{r}\right)\mathtt{OPT} \\
&\leq k \cdot \Delta + \left(\alpha \cdot \frac{r_d}{r}\right)\mathtt{OPT} \\
&\leq \left(1+\frac{\epsilon}{2}\right)\alpha\mathtt{OPT} + \left(\alpha \cdot \frac{r_d}{r}\right)\mathtt{OPT}
\end{aligned}$$

where $\mathbb{E}\left[f_{X_i}(R)\right] \leq (\alpha/r)\texttt{OPT}$ at a downsampling round i by the algorithm. By combining the previous inequalities, we get

$$f\left(S \cup X\right) \geq \texttt{OPT} - \left(1 + \frac{\epsilon}{2}\right)\alpha\texttt{OPT} - f(S \cup X) - \frac{r_d}{r}\alpha\texttt{OPT} \geq \left(\frac{1}{3} - \epsilon\right)\texttt{OPT}$$

where $r_d \leq \log_{1+\epsilon/2} n$ by Lemma 3 with $c = 1 + \epsilon/2$ and since $r = \frac{3}{\epsilon} \cdot \log_{1+\epsilon/2} n$.

What remains is the case where the algorithm returns X. Let X_i and R_i^+ be the set X and the sample R added to X at the ith round of up-sampling, $i \in [r]$. By standard concentration bound (Lemma 13), with $m = (r/\epsilon)^2 \log(2r/\delta)$, w.p. $1 - \delta/r$, $f_{X_i}\left(R_i^+\right) \geq \mathbb{E}_{R\sim\mathcal{D}}\left[f_{X_i}(R)\right] - \epsilon\texttt{OPT}/r$. By a union bound this holds for all r rounds of up-sampling with probability $1 - \delta$. We obtain

$$f(X) = \sum_{i=1}^{r} f_{X_i}\left(R_i^+\right) \geq \sum_{i=1}^{r}\left(\mathop{\mathbb{E}}_{R\sim\mathcal{D}}\left[f_{X_i}(R)\right] - \frac{\epsilon\texttt{OPT}}{r}\right) \geq \sum_{i=1}^{r}\frac{\alpha\texttt{OPT}}{r} - \epsilon\texttt{OPT} = \left(\frac{1}{3} - \epsilon\right)\texttt{OPT}. \qquad \square$$

2.4 The full algorithm

We briefly discuss the two missing pieces, estimating the expectations and the assumption that we know $\texttt{OPT}$, both needed to implement the idealized algorithm. To estimate expectations within arbitrarily fine precision $\epsilon > 0$ in one round, we query $m = \text{poly}(n)$ sets $X \cup R_1, \ldots, X \cup R_m$ where $R_1, \ldots, R_m$ are sampled according to $\mathcal{U}(S \setminus X, k/r)$ (Lemma 11 in Appendix C.1 via standard concentration arguments). To guess $\texttt{OPT}$, we pick $\log_{1+\epsilon} n$ different values $v^\star$ as proxies for $\texttt{OPT}$, one of which must be an ϵ multiplicative approximation to $\texttt{OPT}$ (Lemma 12 using submodularity). We then run the algorithm for each of these proxies in parallel, and return the solution with highest value. With these two final pieces, we obtain the main result for this section. We describe the implementable version Adaptive-Sampling-Full formally in Appendix C and analyze it in Appendix D.5.

Theorem 3. *For any $\epsilon, \delta > 0$,* Adaptive-Sampling-Full *is a $\left(\log_{1+\epsilon/3} n \cdot \frac{3}{\epsilon} + 2\right)$-adaptive algorithm that, w.p. $1 - \delta$, obtains a $(\frac{1}{3} - \epsilon)$-approximation, with sample complexity at every round $m = \frac{64}{\epsilon^2}\left(nk^2 \log\left(\frac{2n}{\delta}\right) + \log^2_{1+\epsilon/3} n \cdot \log\left(\frac{2}{\delta}\right) \cdot \frac{1}{\epsilon}\right)$, for maximizing a monotone submodular function under a cardinality constraint, with parameters $r = \frac{3}{\epsilon} \cdot \log_{1+\epsilon/3} n$, $\alpha = \frac{1}{3}$, and $\Delta = (1 + \epsilon)\frac{v^\star}{3k}$.*

3 The Adaptive Complexity of Submodular Maximization is $\tilde{\Omega}(\log n)$

In this section, we show that the adaptive complexity of maximizing a monotone submodular function under a cardinality constraint is $\tilde{\Omega}(\log n)$ with a hardness result showing that with strictly less than $\tilde{\Omega}(\log n)$ rounds, the best approximation possible is $\frac{1}{\log n}$. With the algorithm from the previous section, we get that the adaptive complexity of submodular maximization is $\log n$, up to lower-order terms, to obtain a constant factor approximation. This hardness result uses an approach related to the round elimination technique used in communication complexity (e.g. [MNSW95]).

3.1 The round elimination lemma

The following simple lemma gives two conditions that, if satisfied by some collections of functions, imply the desired hardness result. The main condition is that an r-adaptive algorithm for a family

of functions $\mathcal{F}_r$ can be modified into an $(r-1)$-adaptive algorithm for a more restricted family $\mathcal{F}_{r-1}$. The base case of this inductive argument is that with no queries, there does not exist any α-approximation algorithm for $\mathcal{F}_0$. A similar round elimination technique is used in communication complexity to characterize the tradeoff between the number of rounds and the total amount of communication of a protocol. Here, the tradeoff is different and is between the number of rounds and the approximation of an algorithm.

Lemma 5. *Assume $r \in \text{poly}(n)$. If there exist families of functions $\mathcal{F}_0, \dots, \mathcal{F}_r$ such that the following two conditions hold:*

- ***Round elimination.*** *For all $i \in \{1, \dots, r\}$, if there exists an i-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_i$, then there exists an $i-1$ adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation algorithm for $\mathcal{F}_{i-1}$;*
- ***Last round.*** *There does not exist a 0-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_0$;*

Then, there is no r-adaptive algorithm that obtains, w.p. $o(1)$, an α-approximation for $\mathcal{F}_r$.

The proof follows immediately by induction on the number of rounds r.

3.2 The onion construction

The main technical challenge is to "fit" $r+1$ families of functions $\mathcal{F}_0, \dots, \mathcal{F}_r$ in the class of submodular functions while also having every family $\mathcal{F}_i$ be significantly richer than $\mathcal{F}_{i-1}$. In our context, significantly richer means that an i-adaptive algorithm for $\mathcal{F}_i$ can be transformed into an $(i-1)$-adaptive algorithm for $\mathcal{F}_{i-1}$. To do so, at a high level, we want to show that after one round of querying a function in $\mathcal{F}_i$, functions in $\mathcal{F}_{i-1}$ are indistinguishable. If functions in $\mathcal{F}_{i-1}$ are indistinguishable to an i-adaptive algorithm after one round of querying, then the last $i-1$ rounds of this algorithm form an $(i-1)$-adaptive algorithm for $\mathcal{F}_{i-1}$.

We construct functions that depend on a partition P of the ground set N into layers $L_0, \dots, L_r, L^\star$ (illustrated in Figure 1). The main motivation behind this layered construction is to create a hard instance s.t. an algorithm cannot distinguish layer L_i from $L^\star$ before round $i+1$. The size of the layers decreases as i grows, with $L^\star$ being the smallest layer. More precisely, we set $|L_i| = n^{1-\frac{i}{r+1}}$ for $i > 0$, $|L^\star| = n^{\frac{1}{2r+2}}$, and L_0 consists of the remaining elements. We define $\ell_i(S) := |L_i \cap S|$ and abuse notation with $\ell_i = \ell_i(S)$ when it is clear from context. The hard function is defined as

$$f^P(S) := \sum_{i=0}^{r} \min(\ell_i, \log^2 n) + \ell^\star + \min\left(\frac{|S|}{8n^{\frac{1}{r+1}}}, 1\right) \cdot \left(2n^{\frac{1}{2r+2}} - \left(\sum_{i=0}^{r} \min(\ell_i, \log^2 n) + \ell^\star\right)\right).$$

To gain some intuition about this function, we note the following two simple facts about f^P:

- If a query is large, i.e. $|S| \geq 8n^{\frac{1}{r+1}}$, then $f^P(S) = 2n^{\frac{1}{2r+2}}$. Informally, all the layers are hidden since no information can be learned about the partition from query S;
- On the other hand, if $|S \cap L_i| \leq \log^2 n$, i.e. $\ell_i \leq \log^2 n$, then elements in L_i and $L^\star$ are indistinguishable to an algorithm that is given the value $f(S)$ since $\min(\ell_i, \log^2 n) = \ell_i$.

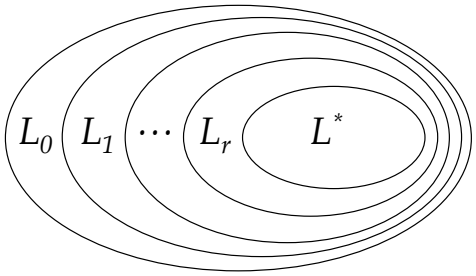

Figure 1: The partition of the elements into layers $L_0, \ldots, L_r, L^\star$ for the hard functions. An algorithm cannot learn L_i before round $i+1$ and $L^\star$ is the optimal solution.

Thus, the queries need to be of size smaller than $8n^{\frac{1}{r+1}}$ while also containing at least $\log^2 n$ elements in L_i for the algorithm to learn some information to distinguish layers L_i and $L^\star$. Since the size of layers diminishes at a rate faster than $\frac{\log^2 n}{8n^{\frac{1}{r+1}}}$, it is hard for the algorithm to distinguish layers L_{i+1} and $L^\star$ if it has not distinguished L_i and $L^\star$ in previous rounds. An interpretation of this construction is that an algorithm can only learn the outmost remaining layer at any round.

3.3 Round elimination for the construction

In order to argue that functions in $\mathcal{F}_{i-1}$ are indistinguishable after one round of querying $f \in \mathcal{F}_i$, we begin by reducing the problem of showing indistinguishability from non-adaptive queries to showing structural properties of a randomized collection of functions $\mathcal{F}_{R_i}$ (the proof is in Appendix E).

Lemma 6. *Let $\mathcal{F}_{R_i}$ be a randomized collection of functions in some $\mathfrak{F}$. Assume that for all $S \subseteq N$, w.p. $1 - n^{-\omega(1)}$ over $\mathcal{F}_{R_i}$, for all $f_1, f_2 \in \mathcal{F}_{R_i}$, we have that $f_1(S) = f_2(S)$. Then, for any (possibly randomized) collection of $\mathrm{poly}(n)$ non-adaptive queries $\mathcal{Q}$, there exists a deterministic collection of functions $\mathcal{F} \in \mathfrak{F}$ such that with probability $1 - n^{-\omega(1)}$ over the randomization of $\mathcal{Q}$, for all queries $S \in \mathcal{Q}$ and all $f_1, f_2 \in \mathcal{F}$, $f_1(S) = f_2(S)$.*

The randomized collection of functions $\mathcal{F}_{R_i}$. For round $r - i$, we define the randomized collection of functions $\mathcal{F}_{R_i} \in \mathfrak{F}_{r-i}$, for some $\mathfrak{F}_{r-i}$, needed for Lemma 6. Informally, the layers $L_0, \ldots, L_{i-1}$ are fixed and the ith layer is a random subset R_i of the remaining elements $N \setminus \cup_{j=0}^{i-1} L_j$. The collection $\mathcal{F}_{R_i}$ is then all functions with layers $L_0, \ldots, L_{i-1}, R_i$. Formally, given $L_0, \ldots, L_{i-1}$, the randomized collection of functions $\mathcal{F}_{R_i}$ is

$$\mathcal{F}_{R_i}(L_0, \ldots, L_{i-1}) := \left\{ f^{(L_0, \ldots, L_{i-1}, R_i, S_{i+1} \ldots S_r, S^\star)} : \sqcup_{j \in \{i+1, \ldots, r, \star\}} S_j = N \setminus \left(\cup_{j=0}^{i-1} L_j \cup R_i \right) \right\}$$

where $R_i \sim \mathcal{U}(N \setminus \cup_{j=0}^{i-1} L_j, n^{1 - \frac{i}{r+1}})$ is a uniformly random subset of size $n^{1-\frac{i}{r+1}}$ of the remaining elements $N \setminus \cup_{j=0}^{i-1} L_j$ and $\sqcup$ denotes the disjoint union of sets. We define $\mathfrak{F}_{r-i}$ to be the collection of all such $\mathcal{F}_{R_i}(L_0, \ldots, L_{i-1})$ over all subsets R_i of the remaining elements $N \setminus \cup_{j=0}^{i-1} L_j$. Next, we show the desired property for the randomized collection of functions $\mathcal{F}_{R_i}$ to apply Lemma 6.

Lemma 7. *Assume $r \leq \log n$. For all $S \subseteq N$ and $i \in [r]$, with probability $1 - n^{-\omega(1)}$ over the randomization of $\mathcal{F}_{R_i} \in \mathfrak{F}_{r-i}$, for all $f_1, f_2 \in \mathcal{F}_{R_i}$, $f_1(S) = f_2(S)$.*

Proof Sketch (full proof in Appendix E). The proof consists of two cases depending on the size of S. If $S \geq 8n^{\frac{1}{r+1}}$, then we immediately conclude that $f(S) = 2n^{\frac{1}{2r+2}}$ for all $f \in \mathcal{F}_{R_i}$. The second and main case is if $S < 8n^{\frac{1}{r+1}}$. We first give a concentration bound showing that for a fixed set S and a random set R, if the expected size of their intersection is constant, then their intersection is of size at most $\log^2 n$ with high probability (Lemma 18). Using this concentration bound, we get that for a fixed S, with high probability, for all $f \in \mathcal{F}_{R_i}$, $\sum_{j=i+1}^{r} \ell_j < \log^2 n$. We conclude by showing that this implies that $f_1(S) = f_2(S)$ for all $f_1, f_2 \in \mathcal{F}_{R_i}$. □

Combining the two previous lemmas, we are ready to show the round elimination condition.

Lemma 8. *Assume $r \leq \log n$. For all $i \in \{1, \ldots, r\}$ and all collection of functions $\mathcal{F}_i \in \mathfrak{F}_i$, if there exists an i-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_i$, then there exists a collection of functions $\mathcal{F}_{i-1} \in \mathfrak{F}_{i-1}$ such that there exists an $(i-1)$-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_{i-1}$.*

Proof. Assume that for some $\mathcal{F}_i \in \mathfrak{F}_i$, there exists an i-adaptive algorithm $\mathcal{A}$ that obtains, w.p. $n^{-\omega(1)}$, an α-approximation . Let $L_0, \ldots, L_{r-i}$ be the layers of all $f^P \in \mathcal{F}_i$ and consider the randomized collection of functions $\mathcal{F}_{R_{r-i+1}}(L_0, \ldots, L_{r-i}) \in \mathfrak{F}_{i-1}$. By combining Lemma 6 and Lemma 7, there exists a deterministic collection of functions $\mathcal{F}_{i-1} \in \mathfrak{F}_{i-1}$ such that, w.p. $1 - n^{-\omega(1)}$ over the randomization of the algorithm $\mathcal{A}$ for the non-adaptive queries $\mathcal{Q}$ in the first round, for all queries $S \in \mathcal{Q}$, $f_1(S) = f_2(S)$ for all $f_1, f_2 \in \mathcal{F}_{i-1}$. Since $\mathcal{F}_{i-1} \subseteq \mathcal{F}_i$, algorithm $\mathcal{A}$ is also an i-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_{i-1}$.

For optimizing $f \in \mathcal{F}_{i-1}$, the decisions of the algorithm are, w.p. $1 - n^{-\omega(1)}$, *independent* of the queries made in the first round, which is the case when for all queries S, $f_1(S) = f_2(S)$ for all $f_1, f_2 \in \mathcal{F}_{i-1}$. Consider the algorithm $\mathcal{A}'$ that consists of the last $i-1$ rounds of algorithm $\mathcal{A}$ when for all queries S by $\mathcal{A}$ in the first round, $f_1(S) = f_2(S)$ for all $f_1, f_2 \in \mathcal{F}_{i-1}$. We get that $\mathcal{A}'$ is an $i-1$ adaptive algorithm that obtains, w.p. $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_{i-1}$. □

It remains to show the last round condition needed for Lemma 5.

Lemma 9. *For all $\mathcal{F} \in \mathfrak{F}_0$, there does not exist a 0-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an $n^{-\frac{1}{2r+2}} \cdot \left((r+2)\log^2 n + 1\right)$ approximation.*

Proof Sketch (full proof in Appendix E). The proof uses a probabilistic argument and considers f picked at random from $\mathcal{F}$. The decisions of a 0-adaptive algorithm are then *independent* of this randomization since there are no queries. Next, by using the same concentration used previously (Lemma 18) we show that with high probability, the solution returned by the algorithm contains a small number of elements in $L^\star$, and thus obtains a low value compared to the optimal $L^\star$. □

Lemma 19 in Appendix E shows that f^P is monotone and submodular. By combining lemmas 5, 8, 9, and 19, we obtain the main result for this section.

Theorem 4. *For any $r \leq \log n$, there is no r-adaptive algorithm for maximizing a monotone submodular function under a cardinality constraint that obtains, with probability $o(1)$, an $n^{-\frac{1}{2r+2}} \cdot \left((r+3)\log^2 n\right)$ approximation. In particular, there is no $\frac{\log n}{12 \log\log n}$-adaptive algorithm that obtains, with probability $\omega\left(\frac{1}{n}\right)$, a $\frac{1}{\log n}$-approximation.*

Acknowledgements

We thank Sergei Vassilvitskii, Avrim Blum, Sepehr Assadi, Vitaly Feldman, and Amin Karbasi for helpful discussions.

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 *Conference on Learning Theory*, 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.

[Bal15] Maria-Florina Balcan. Learning submodular functions with applications to multi-agent systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, 2015.

[BCIW12] Maria-Florina Balcan, Florin Constantin, Satoru Iwata, and Lei Wang. Learning valuation functions. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, 2012.

[BDF+12] Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, 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 *International Conference on Machine Learning*, 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 *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, 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.

[BH11] Maria-Florina Balcan and Nicholas JA Harvey. Learning submodular functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 793–802. ACM, 2011.

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

[BMW16] Mark Braverman, Jieming Mao, and S Matthew Weinberg. Parallel algorithms for select and partition with noisy comparisons. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 851–862. ACM, 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 *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 23–32. ACM, 2011.

[BRM98] Guy E Blelloch and Margaret Reid-Miller. Fast set operations using treaps. In *Proceedings of the tenth annual ACM symposium on Parallel algorithms and architectures*, pages 16–26. ACM, 1998.

[BRS89] Bonnie Berger, John Rompel, and Peter W Shor. Efficient nc algorithms for set cover with applications to learning and geometry. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 54–59. IEEE, 1989.

[BRS17] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The limitations of optimization from samples. *Proceedings of the Forty-Ninth Annual ACM on Symposium on Theory of Computing*, 2017.

[BS17a] Eric Balkanski and Yaron Singer. Minimizing a submodular function from samples. 2017.

[BS17b] Eric Balkanski and Yaron Singer. The sample complexity of optimizing a convex function. In *Conference on Learning Theory*, pages 275–301, 2017.

[BST12] Guy E Blelloch, Harsha Vardhan Simhadri, and Kanat Tangwongsan. Parallel and i/o efficient set covering algorithms. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 82–90. ACM, 2012.

[BVP11] Gregory R Bowman, Vincent A Voelz, and Vijay S Pande. Taming the complexity of protein folding. *Current opinion in structural biology*, 21(1):4–11, 2011.

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

[CKT10] Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-cover in map-reduce. In *Proceedings of the 19th international conference on World wide web*, pages 231–240. ACM, 2010.

[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 *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 81–91. ACM, 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 *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, 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.

[FK14] Vitaly Feldman and Pravesh Kothari. Learning coverage functions and private release of marginals. In *Conference on Learning Theory*, pages 679–702, 2014.

[FMV11] Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[FV13] Vitaly Feldman and Jan Vondrák. Optimal bounds on approximation of submodular and XOS functions by juntas. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, 2013.

[FV15] Vitaly Feldman and Jan Vondrák. Tight bounds on low-degree spectral concentration of submodular and xos functions. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 923–942. IEEE, 2015.

[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, 2009. DSP/SPE 2009. IEEE 13th*, 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 *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 285–294. IEEE, 2011.

[JBS13] Stefanie Jegelka, Francis Bach, and Suvrit Sra. Reflection methods for user-friendly submodular optimization. In *Advances in Neural Information Processing Systems*, pages 1313–1321, 2013.

[JLB11] Stefanie Jegelka, Hui Lin, and Jeff A Bilmes. On fast approximate submodular minimization. In *Advances in Neural Information Processing Systems*, pages 460–468, 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 *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. Society for Industrial and Applied Mathematics, 2010.

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

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

[MNSW95] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, 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 *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 153–162. ACM, 2015.

[NJJ14] Robert Nishihara, Stefanie Jegelka, and Michael I Jordan. On the convergence rate of decomposable submodular function minimization. In *Advances in Neural Information Processing Systems*, pages 640–648, 2014.

[NPS15] Harikrishna Narasimhan, David C. Parkes, and Yaron Singer. Learnability of influence in networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3186–3194, 2015.

[NW91] Noam Nisan and Avi Widgerson. Rounds in communication complexity revisited. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 419–429. ACM, 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.

[Tho90] Steven K Thompson. Adaptive cluster sampling. *Journal of the American Statistical Association*, 85(412):1050–1059, 1990.

[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.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[WIB14] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *International Conference on Machine Learning*, pages 1494–1502, 2014.

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

In this section, we discuss two related models, the Map-Reduce model for distributed computation and the PRAM model. These models are compared to the notion of adaptivity in the context of submodular optimization.

A.2.1 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.

A.2.2 PRAM

In the PRAM model, the notion of depth is closely related to the concept of adaptivity studied in this paper. Our positive result extends to the PRAM model, showing that 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 $1/3$ for maximizing any monotone submodular function under a cardinality constraint, where d_f is the depth required to evaluate the function on a set.

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 number of parallel steps of this algorithm on the PRAM, in other words, it is the longest chain of dependencies of the algorithm, including operations which are not necessarily queries. The problem of designing low-depth algorithms has been heavily studied (e.g. [Ble96, BPT11, BRS89, RV98, BRM98, BST12]).

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. However, for the applications we consider, the runtime of the algorithmic computations which are not queries are usually insignificant compared to the time to evaluate a query. In addition, the PRAM model assumes that the input is loaded in memory while we consider the value query model where the algorithm is given oracle access to a function of potentially exponential size. In crowdsourcing applications, for example, where the value of a set can be queried on a crowdsourcing platform, there does not necessarily exist a succinct representation of the underlying function.

Our positive results extend with an additional $d_f \cdot \tilde{O}(\log n)$ factor in the depth compared to the number of adaptive rounds, 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].

More broadly, there has been a recent interest in machine learning to scale submodular optimization algorithms for applications over large datasets [JLB11, JBS13, WIB14, NJJ14, PJG+14].

B Applications

Beyond being a fundamental concept, adaptivity is important for applications where sequentiality is the main runtime bottleneck.

Crowdsourcing and data summarization. One class of such problems where adaptivity plays an important role are human-in-the-loop problems. At a high level, these 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 impractical. A crowdsourcing platform consists of posted tasks and crowdworkers who are remunerated for performing these posted tasks. For the submodular problem of data summarization, where the objective is to select a small representative subset of a dataset, the quality of subsets as representatives 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 send out a large number of tasks to be performed simultaneously by different crowdworkers.

Biological simulations. Adaptivity is also studied in molecular biology to simulate protein folding. Adaptive sampling techniques are used to obtain significant improvements in execution of simulations and discovery of low energy states [BVP11].

Experimental Design. 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].

Influence Maximization. The submodular problem of influence maximization, initiated studied by [DR01, RD02, KKT03] has since then received considerable attention (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 a waiting time occurs when observing the total number of nodes influenced by some seed set of nodes.

Advertising. 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 behavior of customers.

C The Full Algorithm

C.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 Down-Sampling and Adaptive-Sampling algorithms. Recall that $\mathcal{U}(S,t)$ denotes the uniform distribution over subsets of S of size t. The values we are interested in estimating are $\mathbb{E}_{R\sim\mathcal{U}(S,t)}\left[f_X(R)\right]$ and $\mathbb{E}_{R\sim\mathcal{U}(S,t)}\left[f_{X\cup R\setminus\{a\}}(a)\right]$. We denote the corresponding estimates by $v_X(S,t)$ and $v_X(S,t,a)$, which are computed in Algorithms 4 and 5. These algorithms first sample m sets from $\mathcal{U}(S,t)$, where m is the sample complexity, then query the desired sets to obtain a random realization of $f_X(R)$ and $f_{X\cup R\setminus\{a\}}(a)$, and finally averages the m random realizations of these values.

Algorithm 4 Estimate: computes estimate $v_X(S,t)$ of $\mathbb{E}_{R\sim\mathcal{U}(S,t)}\left[f_X(R)\right]$.

Input: set $S\subseteq N$, size $t\in[n]$, sample complexity m.
Sample $R_1,\ldots,R_m \overset{\text{i.i.d.}}{\sim} \mathcal{U}(S,t)$
Query $\{X, X\cup R_1,\ldots,X\cup R_m\}$
$v_X(S,t)\leftarrow \frac{1}{m}\sum_{i=1}^{m} f(X\cup R_i)-f(X)$
return $v_X(S,t)$

Algorithm 5 Estimate2: Computes estimate $v_X(S,t,a)$ of $\mathbb{E}_{R\sim\mathcal{U}(S,t)}\left[f_{X\cup R\setminus\{a\}}(a)\right]$.

Input: set $S\subseteq N$, size $t\in[n]$, sample complexity m, element $a\in N$.
Sample $R_1,\ldots,R_m \overset{\text{i.i.d.}}{\sim} \mathcal{U}(S,t)$
Query $\{X\cup R_1\cup\{a\}, X\cup R_1\setminus\{a\},\ldots,X\cup R_m\cup\{a\}, X\cup R_m\setminus\{a\}\}$
$v_X(S,t,a)\leftarrow \frac{1}{m}\sum_{i=1}^{m} f(X\cup R_i\cup\{a\})-f(X\cup R_i\setminus\{a\})$
return $v_X(S,t,a)$

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 10 (Hoeffding's inequality)**.** *Let $X_1,\ldots,X_n$ be independent random variables with values in $[0,b]$. Let $X=\frac{1}{m}\sum_{i=1}^{m}X_i$. Then for any $\epsilon>0$,*

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

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

Lemma 11. *Let $m=\frac{1}{2}\left(\frac{OPT}{\epsilon}\right)^2\log\left(\frac{2}{\delta}\right)$, then for all $X,S\subseteq N$ and $t\in[n]$ such that $|X|+t\leq k$, with probability at least $1-\delta$ over the samples $R_1,\ldots,R_m$,*

$$\left|v_X(S,t)-\mathop{\mathbb{E}}_{R\sim\mathcal{U}(S,t)}\left[f_X(R)\right]\right|\leq\epsilon$$

Similarly, let $m = \frac{1}{2}\left(\frac{\mathtt{OPT}}{\epsilon}\right)^2 \log\left(\frac{2}{\delta}\right)$*, then for all* $X, S \subseteq N$*,* $t \in [n]$*, and* $a \in N$ *such that* $|X| + t \leq k$*, with probability at least* $1 - \delta$ *over the samples* $R_1, \ldots, R_m$*,*

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

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

Proof. Note that

$$\mathbb{E}\left[v_X(S,t)\right] = \mathop{\mathbb{E}}_{R \sim \mathcal{U}(S,t)} \left[f_X(R)\right] \qquad \text{and} \qquad \mathbb{E}\left[v_X(S,t,a)\right] = \mathop{\mathbb{E}}_{R \sim \mathcal{U}(S,t)} \left[f_{X \cup R \setminus \{a\}}(a)\right]$$

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

$$\Pr\left[\left| v_X(S,t) - \mathop{\mathbb{E}}_{R \sim \mathcal{U}(S,t)} \left[f_X(R)\right] \right| \geq \epsilon\right] \leq 2e^{-\frac{2m\epsilon^2}{\mathtt{OPT}^2}} \leq \delta$$

for $\epsilon > 0$. Similarly, we get

$$\Pr\left[\left| v_X(S,t,a) - \mathop{\mathbb{E}}_{R \sim \mathcal{U}(S,t)} \left[f_{X \cup R \setminus \{a\}}(a)\right] \right| \geq \epsilon\right] \leq \delta.$$

Thus, with $m = n\left(\frac{\mathtt{OPT}}{\epsilon}\right)^2 \log\left(\frac{2n}{\delta}\right)$ 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 $1 - \delta$. □

We can now describe the (almost) full version of the main algorithm which uses these estimates. One additional small difference with Adaptive-Sampling is that we force the algorithm to stop after r^+ rounds to obtain the adaptive complexity with probability 1. The loss from the event, happening with low probability, that the algorithm is forced to stop is accounted for in the δ probability of failure of the approximation guarantee of the algorithm.

Algorithm 6 Adaptive-Sampling-Proxy, simultaneously downsamples and upsamples.

Input: bounds on up-sampling rounds r and on total rounds r^+, approximation α, threshold parameter Δ, sample complexity m, and proxy $v^\star$

```
Initialize X ← ∅, S ← N, t ← k/r, c ← 1
while |X| < k, |S ∪ X| > k, and c < r^+ do                    Adaptive loop
    v_X(S,t) ← Estimate(S,t,m)
    if v_X(S,t) ≥ (α/r) · v^⋆ then
        X ← X ∪ argmax{f(X ∪ R_i) : R_i ∼ U(S,t)}_{i=1}^m
        S ← S \ X
    else
        for a ∈ S do                                          Non-adaptive loop
            v_X(S,t,a) ← Estimate2(S \ X,t,m,a)
        S ← S \ {a : v_X(S,t,a) ≤ Δ}
    c ← c + 1
return X if |X| = k, or S ∪ X otherwise
```

C.2 Estimating OPT

The main idea to estimate OPT is to have $O(\log n)$ values v_i such that one of them is guaranteed to be a $(1-\epsilon)$-approximation to 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 OPT.

Lemma 12. *Let* $a^\star = \operatorname{argmax}_{a \in N} f(\{a\})$ *be the optimal singleton, and*

$$v_i = (1+\epsilon)^i \cdot f\left(\{a^\star\}\right).$$

Then, there exists some $i \in \left[\frac{\log n}{\log(1+\epsilon)}\right]$ *such that*

$$\mathit{OPT} \leq v_i \leq (1+\epsilon) \cdot \mathit{OPT}.$$

Proof. 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 $v_0 \leq \texttt{OPT} \leq v_{\frac{\log n}{\log(1+\epsilon)}}$. By the definition of v_i, we then conclude that there must exists some $i \in \left[\frac{\log n}{\log(1+\epsilon)}\right]$ such that $\texttt{OPT} \leq v_i \leq (1+\epsilon) \cdot \texttt{OPT}$. □

Since the solution obtained for the unknown v_i which approximates 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 Adaptive-Sampling-Full which we describe next.

Algorithm 7 Adaptive-Sampling-Full, simultaneously downsamples and upsamples.

Input: bounds on up-sampling rounds r and on total rounds r^+, approximation α, threshold parameter Δ, sample complexity m, and precision ϵ

Initialize $L \leftarrow \emptyset$

Query $\{\{a_1\}, \ldots, \{a_n\}\}$

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

for $i \in \left\{0, \ldots, \log_{1+\epsilon/3} n\right\}$ **do** Non-adaptive loop

  $v^\star \leftarrow (1+\epsilon)^i \cdot f\left(\{a^\star\}\right)$

  Add solution from Adaptive-Sampling-Proxy($v^\star$) to L

return $\operatorname{argmax}_{S \in L} f(S)$

D Missing Discussion and Analysis from Section 2

D.1 Continuous interpretation of algorithm via the multilinear extension

The Down-Sampling algorithm also has a simple description using the multilinear extension F of f.

Algorithm 8 DownSamplingContinuous, a continuous description of Down-Sampling.

Input: approximation α and precision ϵ
$\mathbf{x} \leftarrow (\frac{k}{n}, \ldots, \frac{k}{n})$,
while $F(\mathbf{x}) < \alpha\texttt{OPT}$ **do**
$\mathcal{M} \leftarrow \left\{\mathbf{v} : \|\mathbf{v}\|_0 \leq (1-\epsilon)\|\mathbf{x}\|_0, \|\mathbf{v}\|_1 = k, \mathbf{v} \leq \frac{1}{1-\epsilon}\mathbf{x}\right\}$
$\mathbf{x} \leftarrow \text{argmax}_{\mathbf{v}\in\mathcal{M}}\langle \mathbf{v}, \nabla F(\mathbf{x})\rangle$
return $\mathbf{x}$

The multilinear extension $F : [0,1]^n \to \mathbb{R}$ of a submodular function f is a popular tool for continuous approaches to submodular optimization, where $F(\mathbf{x})$ is the expected value $\mathbb{E}[f(R)]$ of a random set R containing each element a_i independently with probability x_i. An interpretation of this algorithm is a continuous point $\mathbf{x}$ which, at every iteration, is projected to a lower dimension ($\|\mathbf{v}\|_0 \leq (1-\epsilon)\|\mathbf{x}\|_0$) among the remaining dimensions ($\mathbf{v} \leq \frac{1}{1-\epsilon}\mathbf{x}$) on the boundary of the polytope of feasible points ($\|\mathbf{v}\|_1 = k$).

D.2 Missing analysis from Section 2.1

D.2.1 Tradeoff with down-sampling

We first discuss the tradeoff between the approximation obtained when the algorithm terminates due to $\mathbb{E}[f(R)] \geq \alpha\texttt{OPT}$ and the one when $|S| \leq k$. We argue that this tradeoff implies that more rounds do not improve the approximation guarantee for down-sampling.

Notice that when the threshold $\alpha\texttt{OPT}$ to return R increases, then the threshold $\Delta = c_\Delta \cdot \frac{\alpha\texttt{OPT}}{k}$ needed to remove elements also increases. If this threshold to remove elements increase, then the algorithm potentially discards optimal elements with higher value. Thus, the apporoximation guarantee obtained by the solution S worsens. This tradeoff is independent of the number of rounds and adding more rounds thus does not improve the approximation guarantee.

D.2.2 Analysis of down-sampling

We bound the number of elements removed from S in each round.

Lemma 1. *Let $f : 2^N \to \mathbb{R}$ be a monotone submodular function. For all $S \subseteq N$, $t \in [n]$, and $\Delta > 0$, let $\mathcal{D} = \mathcal{U}(S,t)$ and the discarded elements be $S^- = \left\{a : \mathbb{E}_{R\sim\mathcal{D}}\left[f_{R\setminus\{a\}}(a)\right] < \Delta\right\}$. Then:*

$$\left|S \setminus S^-\right| \leq \frac{\mathbb{E}_{R\sim\mathcal{D}}[f(R)]}{t\cdot\Delta} \cdot |S|.$$

Proof. At a high level, we first lower bound the value of a random set $R \sim \mathcal{D}$ by the marginal contributions of the remaining elements $S \setminus S^-$. Then, we lower bound these marginal contributions with the threshold Δ since these elements must have large enough marginal contributions to not

be removed. The first lower bound is the following:

$$\begin{aligned}
\mathbb{E}\left[f(R)\right] &\geq \mathbb{E}\left[f(R \cap (S \setminus S^-))\right] && \text{monotonicity} \\
&\geq \mathbb{E}\left[\sum_{a \in R \cap (S \setminus S^-)} f_{R \cap (S \setminus S^-) \setminus \{a\}}(a)\right] && \text{Submodularity} \\
&\geq \mathbb{E}\left[\sum_{a \in S \setminus S^-} \mathbb{1}_{a \in R} \cdot f_{R \setminus \{a\}}(a)\right] && \text{Submodularity} \\
&= \sum_{a \in S \setminus S^-} \mathbb{E}\left[\mathbb{1}_{a \in R} \cdot f_{R \setminus \{a\}}(a)\right].
\end{aligned}$$

By bounding the marginal contribution of remaining elements with the threshold Δ, we obtain

$$\begin{aligned}
\sum_{a \in S \setminus S^-} \mathbb{E}\left[\mathbb{1}_{a \in R} \cdot f_{R \setminus \{a\}}(a)\right] &= \sum_{a \in S \setminus S^-} \Pr\left[a \in R\right] \cdot \mathbb{E}\left[f_{R \setminus \{a\}}(a) | a \in R\right] \\
&\geq \sum_{a \in S \setminus S^-} \Pr\left[a \in R\right] \cdot \mathbb{E}\left[f_{R \setminus \{a\}}(a)\right] && \text{Submodularity} \\
&\geq \sum_{a \in S \setminus S^-} \Pr\left[a \in R\right] \cdot \Delta && \text{definition of } \Delta \\
&= |S \setminus S^-| \cdot \frac{t}{|S|} \cdot \Delta && \text{definition of } R
\end{aligned}$$

□

Corollary 1. Down-Sampling *with* $\Delta = \frac{\texttt{OPT}}{4k}$ *and* $\alpha = \frac{1}{\log n}$ *is* $\mathcal{O}(\frac{\log n}{\log \log n})$*-adaptive and obtains, in expectation, a* $\left(\frac{1}{\log n}\right)$*-approximation.*

Proof. Let $t = k$, $\Delta = \frac{\texttt{OPT}}{4k}$, $\alpha = \frac{1}{\log n}$, and $\epsilon = 1/2$ for Lemma 2. We first show the adaptive complexity and then the approximation guarantee. Recall that $\mathcal{D} = \mathcal{U}(S, t)$ is the uniform distribution over subsets of S of size t.

The adaptive complexity. We bound the number of rounds r where elements are removed from S. Notice that if Down-Sampling removes elements from S at some round, then $\mathbb{E}_{R \sim \mathcal{D}}\left[f(R)\right] < \alpha\texttt{OPT}$. We get

$$\begin{aligned}
|S \setminus S^-| &\leq \frac{\mathbb{E}_{R \sim \mathcal{D}}\left[f(R)\right]}{t \cdot \Delta} \cdot |S| && \text{Lemma 1} \\
&= \frac{\mathbb{E}_{R \sim \mathcal{D}}\left[f(R)\right]}{\texttt{OPT}} \cdot |S| \\
&\leq \frac{4}{\log n} \cdot |S| && \text{Algorithm}
\end{aligned}$$

Thus, after r rounds, there are $|S| \leq (4/\log n)^r \cdot n$ elements remaining. With

$$r = \frac{\log(n/k)}{\log\left(\frac{\log n}{4}\right)} \leq \frac{\log n}{\log\left(\frac{\log n}{4}\right)},$$

we obtain that $|S| \leq k$ and the algorithm terminates.

The approximation guarantee. There are two cases. If the algorithm terminates because $\mathbb{E}[f(R)] \geq \alpha\texttt{OPT}$, then returning a random set R is immediately (in expectation) a $\frac{1}{\log n}$-approximation with $\alpha = \frac{1}{\log n}$.

Otherwise, the algorithm returns S and we assume this is the case for the remaining of this proof. Let S_i and S_i^- be the sets S and $S^- := \left\{a \in S : \mathbb{E}_{R\sim\mathcal{D}}\left[f_{R\setminus\{a\}}(a)\right] < \Delta\right\}$ at round $i \in [r]$ of Down-Sampling where the notation S and S^- is by default these sets when the algorithm terminates. First, by monotonicity and subadditivity, we have

$$f(S) \geq f(O) - f(O \setminus S) = \texttt{OPT} - f\left(\left(\cup_{i=1}^r S_i^-\right) \cap O\right)$$

since $O \setminus S$ are the discarded optimal elements. If Down-Sampling removes elements from S_i at some round i, then $\mathbb{E}_{R\sim\mathcal{U}(S_i,k)}[f(R)] < \alpha\texttt{OPT}$ and we get

$$\begin{aligned}
f\left(\left(\cup_{i=1}^r S_i^-\right) \cap O\right) &= f\left(\cup_{i=1}^r \left(S_i^- \cap O\right)\right) \\
&\leq \sum_{i=1}^r f\left(O \cap S_i^-\right) && \text{Subadditivity} \\
&\leq \sum_{i=1}^r \left(|O \cap S_i^-|\Delta + \mathop{\mathbb{E}}_{R\sim\mathcal{D}}[f(R)]\right) && \text{Lemma 2} \\
&\leq |O \cap (\cup S_i^-)|\Delta + \sum_{i=1}^r \mathop{\mathbb{E}}_{R\sim\mathcal{D}}[f(R)] \\
&\leq k\Delta + r \cdot \mathop{\mathbb{E}}_{R\sim\mathcal{D}}[f(R)] \\
&\leq \frac{\texttt{OPT}}{2} + r\alpha\texttt{OPT} && \text{Algorithm}
\end{aligned}$$

By combining the above inequalities, we conclude that

$$\begin{aligned}
f(S) &\geq \texttt{OPT} - \frac{\texttt{OPT}}{2} + r\alpha\texttt{OPT} \\
&\geq \frac{1}{2}\texttt{OPT} - \frac{1}{\log\left(\frac{\log n}{4}\right)}\texttt{OPT}
\end{aligned}$$

□

D.3 Missing analysis from Section 2.2

Proposition 2. *For any constant $c \leq k/r$,* Up-Sampling *is an r-adaptive algorithm and obtains, w.p. $1-o(1)$, a $\left(1-\frac{1}{e}\right)\frac{c\cdot r}{k+c}$ approximation, with sample complexity $m = cn^{2+c}\log n$ at every round.*

Proof. Let S_i be the set S at round i of the upsampling algorithm (Algorithm 2). We first argue that for all subsets T of $N \setminus S_i$ of size c, T is contained in at least one sample drawn at round $i+1$. First, assume $c = k/r$. Then this is the coupon collector problem and with $n^2 \cdot \binom{n}{c} \cdot \log\left(\binom{n}{c}\right)$ samples, all subsets T of size c are observed at least once with probability at least $1 - 1/n^2$. If $c < k/r$, then observing sets of size k/r increases the probability that it contains set T of size c, so it is still the case that all subsets T of size c are observed at least once with probability at least $1 - 1/n^2$. By a union bound, this is the case for all rounds with probability at least $1 - 1/n$.

Let $\mathcal{S}^O_{i-1}$ be a collection of at most $\lceil k/c \rceil$ sets of size c partitioning elements in $O \setminus S_{i-1}$. Then,

$$\begin{aligned}\texttt{OPT} &\leq f(S_{i-1}) + \sum_{T \in \mathcal{S}^O_{i-1}} f_{S_{i-1}}(T) \\ &\leq f(S_{i-1}) + \sum_{T \in \mathcal{S}^O_{i-1}} (f(S_i) - f(S_{i-1})) \\ &\leq f(S_{i-1}) + (k/c+1)(f(S_i) - f(S_{i-1})\end{aligned}$$

where the first inequality is by submodularity and the second is by monotonicity and since at least one sample contains the c elements with largest marginal contribution to S_{i-1}. Thus, by a similar induction as in the analysis for the classical greedy algorithm,

$$f(S_i) \geq \left(1 - \left(1 - \frac{1}{k/c+1}\right)^i\right) \texttt{OPT}.$$

Thus, with $i = r$, the set S returned by the upsampling algorithm obtains the following approximation:

$$1 - \left(1 - \frac{1}{k/c+1}\right)^r = 1 - \left(\left(1 - \frac{1}{k/c+1}\right)^{k/c+1}\right)^{r/(k/c+1)} \geq 1 - e^{-r/(k/c+1)} \geq \left(1 - \frac{1}{e}\right)\frac{cr}{k+c}.$$

□

D.4 Missing Analysis from Section 2.3

Lemma 13. *For any $X, S \subseteq N$ such that $|X \cup R| \leq k$, let $\mathcal{D} = \mathcal{U}\left(S, \frac{k}{r}\right)$ and $R^+ = \text{argmax}_{i \in [m]} f(X \cup R_i)$. Then, with probability $1 - \delta$ over the samples drawn from $\mathcal{D}$,*

$$f_X(R^+) \geq \mathop{\mathbb{E}}_{R \sim \mathcal{D}}[f_X(R)] - \epsilon$$

with sample complexity $m = \frac{1}{2}\left(\frac{\texttt{OPT}}{\epsilon}\right)^2 \log\left(\frac{2}{\delta}\right)$.

Proof. By Lemma 11, with $m = \frac{1}{2}\left(\frac{\texttt{OPT}}{\epsilon}\right)^2 \log\left(\frac{2}{\delta}\right)$, with probability $1 - \delta$,

$$\left| v_X(S,t) - \mathop{\mathbb{E}}_{R \sim \mathcal{D}}[f_X(R)] \right| \leq \epsilon.$$

Since $v_X(S,t) = \frac{1}{m}\sum_{i=1}^m f_X(R_i)$, it must be the case that for at least one sample R used to compute $v_X(S,t)$,

$$f_X(R) \geq \mathop{\mathbb{E}}_{R \sim \mathcal{D}}[f_X(R)] - \epsilon.$$

We conclude by observing that the sample with largest marginal contribution $f_X(R) = f(X \cup R) - f(X)$ is returned. □

Lemma 3. Adaptive-Sampling *is $(r_u + r_d)$-adaptive with $r_u + r_d \leq r + \log_c n$ when $\Delta = c \cdot \frac{\alpha \texttt{OPT}}{k}$.*

Proof. We first bound the number of rounds r_d where Adaptive-Sampling downsamples. If $|S| \leq \frac{k}{r}$, then $|S \cup X| \leq k$ and the algorithm terminates since $|X| \leq k - \frac{k}{r}$ if the algorithm has not (yet) returned X. Notice that if Adaptive-Sampling removes elements from S at some round, then $\mathbb{E}_{R\sim\mathcal{D}}[f_X(R)] < \frac{\alpha}{r}\texttt{OPT}$. Thus, the number of elements remaining in S after one round of removing elements from S is

$$\begin{aligned} |S \setminus S^-| &\leq \frac{\mathbb{E}_{R\sim\mathcal{D}}[f_X(R)]}{t\cdot\Delta}\cdot|S| && \text{Lemma 1 with submodular function } f_X(\cdot) \\ &\leq \frac{\alpha\texttt{OPT}/r}{t\cdot\Delta}\cdot|S| && \text{Algorithm} \\ &\leq \frac{1}{c_\Delta}\cdot|S| && \Delta = c_\Delta\frac{\alpha\texttt{OPT}}{k} \text{ and } t = \frac{k}{r} \end{aligned}$$

Thus, after r_d rounds of removing elements, there are $|S| \leq (1/c_\Delta)^{r_d}\cdot n$ elements remaining. With

$$r_d = \frac{\log\left(\frac{n}{k/r}\right)}{\log c_\Delta} \leq \frac{\log n}{\log c_\Delta},$$

$|S| \leq k/r$ and the algorithm terminates.

For the second part of the lemma, after r rounds of upsampling, r disjoint samples of size $\frac{k}{r}$ have been added to X. Thus $|X| = k$ and the algorithm terminates with $r_u = r$. □

D.5 Missing Analysis from Section 2.4

D.5.1 Main lemmas with estimates

We bound the number of elements removed from S in each round.

Lemma 14. *Let $\epsilon_1, \delta > 0$. For all $X, S \subseteq N$, $t \in [n]$, and $\Delta \in \mathbb{R}$, let $\mathcal{D} = \mathcal{U}(S,t)$ and the removed elements be $S^- = \{a : v_X(S,t,a) < \Delta\}$. Then, with probability $1-\delta$,*

$$\left|S \setminus S^-\right| \leq \frac{v_X(S,t)}{t\cdot(\Delta - 2\epsilon_1)}\cdot|S|.$$

with sample complexity $m = n\left(\frac{\texttt{OPT}}{\epsilon_1}\right)^2\log\left(\frac{2n}{\delta}\right)$ at each round

Proof. At a high level, we first lower bound the value of a random set $R \sim \mathcal{D}$ by the marginal contributions of the remaining elements $S \setminus S^-$. Then, we lower bound these marginal contributions with the threshold Δ since these elements must have large enough marginal contributions to not

be removed. The first lower bound is the following:

$$\begin{aligned}
v_X(S,t)+\epsilon_1 &\geq \mathbb{E}\left[f_X(R)\right] && \text{Lemma 11 with } \epsilon=\epsilon_1\\
&\geq \mathbb{E}\left[f_X(R\cap(S\setminus S^-))\right] && \text{monotonicity}\\
&\geq \mathbb{E}\left[\sum_{a\in R\cap(S\setminus S^-)} f_{X\cup(R\cap(S\setminus S^-))\setminus\{a\}}(a)\right] && \text{Submodularity}\\
&\geq \mathbb{E}\left[\sum_{a\in S\setminus S^-} \mathbb{1}_{a\in R}\cdot f_{X\cup R\setminus\{a\}}(a)\right] && \text{Submodularity}\\
&= \sum_{a\in S\setminus S^-} \mathbb{E}\left[\mathbb{1}_{a\in R}\cdot f_{X\cup R\setminus\{a\}}(a)\right].
\end{aligned}$$

By bounding the marginal contribution of remaining elements with the threshold Δ, we obtain

$$\begin{aligned}
\sum_{a\in S\setminus S^-} \mathbb{E}\left[\mathbb{1}_{a\in R}\cdot f_{X\cup R\setminus\{a\}}(a)\right] &= \sum_{a\in S\setminus S^-} \Pr\left[a\in R\right]\cdot\mathbb{E}\left[f_{X\cup R\setminus\{a\}}(a)|a\in R\right]\\
&\geq \sum_{a\in S\setminus S^-} \Pr\left[a\in R\right]\cdot\mathbb{E}\left[f_{X\cup R\setminus\{a\}}(a)\right] && \text{Submodularity}\\
&\geq \sum_{a\in S\setminus S^-} \Pr\left[a\in R\right]\cdot(v_X(S,t,a)-\epsilon_1) && \text{Lemma 11 with } \epsilon=\epsilon_1\\
&\geq \sum_{a\in S\setminus S^-} \Pr\left[a\in R\right]\cdot(\Delta-\epsilon_1) && \text{definition of } \Delta\\
&= |S\setminus S^-|\cdot\frac{t}{|S|}\cdot(\Delta-\epsilon_1) && \text{definition of } R
\end{aligned}$$

where, with probability $1-\delta$, all the estimates hold with sample complexity $m=n\left(\frac{\texttt{OPT}}{\epsilon_1}\right)^2\log\left(\frac{2n}{\delta}\right)$ per round by Lemma 11. □

We bound the value of elements that survive the downsampling rounds.

Lemma 15. *For any $\epsilon,\delta>0$, $S\subseteq N$. and $t\in[n]$, let $\mathcal{D}=\mathcal{U}(S,t)$ and $\Delta\in\mathbb{R}_+$. Then, with probability $1-\delta$, the loss from removing elements $S^-:=\{a\in S: v_X(S,t,a)<\Delta\}$ is approximately bounded by the value of $R\sim\mathcal{D}$:*

$$f_X(O\cap S^-)\leq(1+\epsilon)|O\cap S^-|\Delta+\mathop{\mathbb{E}}_{R\sim\mathcal{D}}[f(R)]$$

with sample complexity $m=n\left(\frac{2\texttt{OPT}}{\epsilon\Delta}\right)^2\log\left(\frac{2n}{\delta}\right)$ per round.

Proof. The value of $O \cap S^-$ is upper bounded using the threshold Δ for elements to be in S^-,

$$\begin{aligned}
f_X(O \cap S^-) &\leq \mathbb{E}\left[f_X((O \cap S^-) \cup R) - f_X(R)\right] + \mathbb{E}\left[f_X(R)\right] && \text{monotonicity} \\
&\leq \mathbb{E}\left[\sum_{a \in O \cap S^-} f_{X \cup R}(a)\right] + \mathbb{E}\left[f_X(R)\right] && \text{Submodularity} \\
&= \sum_{a \in O \cap S^-} \mathbb{E}\left[f_{X \cup R}(a)\right] + \mathbb{E}\left[f_X(R)\right] \\
&\leq \sum_{a \in O \cap S^-} \mathbb{E}\left[f_{X \cup R \setminus \{a\}}(a)\right] + \mathbb{E}\left[f_X(R)\right] && \text{Monotonicity} \\
&\leq \sum_{a \in O \cap S^-} (v_X(S,t,a) + \epsilon\Delta) + \mathbb{E}\left[f_X(R)\right] && \text{Lemma 11 with } \epsilon = \epsilon\Delta \\
&\leq |O \cap S^-| \cdot (\Delta + \epsilon\Delta) + \mathbb{E}\left[f_X(R)\right] && \text{definition of } S^- \\
&= |O \cap S^-| \cdot (1+\epsilon) \cdot \Delta + \mathbb{E}\left[f_X(R)\right].
\end{aligned}$$

where, with probability $1-\delta$, all the estimates hold with sample complexity $m = n\left(\frac{2\texttt{OPT}}{\epsilon\Delta}\right)^2 \log\left(\frac{2n}{\delta}\right)$ per round by Lemma 11. □

D.5.2 The adaptive complexity

In the full algorithm, we force the algorithm to stop after r^+ rounds to obtain the adaptive complexity with probability 1. The loss from the event, happening with low probability, that the algorithm is forced to stop is accounted for in the δ probability of failure of the approximation of the algorithm.

Lemma 16. *For any $c_\Delta > 1$ and $\epsilon_1, \delta > 0$, assume $\Delta = c_\Delta \frac{\alpha v^\star}{k} + 2\epsilon_1$, $r^+ = r + \log_{c_\Delta} n$. Then, with probability $1-\delta$,* Adaptive-Sampling-Full *terminates either due $|X| = k$ or $|X \cup S| \leq k$, with sample complexity $m = n\left(\frac{\texttt{OPT}}{\epsilon_1}\right)^2 \log\left(\frac{2n}{\delta}\right)$ at each round. In addition,* Adaptive-Sampling-Full *is an r^+-adaptive algorithm.*

Proof. Adaptive-Sampling-Full is trivially an r^+-adaptive algorithm since the algorithm is forced to terminate after r^+ rounds. Next, we bound the number of rounds r_d where Adaptive-Sampling-Full downsamples. If $|S| \leq \frac{k}{r}$, then $|S \cup X| \leq k$ and the algorithm terminates since $|X| \leq k - \frac{k}{r}$ if the algorithm has not (yet) returned X. Notice that if Adaptive-Sampling-Full downsamples and removes elements from S, then $v_X(S,t) \geq (\alpha/r) \cdot v^\star$ Thus, the number of elements remaining in S after one round of removing elements from S is

$$\begin{aligned}
\left|S \setminus S^-\right| &\leq \frac{v_X(S,t)}{t \cdot (\Delta - 2\epsilon_1)} \cdot |S| && \text{Lemma 1} \\
&\leq \frac{\alpha v^\star / r}{t \cdot (\Delta - 2\epsilon_1)} \cdot |S| && \text{Algorithm} \\
&\leq \frac{1}{c_\Delta} \cdot |S| && \Delta = c_\Delta \frac{\alpha v^\star}{k} + 2\epsilon_1 \text{ and } t = \frac{k}{r}
\end{aligned}$$

Thus, after r_d rounds of removing elements, there are $|S| \leq (1/c_\Delta)^{r_d} \cdot n$ elements remaining. With

$$r_d = \frac{\log\left(\frac{n}{k/r}\right)}{\log c_\Delta} \leq \frac{\log n}{\log c_\Delta},$$

$|S| \leq k/r$ and the algorithm terminates. The sample complexity for Lemma 1 is $m = n\left(\frac{\texttt{OPT}}{\epsilon_1}\right)^2 \log\left(\frac{2n}{\delta}\right)$ at each round.

Next, after r rounds of upsampling, r disjoint samples of size $\frac{k}{r}$ have been added to X. Thus $|X| = k$ and the algorithm terminates with $r_u = r$. □

D.5.3 The approximation guarantee

Lemma 17. *Assume $\texttt{OPT} \leq v^\star \leq (1+\epsilon/3)\texttt{OPT}$, that either $|X| = k$ or $|X \cup S| \leq k$. Then,* Adaptive-Sampling-Proxy *is a $\frac{1}{3} - \epsilon$-approximation with probability $1-\delta$ and sample complexity $m = \frac{64}{\epsilon^2}\left(nk^2 \log\left(\frac{2n}{\delta}\right) + \log^2_{1+\epsilon/3} n \cdot \log\left(\frac{2}{\delta}\right) \cdot \frac{1}{\epsilon}\right)$ per round, with $\alpha = 1/3$, $r = \frac{3}{\epsilon} \cdot \log_{1+\epsilon/3} n$, $\Delta = \left(1+\frac{\epsilon}{3}\right)\frac{\alpha v^\star}{k} + 2\epsilon_1$, and $\epsilon_1 = \epsilon v^\star / 9k$.*

Proof. There are two cases depending on if X or $X \cup S$ is the solution.

Algorithm returns X. Let X_i and R_i^+ be the set X and the sample R added to X at the ith round of upsampling, $i \in [r]$. First note that since the sample with largest marginal contribution must have contribution larger than the average marginal contribution of the samples and since this is an upsampling round:

$$f_{X_i}\left(R_i^+\right) \geq v_X(S,t) \geq \alpha \cdot \frac{v^\star}{r}$$

Thus,

$$f(X) = \sum_{i=1}^{r} f_{X_i}\left(R_i^+\right) \geq \sum_{i=1}^{r} \alpha \cdot \frac{v^\star}{r} = \alpha \cdot v^\star \leq \frac{1}{3} \cdot \texttt{OPT}.$$

Algorithm returns $X \cup S$. We introduce some notation. Let $O = \{o_1, \ldots, o_k\}$ be the optimal solution indexed by an arbitrary order and X_i and S_i^- be the sets X and $S^- = \{a : \mathbb{E}_{R\sim\mathcal{D}}\left[f_{X\cup R\setminus\{a\}}(a)\right] < \Delta\}$ at the ith round of down-sampling, $i \in [r_d]$. First, by monotonicity, subadditivity, and again monotonicity, we get

$$f\left(S \cup X\right) \geq f(O) - f(O \setminus (S \cup X)) \geq \texttt{OPT} - f(O \setminus S)$$

The remaining of the proof bounds the loss $f(O\setminus(S\cup X))$ from optimal elements that were discarded from S. Next, we bound $f\left(O \setminus S\right)$. The elements in $O \setminus S$ are elements that have been discarded from S, so $O \setminus S = \cup_{i=1}^{r_d}\left(S_i^- \cap O\right)$, and we get

$$f\left(O \setminus S\right)) = f\left(\cup_{i=1}^{r_d}\left(S_i^- \cap O\right)\right) \leq f_X\left(\cup_{i=1}^{r_d}\left(S_i^- \cap O\right)\right) + f(X) \leq \sum_{i=1}^{r_d} f_X\left(O \cap S_i^-\right) + f(S \cup X).$$

where the first inequality is by monotonicity and the second by subadditivity and monotonicity. We obtain

$$\begin{aligned}
f_X\left(O \cap S_i^-\right) &\leq f_{X_i}\left(O \cap S_i^-\right) && \text{submodularity} \\
&\leq (1+\epsilon)|O \cap S_i^-| \cdot \Delta + \mathbb{E}\left[f_{X_i}(R)\right] && \text{Lemma 15} \\
&\leq (1+\epsilon)|O \cap S_i^-| \cdot \Delta + v_X(S,t) + \frac{2v^\star}{3r} && \text{Lemma 11 with } \epsilon = \frac{2v^\star}{3r} \\
&\leq (1+\epsilon)|O \cap S_i^-| \cdot \Delta + \alpha \cdot \frac{v^\star}{r} + \frac{2v^\star}{3r} && \text{Algorithm} \\
&\leq (1+\epsilon)|O \cap S_i^-| \cdot \Delta + \frac{v^\star}{r}
\end{aligned}$$

with sample complexity $m = n\left(\frac{2\texttt{OPT}}{\epsilon\Delta}\right)^2 \log\left(\frac{2n}{\delta}\right)$ for Lemma 15 and $m = 5\left(\frac{r\texttt{OPT}}{v^\star}\right)^2 \log\left(\frac{2}{\delta}\right) \leq 5r^2 \log\left(\frac{2}{\delta}\right)$ for Lemma 11, per round. Thus,

$$\begin{aligned}
f\left(O \setminus S\right) &\leq \sum_{i=1}^{r_d} \left((1+\epsilon)|O \cap S_i^-| \cdot \Delta + \frac{v^\star}{r}\right) + f(S \cup X) \\
&\leq (1+\epsilon)|O \cap (\cup_{i=1}^{r_d} S_i^-)| \cdot \Delta + \frac{r_d v^\star}{r} + f(S \cup X) \\
&\leq (1+\epsilon)k \cdot \Delta + \frac{r_d v^\star}{r} + f(S \cup X) \\
&\leq (1+\epsilon)^2 \frac{v^\star}{3} + \frac{r_d}{r}\texttt{OPT} + f(S \cup X).
\end{aligned}$$

By combining the previous inequalities, we get

$$f\left(S \cup X\right) \geq \texttt{OPT} - (1+\epsilon)^2 \frac{1}{3}v^\star + \frac{r_d}{r}\texttt{OPT} + f(S \cup X).$$

Thus,

$$\begin{aligned}
f\left(S \cup X\right) &\geq \frac{1}{2}\left(\texttt{OPT} - (1+\epsilon)^2 \frac{1}{3}v^\star - \frac{r_d}{r}\texttt{OPT}\right) \\
&\geq \frac{1}{2}\left(\texttt{OPT} - (1+\epsilon)^2 (1+\epsilon/3) \frac{1}{3}\texttt{OPT} - \frac{\epsilon}{3}\texttt{OPT}\right) \\
&\geq \left(\frac{1}{3} - \epsilon\right)\texttt{OPT}
\end{aligned}$$

where $v^\star \leq (1+\epsilon/3)\texttt{OPT}$ by assumption of the Lemma, $r_d \leq \log_{1+\epsilon/3} n$ by Lemma 16 with $c_\Delta = 1+\epsilon/3$, and with $\Delta = c_\Delta \frac{\alpha v^\star}{k} + 2\epsilon_1 = c_\Delta \frac{\alpha v^\star}{k} + \frac{\epsilon v^\star}{2k} = (1+\epsilon)\frac{v^\star}{3k}$, $\epsilon_1 = \epsilon v^\star/9k$, and $r = \frac{3}{\epsilon} \cdot \log_{1+\epsilon/2} n$. Finally, the sample complexity per round is $m = \frac{64}{\epsilon^2}\left(nk^2 \log\left(\frac{2n}{\delta}\right) + \log^2_{1+\epsilon/3} n \cdot \log\left(\frac{2}{\delta}\right) \cdot \frac{1}{\epsilon}\right)$. □

D.5.4 The main result for Adaptive-Sampling-Full

Theorem 3. *For any $\epsilon, \delta > 0$,* Adaptive-Sampling-Full *is a $\left(\log_{1+\epsilon/3} n \cdot \frac{3}{\epsilon} + 2\right)$-adaptive algorithm that, w.p. $1-\delta$, obtains a $(\frac{1}{3}-\epsilon)$-approximation, with sample complexity at every round $m = \frac{64}{\epsilon^2}\left(nk^2 \log\left(\frac{2n}{\delta}\right) + \log^2_{1+\epsilon/3} n \cdot \log\left(\frac{2}{\delta}\right) \cdot \frac{1}{\epsilon}\right)$, for maximizing a monotone submodular function under a cardinality constraint, with parameters $r = \frac{3}{\epsilon} \cdot \log_{1+\epsilon/3} n$, $\alpha = \frac{1}{3}$, and $\Delta = (1+\epsilon)\frac{v^\star}{3k}$.*

Proof. By Lemma 12, there is at least one $v^\star$ in Adaptive-Sampling-Full that is such that $\texttt{OPT} \leq v^\star \leq (1+\epsilon/3)\texttt{OPT}$. The solution to Adaptive-Sampling-Proxy($v^\star$) is then either X or $X \cup S$ with probability $1-\delta/2$ by Lemma 16 with $c_\Delta = 1+\epsilon/3$ and sample complexity $m = \left(\frac{r}{\epsilon}\right)^2 \log\left(\frac{4}{\delta} \cdot \log_{1+\epsilon/3} n\right)$. Then, again with probability $1-\delta/2$, this solution is a $\frac{1}{3}-\epsilon$ approximation by Lemma 17 with sample complexity $m = \frac{64}{\epsilon^2}\left(nk^2 \log\left(\frac{2n}{\delta}\right) + \log^2_{1+\epsilon/3} n \cdot \log\left(\frac{2}{\delta}\right) \cdot \frac{1}{\epsilon}\right)$ at every round. Since Adaptive-Sampling-Full returns the best set among a collection containing this solution, it is a $\frac{1}{3}-\epsilon$ approximation.

Since $\log_{1+\epsilon/3} n$ non-adaptive instances of Adaptive-Sampling-Proxy, each with adaptivity $r^+ = r + r_d \leq \frac{3}{\epsilon} \cdot \log_{1+\epsilon/3} n$, are run, the adaptivity of Adaptive-Sampling-Full is $\frac{3}{\epsilon} \cdot \log_{1+\epsilon/3} n + 2$ since there are two additional rounds to find the optimal singleton and the best solution in L. □

E Missing Analysis from Section 3

We begin with the round elimination lemma.

Lemma 5. *Assume $r \in \text{poly}(n)$. If there exist families of functions $\mathcal{F}_0, \ldots, \mathcal{F}_r$ such that the following two conditions hold:*

- ***Round elimination.*** *For all $i \in \{1, \ldots, r\}$, if there exists an i-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_i$, then there exists an $i-1$ adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation algorithm for $\mathcal{F}_{i-1}$;*
- ***Last round.*** *There does not exist a 0-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an α-approximation for $\mathcal{F}_0$;*

Then, there is no r-adaptive algorithm that obtains, w.p. $o(1)$, an α-approximation for $\mathcal{F}_r$.

Proof. The proof is by induction on the number of rounds r. If $r = 0$, then by the last round condition, $\mathcal{F}_0$ is not α optimizable in 0 adaptive rounds. If $r > 0$, then assume by contradiction that there exists an α-approximation r-adaptive algorithm for $\mathcal{F}_R$. By the round elimination condition, this implies that there exists an α-approximation $r-1$ adaptive algorithm for $\mathcal{F}_{r-1}$. This is a contradiction with the induction hypothesis for $r-1$. □

Next, we focus on the lemmas needed for the round elimination condition (Lemma 8). These lemmas are Lemma 6 and Lemma 7.

Lemma 6. *Let $\mathcal{F}_{R_i}$ be a randomized collection of functions in some $\mathfrak{F}$. Assume that for all $S \subseteq N$, w.p. $1-n^{-\omega(1)}$ over $\mathcal{F}_{R_i}$, for all $f_1, f_2 \in \mathcal{F}_{R_i}$, we have that $f_1(S) = f_2(S)$. Then, for any (possibly randomized) collection of $\text{poly}(n)$ non-adaptive queries $\mathcal{Q}$, there exists a deterministic collection of functions $\mathcal{F} \in \mathfrak{F}$ such that with probability $1-n^{-\omega(1)}$ over the randomization of $\mathcal{Q}$, for all queries $S \in \mathcal{Q}$ and all $f_1, f_2 \in \mathcal{F}$, $f_1(S) = f_2(S)$.*

Proof. We denote by $I(\mathcal{F}, \mathcal{S})$ the event that $f_1(S) = f_2(S)$ for all functions f_1, f_2 in a collection of functions $\mathcal{F}$ and all sets S in a collection of sets $\mathcal{S}$. Let $\mathcal{Q}$ be a randomized collection of $\text{poly}(n)$ non-adaptive queries and let $\mathcal{F}_R$ be a collection of functions s.t. for all $S \subseteq N$, w.p. $1-n^{-\omega(1)}$ over the randomization of $\mathcal{F}_R$, for all $f_1, f_2 \in \mathcal{F}$, we have that $f_1(S) = f_2(S)$.

Let $\mathcal{S}$ be any realization of the randomized collection of queries $\mathcal{Q}$. By a union bound over the $poly(n)$ queries $S \in \mathcal{S}$, $\Pr_{\mathcal{F}_R}[I(\mathcal{F}_R, \mathcal{S})] \geq 1 - n^{-\omega(1)}$. Since $\mathcal{F}_R \in \mathfrak{F}$, we obtain

$$\max_{\mathcal{F} \in \mathfrak{F}} \Pr_{\mathcal{Q}}[I(\mathcal{F}_i, \mathcal{Q})] \geq \Pr_{\mathcal{F}_R} \Pr_{\mathcal{Q}}[I(\mathcal{F}_R, \mathcal{Q})] \geq 1 - n^{-\omega(1)}$$

and there exists some $\mathcal{F} \in \mathfrak{F}$ such that w.p. $1 - n^{-\omega(1)}$ over the randomization of $\mathcal{Q}$, for all queries $S \in \mathcal{Q}$, $f_1(S) = f_2(S)$ for all $f_1, f_2 \in \mathcal{F}$. □

To show the indistinguishability property (Lemma 7) for Lemma 6, we need the following concentration bound, which shows that, a small set S has small intersection with small random sets with high probability.

Lemma 18. *Let R be a uniformly random subset of a set T. Consider a subset $S \subseteq T$ that is independent of the randomization of R and such that $|S| \cdot |R|/|T| \leq e^{-1}$, then*

$$\Pr\left[|S \cap R| \geq \log^2 n\right] \leq n^{-\omega(1)}.$$

Proof. We start by considering a subset L of S of size $\log^2 n$. We first bound the probability that L is a subset of R,

$$\Pr[L \subseteq R] \leq \prod_{a \in L} \Pr[a \in R] \leq \prod_{a \in L} \frac{|R|}{|T|} = \left(\frac{|R|}{|T|}\right)^{\log^2 n}.$$

We then bound the probability that $|S \cap R| \geq \log^2 n$ with a union bound over the events that a set L is a subset of R, for all subsets L of S of size $\log^2 n$:

$$\begin{aligned}
\Pr\left[|S \cap R| \geq \log^2 n\right] &\leq \sum_{L \subseteq S \,:\, |L| = \log^2 n} \Pr[L \subseteq R] \\
&\leq \binom{|S|}{\log^2 n} \cdot \left(\frac{|R|}{|T|}\right)^{\log^2 n} \\
&\leq \left(\frac{|S| \cdot |R|}{|T|}\right)^{\log^2 n} \\
&\leq \left(e^{-1}\right)^{\log^2 n} \\
&= n^{-\log n}
\end{aligned}$$

where the last inequality follows from the assumption that $|S| \cdot |R|/|T| \leq e^{-1}$. □

Corollary 1. *Assume $r \leq \log n$. For all $i \in [r]$, let $L_1, \ldots, L_{i-1}$ be fixed layers and S be a set of size $|S| \leq \frac{1}{8} n^{\frac{1}{r+1}}$ that is independent of the randomization of R_i, then*

$$\Pr_{S_i}\left[\left|S \cap \left(\cup_{j=i+1}^{r} L_j\right)\right| \leq \log^2 n\right] \geq 1 - n^{-\omega(1)}.$$

Proof. Since $r \leq \log n$, we get $|L_{j+1}| \leq \frac{1}{2}|L_j|$ for all j, which implies that $\sum_{j=i}^{r} |L_j| < 2|L_i|$.

Without loss, assume $S \subseteq R_i \cup \left(\cup_{j=i+1}^{r} L_j\right)$. The claim then immediately follows from Lemma 18 with $T = N \setminus \left(\cup_{j=1}^{i-1} S_j\right) = R_i \cup \left(\cup_{j=i+1}^{r} L_j\right)$ and $R = N \setminus \left(\cup_{j=1}^{i-1} L_i \cup R_i\right) = \cup_{j=i+1}^{r} L_j$, since

$$\frac{|S| \cdot \left|\cup_{j=i+1}^{r} L_j\right|}{\left|R_i \cup \left(\cup_{j=i+1}^{r} L_j\right)\right|} \leq \frac{\frac{1}{8} n^{\frac{1}{r+1}} \cdot 2n^{1-\frac{i+1}{r+1}}}{2n^{1-\frac{i}{r+1}}} \leq \frac{1}{8} \leq e^{-1}.$$

□

Lemma 7. *Assume $r \leq \log n$. For all $S \subseteq N$ and $i \in [r]$, with probability $1 - n^{-\omega(1)}$ over the randomization of $\mathcal{F}_{R_i} \in \mathfrak{F}_{r-i}$, for all $f_1, f_2 \in \mathcal{F}_{R_i}$, $f_1(S) = f_2(S)$.*

Proof. If $|S| \geq 8n^{\frac{1}{r+1}}$, then

$$f^P(S) = 2n^{\frac{1}{2r+2}}$$

for all $f^P \in \mathcal{F}_R$ with probability 1.

If $|S| < 8n^{\frac{1}{r+1}}$, then by Corollary 1, $\Pr_{R_i}\left[|S \cap \left(\cup_{j=i+1}^{r} L_j\right)| \leq \log^2 n\right] \geq 1 - n^{-\omega(1)}$. Thus, with probability $1 - n^{-\omega(1)}$ over the randomization of $\mathcal{F}_R$, for all $f^P \in \mathcal{F}_R$,

$$\begin{aligned} f^P(S) := &\sum_{j=1}^{i} \min(\ell_i, \log^2 n) + \left|S \cap \left(\cup_{j=i+1}^{r} L_j\right)\right| + \ell^\star \\ &+ \min\left(\frac{|S|}{8n^{\frac{1}{r+1}}}, 1\right) \cdot \left(2n^{\frac{1}{2r+2}} - \left(\sum_{j=1}^{i} \min(\ell_i, \log^2 n) + \left|S \cap \left(\cup_{j=i+1}^{r} L_j\right)\right| + \ell^\star\right)\right) \end{aligned}$$

The size of $S \cap \left(\cup_{j=i+1}^{r} L_j\right) = S \cap \left(N \setminus \left(\cup_{j=1}^{i-1} L_j \cup R_i\right)\right)$ is the same for all $f \in \mathcal{F}_R$. Thus, we conclude that with probability $1 - n^{-\omega(1)}$ over the randomization of $\mathcal{F}_R$, for all $f_1, f_2 \in \mathcal{F}_R$, $f_1(S) = f_2(S)$. □

Lemma 9. *For all $\mathcal{F} \in \mathfrak{F}_0$, there does not exist a 0-adaptive algorithm that obtains, with probability $n^{-\omega(1)}$, an $n^{-\frac{1}{2r+2}} \cdot \left((r+2)\log^2 n + 1\right)$ approximation.*

Proof. Let $\mathcal{F} \in \mathfrak{F}_0$ and $L_0, \ldots L_{r-1}$ be the fixed layers for all functions in $\mathcal{F}$. Let S be the solution returned by the algorithm. Consider $f_{R^\star} \sim \mathcal{F}$. Note that since the algorithm is 0-adaptive, S is independent of the randomization of $R^\star$ as a uniformly random subset of $L_r \cup L^\star$. Since

$$\frac{|S| \cdot |L^\star|}{|L_r \cup L^\star|} \leq \frac{\frac{1}{3} n^{\frac{1}{2r+2}} n^{\frac{1}{2r+2}}}{n^{1-\frac{r}{r+1}}} \leq e^{-1},$$

we get that

$$|S \cap L^\star| \leq \log^2 n$$

with probability $1 - n^{-\omega(1)}$ by Lemma 18 and we assume this is the case for the remaining of the proof. Since $|S| \leq \log^4 n$, we then obtain that

$$f_{R^\star}(S) \leq (r+1)\log^2 n + \log^2 n + \frac{\frac{1}{3} n^{\frac{1}{2r+2}}}{8n^{\frac{1}{r+1}}} \cdot 2n^{\frac{1}{2r+2}} \leq (r+2)\log^2 n + 1.$$

Then, for all $S \subseteq N$ such that $|S| \leq \frac{1}{3} n^{\frac{1}{2r+2}}$,

$$\mathop{\mathbb{E}}_{f_{R^\star} \sim \mathcal{U}(\mathcal{F})} \left[\frac{f_{R^\star}(R^\star)}{f_{R^\star}(S)} \right] \leq \frac{n^{\frac{1}{2r+2}}}{(r+2)\log^2 n + 1}.$$

Thus there exists at least one function $f \in \mathcal{F}$ such that the algorithm does not obtain, with probability $1 - n^{-\omega(1)}$, a $n^{-\frac{1}{2r+2}} \cdot \big((r+2)\log^2 n + 1\big)$-approximation. □

Lemma 19. *For all partitions P, the function f^P is monotone submodular.*

Proof. Let f be a function defined by a partition $P = (L_0, \ldots, L_r, L^\star)$. The marginal contribution $f_S(a_j) = f(S \cup \{a_j\}) - f(S)$ of an element $a_j \in L_j$, to a set S is

$$f_S(a_j) = \begin{cases} 1 - \dfrac{1}{8n^{\frac{1}{r+1}}} + \dfrac{1}{8n^{\frac{1}{r+1}}} \left(2n^{\frac{1}{2r+2}} - \displaystyle\sum_{i=1}^{r+1} \min(\ell_i, \log^2 n) - \ell^\star \right) - \dfrac{|S|}{8n^{\frac{1}{r+1}}} & \text{case } A \\ \dfrac{1}{8n^{\frac{1}{r+1}}} \left(2n^{\frac{1}{2r+2}} - \displaystyle\sum_{i=1}^{r+1} \min(\ell_i, \log^2 n) - \ell^\star \right) & \text{case } B \\ 1 - \min\left(\dfrac{|S|}{8n^{\frac{1}{r+1}}}, 1 \right) & \text{case } C \\ 0 & \text{case } D \end{cases}$$

where the cases are:

- case A if $|S| < 8n^{\frac{1}{r+1}}$, $j \neq r+2$, and $\ell_j < \log^2 n$,
- case B if $|S| < 8n^{\frac{1}{r+1}}$, $j \neq r+2$, and $\ell_j \geq \log^2 n$,
- case C if $|S| \geq 8n^{\frac{1}{r+1}}$, $j \neq r+2$, and $\ell_j < \log^2 n$,
- and case D if $|S| \geq 8n^{\frac{1}{r+1}}$, $j \neq r+2$, and $\ell_j \geq \log^2 n$.

Monotone. In case A, we have $-\frac{1}{8n^{\frac{1}{r+1}}} - \frac{|S|}{8n^{\frac{1}{r+1}}} \geq -1$ and $f_S(a_j) \geq 0$. It is easy to see that for all other cases, $f_S(a_j) \geq 0$ as well. Thus, f is monotone.

Submodular. Within each case, the marginal contributions are decreasing, as ℓ_i increases, for all i, and as $|S|$ increases. It remains to show that the marginal contributions are decreasing from one case to another when ℓ_i and $|S|$ increase.

- Since $-\frac{1}{8n^{\frac{1}{r+1}}} - \frac{|S|}{8n^{\frac{1}{r+1}}} \geq -1$ in A, marginal contributions are decreasing from A to B.
- Since $2n^{\frac{1}{2r+2}} - \sum_{i=1}^{r+1} \min(\ell_i, \log^2 n) - \ell^\star \geq 1$ in A, they are decreasing from A to C.
- Since the marginal contributions are non-negative, they are decreasing from B and C to D.

□