Submodular Maximization with Nearly Optimal Approximation, Adaptivity and Query Complexity

Matthew Fahrbach[*] Vahab Mirrokni[†] Morteza Zadimoghaddam[‡]

Abstract

Submodular optimization generalizes many classic problems in combinatorial optimization and has recently found a wide range of applications in machine learning (e.g., feature engineering and active learning). For many large-scale optimization problems, we are often concerned with the adaptivity complexity of an algorithm, which quantifies the number of sequential rounds where polynomially-many independent function evaluations can be executed in parallel. While low adaptivity is ideal, it is not sufficient for a distributed algorithm to be efficient, since in many practical applications of submodular optimization the number of function evaluations becomes prohibitively expensive. Motivated by these applications, we study the adaptivity and query complexity of adaptive submodular optimization.

Our main result is a distributed algorithm for maximizing a monotone submodular function with cardinality constraint k that achieves a $(1 - 1/e - \varepsilon)$-approximation in expectation. This algorithm runs in $O(\log(n))$ adaptive rounds and makes $O(n)$ calls to the function evaluation oracle in expectation. The approximation guarantee and query complexity are optimal, and the adaptivity is nearly optimal. Moreover, the number of queries is substantially less than in previous works. We also extend our results to the submodular cover problem to demonstrate the generality of our algorithm and techniques.

1 Introduction

Submodular functions have the natural property of diminishing returns, making them prominent in applied fields such as machine learning and data mining. There has been a surge in applying submodular optimization for data summarization [TIWB14, SSS07, SSSJ12], recommendation systems [EAG11], and feature selection for learning models [DK08, KED+17], to name a few applications. There are also numerous recent works that focus on maximizing submodular functions from a theoretical perspective. Depending on the setting where the submodular maximization algorithms are applied, new challenges emerge and hence more practical algorithms have been designed to solve the problem in distributed [MKSK13, MZ15, BENW15], streaming [BMKK14], and robust [MKK17, MBNF+17, KZK18] optimization frameworks. Most of the existing work assumes access to an oracle that evaluates the submodular function. However, function evaluations (oracle queries) can take a long time to process—for example, the value of a set depends on interactions with the entire input like in Exemplar-based Clustering [DF07] or when the function is computationally hard to evaluate like the log-determinant of sub-matrices [KZK18]. Although distributed algorithms for submodular maximization partition the input into smaller pieces to overcome these problems, each distributed machine may run a sequential algorithm that must wait for the answers of its past queries before making its next query. This motivates the study of the adaptivity complexity of submodular maximization, introduced by Balkanski and Singer [BS18] to study the number of rounds needed to interact with the oracle. As long as we can ask polynomially-many queries in parallel, we can ask them altogether in one round of interaction with the oracle.

To further motivate this adaptive optimization framework, note that in a wide range of machine learning optimization problems, the objective function can only be computed with oracle access to the function. In certain settings, the computation of the oracle is a time-consuming optimization problem that is treated as a black box (e.g., parameter tuning). In these settings, it is desirable to optimize a function with minimal number of rounds of interactions with the oracle. For an example of submodular optimization in this setting, consider the feature selection problem [DK08, KED+17], a critical problem for improving the model accuracy of machine learning models. The accuracy of a machine learning model does not necessarily have a closed-form formula and in some settings may be computed only by

[*]School of Computer Science, Georgia Institute of Technology. Email: matthew.fahrbach@gatech.edu. Supported in part by a National Science Foundation Graduate Research Fellowship under grant DGE-1650044. Part of this work was done while the author was a summer intern at Google Research, Zürich.

[†]Google Research, New York. Email: mirrokni@google.com.

[‡]Google Research, Zürich. Email: zadim@google.com.

Copyright © 2019
Copyright for this paper is retained by authors

re-evaluating the model with several multi-parameter tuning. It is known that for certain machine learning models, the accuracy of the model satisfies submodularity [DK08, KED+17]. In these cases, we have black-box access to the model accuracy function, which is time-consuming to compute. However, computing the model accuracy for many feature settings can be done in parallel. The adaptive optimization framework [BS18] is a realistic model for these kinds of distributed problems, and the insights from lower bounds and algorithmic techniques developed in this framework have a deep impact on distributed computing for machine learning applications in practice. For more motivation on the importance of round complexity of adaptive optimization framework, we refer the reader to [BS18].

While the number of rounds is an important measure to optimize, the complexity of answering oracle queries also motivates designing algorithms that are efficient in terms of the total number of oracle queries. Typically, we need to make at least a constant number of queries per element in the ground set to have a constant approximation guarantee. A fundamental question is how many queries per element are needed to achieve optimal approximation guarantees without compromising the minimum number of adaptive rounds. In this paper, we address this issue and design a simple algorithm for submodular maximization subject to a cardinality constraint that achieves optimal guarantees for the approximation factor and query complexity. Using the lower bound in [BS18], our algorithm also achieves nearly optmial adaptive round complexity.

1.1 Results and Techniques Our main result is a simple distributed algorithm for maximizing a monotone submodular function with cardinality constraint k that achieves an expected $(1 - 1/e - \varepsilon)$-approximation in $O(\log(n)/\varepsilon^2)$ adaptive rounds and makes $O(n \log(1/\varepsilon)/\varepsilon^3)$ queries to the function evaluation oracle in expectation. We emphasize that while our algorithm runs in $O(\log(n)/\varepsilon^2)$ rounds, only a constant number of queries are made per element. We note that, due to known lower bounds [BS18, MBK+15], the query complexity of the algorithm is optimal up to factors of $1/\varepsilon$ and the adaptivity is optimal up to factors of $1/\log\log(n)$ and $1/\varepsilon$. To achieve this result, we develop a number of techniques and subroutines that can be used in a variety of submodular optimization problems.

First, we develop the algorithm Threshold-Sampling in Section 3, which returns a subset of items from the ground set in $O(\log(n)/\varepsilon)$ adaptive rounds such that the expected marginal gain of each item in the solution is at least the input threshold. Furthermore, upon terminating it guarantees that all unselected items have marginal gain to the returned set less than the threshold. This effectively clears out all high value items. To achieve $O(\log(n)/\varepsilon)$ adaptivite complexity, Threshold-Sampling adds a random subset of candidate items to its current solution in each round in such a way that probabilistically filters out an ε-fraction of the remaining candidates. We then use Threshold-Sampling as a subroutine in a submodular maximization framework that constructs a solution by gradually reducing its threshold for acceptance. This framework runs Threshold-Sampling in parallel starting from many different initial thresholds, one of which is guaranteed to be sufficiently closed to the optimal starting threshold. Consequently, we do not increase the adaptivity complexity because these processes are independent. One of the challenges that arises when analyzing the approximation factor of this algorithm is that Threshold-Sampling returns sets of variable size. We overcome this by constructing an averaged random process that agrees with the state of the maximization algorithm at the beginning and end, but otherwise acts as an intermediate proxy. In Section 4, we demonstrate how to use Threshold-Sampling as a subroutine in a greedy maximization framework to achieve an expected $(1 - 1/e - \varepsilon)$-approximation to OPT.

Our second main technical contribution is the Subsample-Preprocessing algorithm. This algorithm iteratively subsamples the ground set and uses the output guarantees of Threshold-Sampling to reduce the ratio of the interval containing OPT from k to a constant. The adaptivity complexity of this subroutine is $O(\log(n))$ and its query complexity is $O(n)$. In particular, we show how to reduce the ratio of the interval in each step from R to $O(\text{poly}(\log(R))$ by subsampling the ground set and using a key lemma that relates OPT to the optimum in the subsampled set. This approximation guarantee (Lemma 5.1) for OPT is a function of the subsampling rate and may be of independent interest. Our ratio reduction technique and the algorithm Subsample-Preprocessing are presented in Section 5. Finally, in Section 6 we show how to use Threshold-Sampling to solve the submodular cover problem, which demonstrates that our techniques are readily applicable to problems beyond submodular maximization with cardinality constraints.

1.2 Related Work The problem of optimizing query complexity for maximizing a submodular function subject to cardinality constraints has been studied extensively. In fact, a linear-time $(1-1/e-\varepsilon)$-approximation algorithm called stochastic greedy was recently developed for this problem in [MBK+15]. We achieve the

Copyright © 2019
Copyright for this paper is retained by authors

same optimal query complexity in this paper, combined with nearly optimal $O(\log(n))$ adaptive round complexity. The applications of efficient algorithms for submodular maximization are wide spread due to the numerous applications in machine learning and data mining. Submodular maximization has also recently attracted a significant amount of attention in the streaming and distributed settings [LMSV11, KMVV15, MKSK13, BMKK14, MZ15, BENW15, BENW16, CQ19]. We note that the distributed MapReduce model and adaptivity framework of [BS18] are different in that the latter model does not allow for adaptivity within each round. In many previously studied distributed models, such as MapReduce, sequential algorithms on a given machine are allowed to be adaptive within one round for the part of the data they are processing locally. To highlight the fundamental difference between these models, Balkanski and Singer [BS18] showed that no constant-factor approximation is achievable in $O(\log(n)/\log\log(n))$ non-adaptive rounds; however, it is possible to achieve a constant-factor approximation in the MapReduce model in two rounds.

Balkanski and Singer [BS18] introduced the adaptive framework model for submodular maximization and showed that a $(1/3)$-approximation is achievable in $O(\log(n))$ rounds. Furthermore, they showed that $\Omega(\log(n)/\log\log(n))$ rounds are necessary for achieving any constant-factor approximation. They left the problem of achieving the optimal approximation factor of $1 - 1/e$ open, and as a followup posted a paper on arXiv achieving a $(1-1/e-\varepsilon)$-approximation in $O(\log(n))$ rounds [BRS19]. Their algorithm, however, requires $O(nk^2)$ queries [BRS19]. While writing this paper, another related work (on arXiv) was brought to our attention [EN19]. While [EN19] has a similar goal to ours and aims to minimize the number of adaptivity rounds and oracle queries, their query complexity is $O(n\text{poly}(\log(n)))$, or $O(\text{poly}(\log(n)))$ calls per element. In contrast, we present a simple algorithm that achieves optimal query complexity (i.e., a *constant number* of oracle queries per element). The query complexity of our algorithm is optimal up to factors of $1/\varepsilon$. While we did not aggresively optimize the dependence on $1/\varepsilon$, the dependence is better than that in the related works [BS18, BRS19, EN19].

2 Preliminaries

For a set function $f : 2^N \to \mathbb{R}$ and any $S, T \subseteq N$, let $\Delta(T, S) \overset{\text{def}}{=} f(S \cup T) - f(S)$ be the *marginal gain* of f at T with respect to S. We call N the ground set and let $|N| = n$. A function $f : 2^N \to \mathbb{R}$ is *submodular* if for every $S \subseteq T \subseteq N$ and $x \in N \setminus T$ we have $\Delta(x, S) \geq \Delta(x, T)$, where we overload the marginal gain notation for singletons. A natural class of submodular functions are those which are *monotone*, meaning that for every $S \subseteq T \subseteq N$ we have $f(S) \leq f(T)$. In the inputs to our algorithms, we let $f_S(T) \overset{\text{def}}{=} \Delta(T, S)$ denote a new submodular function with respect to S. We also assume that the ground set is global to all algorithms. Let S^* be a solution set to the maximization problem $\max_{S \subseteq N} f(S)$ subject to the cardinality constraint $|S| \leq k$, and let $\mathcal{U}(A, t)$ denote the uniform distribution over all subsets of A of size t.

Our algorithms take as input an *evaluation oracle* for f, which for any query $S \subseteq N$ returns $f(S)$ in $O(1)$ time. Given an evaluation oracle, we define the *adaptivity* of an algorithm to be the minimum number of rounds such that in each round the algorithm can make polynomially-many independent queries to the evaluation oracle. We measure the complexity of our distributed algorithms in terms of their query and adaptivity complexity. Last, we remark that in our runtime guarantees we take $1/\delta = \Omega(\text{poly}(n))$ so that the claims hold with high probability.

3 Threshold-Sampling Algorithm

We start by giving a high-level description of the Threshold-Sampling algorithm. For an input threshold τ, the algorithm iteratively builds a solution S and maintains a set of unselected candidate elements A over $O(\log(n)/\varepsilon)$ adaptive rounds. Initially, the solution is empty and all elements are candidates. In each round, the algorithm starts by filtering out candidiate elements whose current marginal gain is less than the threshold. Then the algorithm efficiently finds the largest set size t^* such that for $T \sim U(A, t^*)$ uniformly at random we have the property $\mathbb{E}[\Delta(T, S)/|T|] \geq (1 - \varepsilon)\tau$. Next, the algorithm samples $T \sim U(A, t^*)$ and updates the current solution to $S \cup T$. This probabilistic guarantee has two beneficial effects. First, it ensures that in expectation the average contribution of each element in the returned set is at least $(1 - \varepsilon)\tau$. Second, it implies that an expected ε-fraction of candidates are filtered out of A in each round. Therefore, the number of remaining elements that the algorithm considers in each round decreases geometrically in expectation. It follows that $O(\log(n)/\varepsilon)$ rounds are sufficient to guarantee with high probability that when the algorithm terminates, we have $|S| = k$ or the marginal gain of all remaining elements is below the threshold.

Before presenting and analyzing Threshold-Sampling, we define the distribution $\mathcal{D}_t$ from which Threshold-Sampling samples when estimating the maximum set size t^* in each round. Observe that sampling from this distribution can be simulated with with two calls to the evaluation oracle.

Copyright © 2019
Copyright for this paper is retained by authors

Definition 3.1. *Conditioned on the current state of the algorithm, consider the process where the set* $T \sim \mathcal{U}(A, t-1)$ *and then the element* $x \sim A \setminus T$ *are drawn uniformly at random. Let* $\mathcal{D}_t$ *denote the probability distribution over the indicator random variable*

$$I_t = \mathbb{1}[\Delta(x, S \cup T) \geq \tau].$$

It is useful to think of $\mathbb{E}[I_t]$ as the probability that the t-th marginal is at least the threshold τ if the candidates in A are inserted into S according to a uniformly random permutation.

Now that $\mathcal{D}_t$ is defined, we present the Threshold-Sampling algorithm and its guarantees below. Observe that this algorithm calls the Reduced-Mean subroutine, which detects when the mean of $\mathcal{D}_t$ falls below $1-\varepsilon$. We give the exact guarantees of Reduced-Mean in Lemma 3.2. Relating the mean of $\mathcal{D}_t$ to threshold values, this means that after sampling $T \sim U(A, t^*)$ and adding the elements of T to S, the expected marginal gain of the remaining candidates is at most $(1-\varepsilon)\tau$. This is the invariant we want to maintain in each iteration for an $O(\log(n/\delta)/\varepsilon)$ adaptive algorithm. We explain the mechanics of Threshold-Sampling in detail and prove Lemma 3.1 in Section 3.1.

Algorithm 1 Threshold-Sampling

Input: evaluation oracle for $f : 2^N \to \mathbb{R}$, constraint k, threshold τ, error ε, failure probability δ

```
Set smaller error ε̂ ← ε/3
Set iteration bounds r ← ⌈log_{(1−ε̂)^{−1}}(2n/δ)⌉, m ← ⌈log(k)/ε̂⌉
Set smaller failure probability δ̂ ← δ/(2r(m + 1))
Initialize S ← ∅, A ← N
for r rounds do
    Filter A ← {x ∈ A : Δ(x, S) ≥ τ}
    if |A| = 0 then
        break
    for i = 0 to m do
        Set t ← min{⌊(1 + ε̂)^i⌋, |A|}
        if Reduced-Mean(D_t, ε̂, δ̂) then
            break
    Sample T ∼ U(A, min{t, k − |S|})
    Update S ← S ∪ T
    if |S| = k then
        break
return S
```

Lemma 3.1. *The algorithm* Threshold-Sampling *outputs* $S \subseteq N$ *with* $|S| \leq k$ *in* $O(\log(n/\delta)/\varepsilon)$ *adaptive rounds such that the following properties hold with probability at least* $1-\delta$*:*

1. *There are* $O(n/\varepsilon)$ *oracle queries in expectation.*
2. *The expected average marginal*
$$\mathbb{E}[f(S)/|S|] \geq (1-\varepsilon)\tau.$$
3. *If* $|S| < k$*, then* $\Delta(x, S) < \tau$ *for all* $x \in N$*.*

Algorithm 2 Reduced-Mean

Input: access to a Bernoulli distribution $\mathcal{D}$, error ε, failure probability δ

```
Set number of samples m ← 16⌈log(2/δ)/ε²⌉
Sample X_1, X_2, ..., X_m ∼ D
Set μ̄ ← (1/m) Σ_{i=1}^{m} X_i
if μ̄ ≤ 1 − 1.5ε then
    return true
return false
```

We briefly remark that the Reduced-Mean subroutine is a standard unbiased estimator for the mean of a Bernoulli distribution. Since $\mathcal{D}_t$ is a uniform distribution over indicator random variables, it is a Bernoulli distribution. The guarantees of in Lemma 3.2 are consequences of Chernoff bounds and the proof of Lemma 3.2 is given in Appendix A.2.

Lemma 3.2. *For any Bernoulli distribution* $\mathcal{D}$, Reduced-Mean *uses* $O(\log(\delta^{-1})/\varepsilon^2)$ *samples to correctly report one of the following properties with probability at least* $1-\delta$*:*

1. *If the output is* `true`*, then the mean of* $\mathcal{D}$ *is* $\mu \leq 1-\varepsilon$*.*
2. *If the output is* `false`*, then the mean of* $\mathcal{D}$ *is* $\mu \geq 1-2\varepsilon$*.*

3.1 Analysis of Threshold-Sampling In order to prove the guarantees of Threshold-Sampling (Lemma 3.1), we first give a result that demonstrates the monotonic behavior of $\mathcal{D}_t$ at any point in the algorithm. This is a simple consequence of submodularity and the proof can be found in Appendix A.1.

Lemma 3.3. *In each round of* Threshold-Sampling*, we have* $\mathbb{E}[I_1] \geq \mathbb{E}[I_2] \geq \cdots \geq \mathbb{E}[I_{|A|}]$*.*

Now we show that if we choose the maximum set size t^* in each round such that the average marginal gain of a randomly sampled subset of size t^* is at least $(1-\varepsilon)\tau$, then we expect to filter an ε-fraction of the remaining candidates in the subsequent round. In Lemma 3.5 we show that our choice of the number of rounds is sufficient to guarantee that all unchosen elements have marginal gain less than τ with high probability.

Copyright © 2019
Copyright for this paper is retained by authors

LEMMA 3.4. *In each round of* THRESHOLD-SAMPLING*, an expected $\hat{\varepsilon}$-fraction of A is filtered with probability at least $1-\hat{\delta}$.*

Proof. This is a consequence of our choice of $t^* = \min\{t, k-|S|\}$ when sampling T. If $t^* = k - |S|$ then the algorithm breaks from the loop and there is no subsequent filtering. Otherwise, for any given round, we condition on the state of the algorithm. Let A_i denote the value of A after the filtering step in the current round, and let A_{i+1} be the random variable for the future value of A_i after being filtered in the next round. The algorithm draws $T \sim \mathcal{U}(A_i, t^*)$ uniformly at random, so by considering the process in Definition 3.1, filtering has the property that for $x \sim A_i \setminus T$, the expectation

$$\mathbb{E}[I_{t^*+1}] = \Pr(\Delta(x, S\cup T) \geq \tau) = \mathbb{E}\left[\frac{|A_{i+1}|}{|A_i \setminus T|}\right].$$

It follows that $\mathbb{E}[I_{t^*}] \leq 1-\hat{\varepsilon}$ with probability at least $1-\hat{\delta}$ by Lemma 3.2, and using Lemma 3.3 gives us $\mathbb{E}[I_{t^*+1}] \leq \mathbb{E}[I_{t^*}] \leq 1-\hat{\varepsilon}$. Furthermore, because $|A_i \setminus T| \leq |A_i|$ for all choices of T, we have

$$\mathbb{E}[|A_{i+1}|] \leq (1-\hat{\varepsilon}) \cdot |A_i|,$$

so an expected $\hat{\varepsilon}$-fraction of elements are filtered in each round with probability at least $1-\hat{\delta}$. □

LEMMA 3.5. *If* THRESHOLD-SAMPLING *terminates with $|S| < k$, $|A| = 0$ with probability at least $1-\delta$.*

Proof. Denote by A_i the random variable for the value of A after it is filtered in the i-th round of THRESHOLD-SAMPLING, and note that $A_0 = N$. By our choice of $\hat{\delta}$ and a union bound, we assume that with probability at least $1-\delta/2$ the expected $\hat{\varepsilon}$-filtrations happen at each step. Summing over all possible states in the i-th round, it follows from Lemma 3.4 that $\mathbb{E}[|A_{i+1}|] \leq (1-\hat{\varepsilon}) \cdot \mathbb{E}[|A_i|]$, which further implies

$$\mathbb{E}[|A_r|] \leq (1-\hat{\varepsilon})^r \cdot \mathbb{E}[|A_0|] = (1-\hat{\varepsilon})^r n.$$

Therefore, by Markov's inequality and our choice of the number of rounds r, we have

$$\Pr(|A_r| \geq 1) \leq (1-\hat{\varepsilon})^{\log_{(1-\hat{\varepsilon})^{-1}}(2n/\delta)} n = \delta/2.$$

It follows that

$$\Pr(|A_r| = 0) = 1 - \Pr(|A_r| \geq 1) \geq 1-\delta/2,$$

so the algorithm THRESHOLD-SAMPLING filters all elements in A upon completion with probability at least $(1-\delta/2)^2 \geq 1-\delta$. □

Using the guarantees for REDUCED-MEAN and the two lemmas above, we can prove Lemma 3.1.

Proof of Lemma 3.1. We start by showing that the adaptivity complexity of THRESHOLD-SAMPLING is $O(\log(n/\delta)/\varepsilon)$. By construction, the number of rounds is $O(\log_{(1-\varepsilon)^{-1}}(n/\delta))$ and there are polynomially-many queries in each, all of which are independent and rely on the current state of S.

To prove the three properties, we use Lemma 3.5 to assume that with probability at least $1-\delta$ all $O(rm)$ calls to REDUCED-MEAN yield correct outputs, and also that if the algorithm terminates with $|S| < k$ then we have $|A| = 0$. For the Property 1, the total number of oracle queries incurred by calling REDUCED-MEAN is $O(rm\log(\delta^{-1})/\varepsilon^2) = O(\log(n/\delta)\log(k)\log(\delta^{-1})/\varepsilon^4)$ by Lemma 3.2. Note that we can sample from $\mathcal{D}_t$ with two oracle calls. Now we bound the expected number of queries made while filtering over the course of the algorithm. Let A_i be a random variable for the value of A in the i-th round. It follows from the geometric property $\mathbb{E}[|A_{i+1}|] \leq (1-\hat{\varepsilon}) \cdot \mathbb{E}[|A_i|]$ in the proof of Lemma 3.5 and by linearity that the expected number of queries is bounded by

$$\mathbb{E}\left[\sum_{i=0}^{r} |A_i|\right] = \sum_{i=0}^{r} \mathbb{E}[|A_i|] \leq n \sum_{i=0}^{r} (1-\hat{\varepsilon})^i \leq n/\hat{\varepsilon}.$$

Since we set $\delta^{-1} = O(\text{poly}(n))$, the number of expected queries made when filtering dominates the sum of queries made when calling REDUCED-MEAN.

For property 2, it suffices to lower bound the expected marginal of every element added to S if we think of adding each set T to the output S one element at a time according to a uniformly random permutation. Let $t^* = \min\{t, k-|S|\}$ be the size of T at an arbitrary round. If $t^* = 1$ then $\mathbb{E}[\Delta(T,S)] \geq \tau$ by the definition of A. Otherwise, the candidate size $t \geq t^*/(1+\hat{\varepsilon})$ in the previous iteration satisfies $\mathbb{E}[I_t] \geq 1-2\hat{\varepsilon}$. Since $T \sim \mathcal{U}(A, t^*)$ uniformly at random, we can lower bound the expected marginal $\mathbb{E}[\Delta(T,S)]$ by the contribution of the first t elements, giving us

$$\begin{aligned}
\mathbb{E}[\Delta(T,S)] &\geq (\mathbb{E}[I_1] + \mathbb{E}[I_2] + \cdots + \mathbb{E}[I_t])\tau \\
&\geq t(1-2\hat{\varepsilon})\tau \\
&\geq \frac{t^*}{1+\hat{\varepsilon}} \cdot (1-2\hat{\varepsilon})\tau \\
&\geq t^*(1-\varepsilon)\tau.
\end{aligned}$$

The first of the inequalities above uses the definition of I_t in Definition 3.1 and is analogous to Markov's inequality, and the second follows from Lemma 3.3. Since the expected marginal of any individual element is at least $(1-\varepsilon)\tau$, we have Property 2.

Copyright © 2019
Copyright for this paper is retained by authors

To show Property 3, recall that if the algorithm terminates with $|S| < k$, then we have $|A| = 0$ with probability at least $1 - \delta$ by Lemma 3.5. Therefore, it follows from the definition of A and submodularity that $\Delta(x, S) < \tau$ for all $x \in N$. □

4 Exhaustive-Maximization Algorithm

In this section we show how the Threshold-Sampling algorithm fits into a greedy framework for maximizing monotone submodular functions with a cardinality constraint. We start by presenting the Exhaustive-Maximization algorithm, and then we prove its guarantees in Section 4.1. The algorithm Exhaustive-Maximization works as follows. Given an initial threshold τ, Exhaustive-Maximization constructs a solution by repeatedly running Threshold-Sampling at decreasing thresholds $(1-\varepsilon)^j\tau$ conditioned on the current partial solution. It is greedy in the sense that every time Threshold-Sampling is called, the expected average contribution of the elements in the returned set is at least $(1-\varepsilon)\tau$ and the marginal gain of all remaining elements is less than the current threshold. These properties allow us to prove an approximation guarantee with respect to the initial threshold τ.

Algorithm 3 Exhaustive-Maximization

Input: evaluation oracle for $f : 2^N \to \mathbb{R}$, constraint k, error ε, failure probability δ

```
1: Set upper bounds Δ* ← arg max{f(x) : x ∈ N}, r ← ⌈2 log(k)/ε⌉, m ← ⌈log(4)/ε⌉
2: Set smaller failure probability δ̂ ← δ/(r(m+1))
3: Initialize R ← ∅
4: for i = 0 to r in parallel do
5:     Set τ ← (1+ε)^i Δ*/k
6:     Initialize S ← ∅
7:     for j = 0 to m do                ▷ Until (1−ε)^j τ < τ/4
8:         Set T ← Threshold-Sampling(f_S, k − |S|, (1−ε)^j τ, ε, δ̂)
9:         Update S ← S ∪ T
10:        if |S| = k then
11:            break
12:    if f(S) > f(R) then
13:        Update R ← S
14: return R
```

To relate the quality of the solution to OPT, we first let $\Delta^* = \max\{f(x) : x \in N\}$ be an upper bound for all marginal contributions by submodularity and observe that $\Delta^* \le \text{OPT} \le k\Delta^*$. The threshold that Exhaustive-Maximization searches for is $\tau^* = \text{OPT}/k$, so it suffices to run the greedy thresholding algorithm for $O(\log(k)/\varepsilon)$ initial thresholds $(1+\varepsilon)^i\Delta^*/k$ in parallel and return the solution with maximum value. Since the algorithm will try some threshold τ close enough to τ^*, specifically $\tau \le \tau^* \le (1+\varepsilon)\tau$, the approximation to OPT follows. Note that by trying all thresholds in parallel, the adaptivity complexity of the algorithm does not increase. In Section 5 we present efficient preprocessing methods that reducing the ratio of the interval containing OPT.

Theorem 4.1. *For any monotone, nonnegative submodular function f, Exhaustive-Maximization outputs a set $S \subseteq N$ with $|S| \le k$ in $O(\log(n/\delta)/\varepsilon^2)$ adaptive rounds such that with probability at least $1-\delta$ the algorithm makes $O(n\log(k)/\varepsilon^3)$ oracle queries in expectation and $\mathbb{E}[f(S)] \ge (1 - 1/e - \varepsilon)\text{OPT}$.*

4.1 Analysis of Exhaustive-Maximization

To analyze the expected approximation factor of Exhaustive-Maximization, we first assume that all calls to Threshold-Sampling give correct outputs by our choice of $\hat{\delta}$ and a union bound. The analysis that follows is for one execution of the block in the for loop of Exhaustive-Maximization (Line 5 to Line 13) and assumes the initial value of τ is sufficiently close to $\tau^* = \text{OPT}/k$, satisfying the inequality $\tau \le \tau^* \le (1+\varepsilon)\tau$. Furthermore, we assume that the final output set is of size k. We refer to this modified block of Exhaustive-Maximization as the algorithm.

For a fixed input, as the algorithm runs it produces nonempty sets $T_1, T_2, \dots, T_m$, inducing a probability distribution over sequences of subsets. Denote their respective sizes by $t_1, t_2, \dots, t_m$ and the input values $(1-\varepsilon)^j\tau$ for which they were returned by $\tau_1, \tau_2, \dots, \tau_m$. We view the algorithm as a random process that adds elements to the output set S one at a time instead of set by set. Specifically, for each new T_i the algorithm adds each $x \in T_i$ to S in lexicographic order, producing a sequence of subsets $S_0 \subseteq S_1 \subseteq \cdots \subseteq S_k$ with $S_0 = \emptyset$. Note that there is no randomness in adding the elements of T_i once T_i is drawn.

Instead of analyzing the expected value of the partial solution $\mathbb{E}[f(S_i)]$ at each step, we consider an averaged version of this random process that is easier to analyze and whose final expected value is equal to $\mathbb{E}[f(S_k)]$. In particular, when the process draws a set T_ℓ, each element $x \in T_\ell$ contributes the same amount $\Delta(T_\ell, T_1 \cup \cdots \cup T_{\ell-1})/|T_\ell|$ to the value of the output set. Formally, we define the averaged version of the random process as

$$\hat{f}(S_i) \stackrel{\text{def}}{=} f(T_1 \cup \cdots \cup T_{\ell-1}) + \frac{i - (t_1 + \cdots + t_{\ell-1})}{t_\ell} \cdot \Delta(T_\ell, T_1 \cup \cdots \cup T_{\ell-1}),$$

Copyright © 2019
Copyright for this paper is retained by authors

where we use the overloaded notation $S_i = (i, T_1, T_2, \dots, T_\ell)$ to record the history of the process up to adding the i-th element. This means that $t_1 + t_2 + \cdots + t_{\ell-1} < i$ and $t_1 + t_2 + \cdots + t_\ell \geq i$. Note that for a given history $T_1, T_2, \dots, T_m$ of subsets, both processes agree after adding a complete subset. Analogously, we define the marginal of the i-th element X_i of this process to be

$$\hat{\Delta}(X_i, S_{i-1}) \stackrel{\text{def}}{=} \frac{\Delta(T_\ell, T_1 \cup \cdots \cup T_{\ell-1})}{t_\ell}.$$

Since the original algorithm induces a probability distribution over sequences of returned subsets, this defines a distribution over the values of $\hat{f}(S_i)$ and $\hat{\Delta}(X_i, S_{i-1})$ for all indices $i \in [k]$.

Lastly, it will be useful to define the distribution $\mathcal{H}$ over all possible (random bit) histories $(T_1, T_2, \dots, T_m)$ at the termination of the algorithm, and also the distributions $\mathcal{H}_i$, for all $i \in [k]$, over the possible histories *immediately* before adding the i-th element. This means that for each $h = (T_1, T_2, \dots, T_{\ell-1}) \in \operatorname{supp}(\mathcal{H}_i)$ we have $t_1 + t_2 + \cdots + t_{\ell-1} < i$ and there exists a set T_ℓ that can be drawn such that $t_1 + t_2 + \cdots + t_\ell \geq i$. Let $H_i(h)$ be the event over $\mathcal{H}_i$ such that the history is $h = (T_1, T_2, \dots, T_{\ell-1})$ and the next returned subset T_ℓ adds the i-th element. We condition our statements on $H_i(h)$, as this captures the state of the algorithm just before adding the i-th element. To provide intuition for Lemma 4.2, it is worth noting that $\mathcal{H}$ is a refinement of $\mathcal{H}_i$ conditioned on $H_i(h)$. This can be seen by recursively joining leaves in the probability tree of $\mathcal{H}$ until the i-th element is reached. The result is the probability tree of $\mathcal{H}_i$ conditioned on $H_i(h)$. In the statements that follow, the probabilities and expectations conditioned on $H_i(h)$ are over the distribution $\mathcal{H}_i$ and all other expressions are over the distribution $\mathcal{H}$ of final outcomes.

Lemma 4.1. *For all $i \in [k]$, events $H_i(h)$ and thresholds τ such that $\tau \leq \tau^* \leq (1+\varepsilon)\tau$, we have*

$$\mathbb{E}\left[\hat{\Delta}(X_i, S_{i-1}) \,\middle|\, H_i(h)\right] \geq \frac{(1-\varepsilon)^2}{k} \cdot \mathbb{E}\left[\mathrm{OPT} - \hat{f}(S_{i-1}) \,\middle|\, H_i(h)\right].$$

Proof. First we prove the claim for $i = 1$, and then we proceed by case analysis. If $i = 1$ there is no history, so it suffices to show that $E[\hat{f}(S_1)] \geq (1-\varepsilon)^2\mathrm{OPT}/k$. The first element belongs to the subset T_1 returned by Threshold-Sampling, so by Property 2 of Lemma 3.1, it follows that

$$\mathbb{E}\left[\hat{f}(S_1)\right] = \mathbb{E}\left[\frac{f(T_1)}{|T_1|}\right] \geq (1-\varepsilon)\tau \geq (1-\varepsilon)^2 \cdot \frac{\mathrm{OPT}}{k}.$$

Assuming that $i > 1$, let $i^* = t_1 + \cdots + t_{\ell-1} + 1$ be the size of the partial solution after adding the first element in T_ℓ. We consider the cases $i = i^*$ and $i > i^*$ separately. If $i = i^*$, observe that for monotone submodular functions we have

$$\begin{aligned}
f(S^*) &\leq f(S^* \cup S_{i-1}) \\
&\leq f(S_{i-1}) + \sum_{x \in S^*} \Delta(x, S_{i-1}) \\
&\leq f(S_{i-1}) + k \cdot \frac{\tau_\ell}{1-\varepsilon} \\
&\leq f(S_{i-1}) + \frac{k}{(1-\varepsilon)^2} \cdot \mathbb{E}\left[\frac{\Delta(T_\ell, S_{i-1})}{|T_\ell|} \,\middle|\, H_i(h)\right] \\
&= f(S_{i-1}) + \frac{k}{(1-\varepsilon)^2} \cdot \mathbb{E}\left[\hat{\Delta}(X_i, S_{i-1}) \,\middle|\, H_i(h)\right].
\end{aligned}$$

In the third inequality, we have $\tau_\ell/(1-\varepsilon)$ because Threshold-Sampling was run with this parameter immediately before running with the threshold τ_ℓ, which returned T_ℓ. The upper bound for the marginal $\Delta(x, S_{i-1})$ for $x \in S^*$ is then a consequence of Property 2 and Property 3 of Lemma 3.1.

The history $h = (T_1, T_2 \dots, T_{\ell-1})$ is known since we are conditioning on $H_i(h)$, so it follows that

$$f(S^*) - f(S_{i-1}) = \mathbb{E}\left[\mathrm{OPT} - \hat{f}(S_{i-1}) \,\middle|\, H_i(h)\right],$$

because there is no randomness in the expectation. Recall that $f(S_{i-1}) = \mathbb{E}[\hat{f}(S_{i-1}) \mid H_i(h)]$ because the set $S_{i-1} = T_1 \cup \cdots \cup T_{\ell-1}$ is a union of complete sets, and hence there are no partial, averaged contributions. Rearranging the previous inequalities gives

$$\mathbb{E}\left[\hat{\Delta}(X_i, S_{i-1}) \,\middle|\, H_i(h)\right] \geq \frac{(1-\varepsilon)^2}{k} \cdot \mathbb{E}\left[\mathrm{OPT} - \hat{f}(S_{i-1}) \,\middle|\, H_i(h)\right],$$

as desired.

Now we consider the case when $i > i^*$. Because we condition on the history $h = (T_1, T_2, \dots, T_{\ell-1})$ immediately before drawing a set T_ℓ that necessarily contains the i-element, the averaging property of $\hat{f}$ and the analysis for the previous case give us

$$\begin{aligned}
\mathbb{E}\left[\hat{\Delta}(X_i, S_{i-1}) \,\middle|\, H_i(h)\right] &= \mathbb{E}\left[\hat{\Delta}(X_{i^*}, S_{i^*-1}) \,\middle|\, H_i(h)\right] \\
&\geq \frac{(1-\varepsilon)^2}{k} \cdot \mathbb{E}\left[\mathrm{OPT} - \hat{f}(S_{i^*-1}) \,\middle|\, H_i(h)\right] \\
&\geq \frac{(1-\varepsilon)^2}{k} \cdot \mathbb{E}\left[\mathrm{OPT} - \hat{f}(S_{i-1}) \,\middle|\, H_i(h)\right].
\end{aligned}$$

The final inequality makes use of

$$\mathbb{E}[\hat{f}(S_{i-1}) \mid H_i(h)] \geq \mathbb{E}[\hat{f}(S_{i^*-1}) \mid H_i(h)],$$

Copyright © 2019
Copyright for this paper is retained by authors

which is a consequence of monotonicity and the averaging property of $\hat{f}$. This completes the proof for all $i \in [k]$. $\square$

LEMMA 4.2. *If for all $i \in [k]$ and events $H_i(h)$ we have*

$$\mathbb{E}\left[\hat{\Delta}(X_i, S_{i-1}) \;\middle|\; H_i(h)\right] \geq \frac{(1-\varepsilon)^2}{k} \cdot \mathbb{E}\left[\mathrm{OPT} - \hat{f}(S_{i-1}) \;\middle|\; H_i(h)\right],$$

then the algorithm returns a set S_k such that $\mathbb{E}[f(S_k)] \geq (1 - 1/e - \varepsilon)\mathrm{OPT}$.

Proof. Let $\delta_i = \mathrm{OPT} - \hat{f}(S_i)$, and observe that

$$\mathbb{E}[\Delta(X_i, S_{i-1}) \mid H_i(h)] = \mathbb{E}[\delta_{i-1} \mid H_i(h)] - \mathbb{E}[\delta_i \mid H_i(h)]$$

by the linearity of expectation. It follows from the assumption that

$$\mathbb{E}[\delta_i \mid H_i(h)] \leq \left(1 - \frac{(1-\varepsilon)^2}{k}\right) \cdot \mathbb{E}[\delta_{i-1} \mid H_i(h)].$$

Since $\mathcal{H}_i$ conditioned on the event $H_i(h)$ is a partition of the final outcome distribution $\mathcal{H}$, it follows from the law of total probability that

$$\begin{aligned}\mathbb{E}[\delta_i] &= \sum_{h \in \mathrm{supp}(\mathcal{H}_i)} \mathbb{E}[\delta_i \mid H_i(h)] \cdot \Pr(H_i(h)) \\ &\leq \left(1 - \frac{(1-\varepsilon)^2}{k}\right) \sum_{h \in \mathrm{supp}(\mathcal{H}_i)} \mathbb{E}[\delta_{i-1} \mid H_i(h)] \cdot \Pr(H_i(h)) \\ &\leq \left(1 - \frac{(1-\varepsilon)^2}{k}\right) \cdot \mathbb{E}[\delta_{i-1}].\end{aligned}$$

Iterating this inequality over the sequence of expectations $\mathbb{E}[\delta_i]$ and the using the fact $1 - x \leq e^{-x}$,

$$\mathbb{E}[\delta_k] \leq \left(1 - \frac{(1-\varepsilon)^2}{k}\right)^k \cdot \mathbb{E}[\delta_0] \leq \left(\frac{1}{e} + \varepsilon\right) \cdot \mathbb{E}[\delta_0].$$

We clearly have $\mathbb{E}[\hat{f}(S_0)] = \mathbb{E}[f(S_0)]$ and $\mathbb{E}[\hat{f}(S_k)] = \mathbb{E}[f(S_k)]$ by the construction of $\hat{f}$. Moreover, since f is nonnegative we have $\delta_0 = \mathrm{OPT} - \hat{f}(S_0) \leq \mathrm{OPT}$, thus $\mathbb{E}[f(S_k)] \geq (1 - 1/e - \varepsilon)\mathrm{OPT}$, which completes the proof. $\square$

Proof of Theorem 4.1. The cardinality constraint is satisfied by construction, so we start by proving the adaptivity complexity of EXHAUSTIVE-MAXIMIZATION. Lowering bounding OPT by Δ^* takes one adaptive round, and each execution of the block in the parallelized for loop is independent of all previous iterations. Therefore, it suffices to bound the adaptivity complexity of the for loop block (Line 5 to Line 13). Each invocation of THRESHOLD-SAMPLING is potentially dependent on the last since S can be updated in each round. Therefore, because there are $m = O(1/\varepsilon)$ iterations in the block, the total adaptivity complexity is $O(m \log(n/\delta)/\varepsilon) = O(\log(n/\delta)/\varepsilon^2)$ by Lemma 3.1.

Now we analyze the query complexiy of the algorithm. Each call to THRESHOLD-SAMPLING behaves as intended with probability at least $1 - \hat{\delta}$, so by our choice of $\hat{\delta}$ and a union bound, all calls to THRESHOLD-SAMPLING are correct with probability at least $1 - \delta$. Assume that this is the case from now on. By Lemma 3.1, the expected query complexity of THRESHOLD-SAMPLING is $O(n/\varepsilon)$. Therefore, it follows that the expected query complexity of EXHAUSTIVE-MAXIMIZATION is $O(n + rm(n/\varepsilon)) = O(n \log(k)/\varepsilon^3)$.

To prove the approximation guarantee, first observe that $\Delta^* \leq \mathrm{OPT} \leq k\Delta^*$ by submodularity. Therefore, we know that $\Delta^*/k \leq \tau^* \leq \Delta^*$. The values of τ considered are $(1+\varepsilon)^i \Delta^*/k$, so by our choice for the number of iterations $r = O(\log(n)/\varepsilon)$, there exists a τ satisfying $\tau \leq \tau^* \leq (1+\varepsilon)\tau$. Although we do not know this value of τ, we use its existence to give a guarantee by taking the maximum over all potential solutions. Therefore, with probability at least $1 - \delta$ we have

$$\mathbb{E}[f(S)] \geq (1 - 1/e - \varepsilon)\mathrm{OPT}$$

by Lemma 4.1 and Lemma 4.2, assuming that the returned set satisfies $|S| = k$. If instead $|S| < k$, then all unchosen elements $x \in N \setminus S$ have marginals $\Delta(x, S) \leq \tau/4$ by our choice of m and Property 3 of Lemma 3.1. Thus for any $\tau \leq \tau^*$, monotonicity and submodularity give

$$f(S^*) \leq f(S) + \sum_{x \in S^*} \Delta(x, S) \leq f(S) + k\tau/4,$$

which implies $f(S) \geq (1 - 1/4)\mathrm{OPT} \geq (1 - 1/e)\mathrm{OPT}$. This proves the approximation guarantee. $\square$

5 Achieving Linear Query Complexity via Preprocessing

In this section we demonstrate different ways of using THRESHOLD-SAMPLING to preprocess the interval containing OPT and reduce the total query complexity of the algorithm without increasing its adaptivity. In Section 5.1 we show how to we reduce the ratio of the interval containing OPT from $O(k)$ to $O(\log(k))$ in $O(\log\log(k))$ iterations of an imprecise binary search, reducing the query complexity from $O(n \log(k))$ to $O(n \log\log(k))$. In Section 5.2 we show how to iteratevely reduce the ratio of the interval from R

Copyright © 2019
Copyright for this paper is retained by authors

to $O(\log^4(R))$ until it is constant by subsampling the ground set and using the binary search decision subroutine. By tuning the parameters at each step according to the current ratio, we reduce the query complexity to $O(n)$ while maintaining $O(\log(n))$ adaptivity.

5.1 Query Reduction with a Binary Search To see how we can use a binary search, consider the output of THRESHOLD-SAMPLING$(f, k, \tau, 1-p, \delta)$ for an arbitrary value of τ. If $|S| = k$, then by Property 2 of Lemma 3.1 we have $pk\tau \leq \mathbb{E}[f(S)] \leq \text{OPT}$. Otherwise, if $|S| < k$ then by Property 3 of Lemma 3.1 we have $\Delta(x, S) \leq \tau$. In the second case, it follows for monotone submodular functions that $f(S) \leq \text{OPT} \leq f(S) + k\tau$. If $f(S) \leq k\tau$ then $\text{OPT} \leq 2k\tau$, and if $k\tau < f(S) \leq \text{OPT}$ then $pk\tau \leq \text{OPT}$. Therefore, after each call to THRESHOLD-SAMPLING we can determine with probability at least $1 - \hat{\delta}$ that one of the following inequalities is true: $\text{OPT} \leq 2k\tau$ or $pk\tau \leq \text{OPT}$. Note that these decisions may overlap, hence the term imprecise binary search. We give the guarantees of BINARY-SEARCH-MAXIMIZATION below and defer the proof of Corollary 5.1 to Appendix B.1.

COROLLARY 5.1. *For any monotone, nonnegative submodular function f, the algorithm* BINARY-SEARCH-MAXIMIZATION *outputs a subset $S \subseteq N$ with $|S| \leq k$ in $O(\log(n/\delta)/\varepsilon^2)$ adaptive rounds such that with probability at least $1-\delta$ the algorithm makes $O(n \log\log(k)/\varepsilon^3)$ oracle queries in expectation and $\mathbb{E}[f(S)] \geq (1 - 1/e - \varepsilon)\text{OPT}$.*

Algorithm 4 BINARY-SEARCH-MAXIMIZATION

Input: evaluation oracle for $f : 2^N \to \mathbb{R}$, constraint k, error ε, failure probability δ

```
Set max marginal Δ* ← max{f(x) : x ∈ N}
Set interval bounds L ← Δ*, U ← kΔ*
Set balancing parameter p ← 1/log(k)
Set upper bound m ← ⌊log log(k)/log(2)⌋
Set smaller failure probability δ̂ ← δ/(m + 1)
for i = 1 to m do
    Set τ ← sqrt(LU/(2p))/k
    Set S ← THRESHOLD-SAMPLING(f, k, τ, 1−p, δ̂)
    if |S| < k and f(S) ≤ kτ then
        Update U ← 2kτ
    else
        Update L ← pkτ
return EXHAUSTIVE-MAXIMIZATION(f, k, ε, δ̂) modified to search over [L/k, U/k]
```

5.2 Query Reduction by Subsampling In this subsection we describe how to combine THRESHOLD-SAMPLING and subsampling to preprocess the interval containing OPT in $O(\log(n))$ adaptive rounds and with a total of $O(n)$ queries so that the final interval has a constant ratio. There are three main ideas underlying the algorithm SUBSAMPLE-PREPROCESSING. First, we subsample the ground set N so that the query complexity of running THRESHOLD-SAMPLING is sublinear (Lemma 3.1). Second, we relate the optimal solution in the sampled space to OPT in terms of the sampling probability. Third, we repeatedly subsample the ground set N with a granularity that depends on the current ratio R of the feasible interval, and in each of these iterations we run $O(\log(R))$ imprecise binary search decisions in parallel (by calling THRESHOLD-SAMPLING with error $1-p$ as described in Section 5.1) to reduce the ratio from R to $O(\log^4(R))$. Note that the adaptivity of each step is $O(\log(n)/\log(1/p))$ by Lemma 3.1 because the calls are distributed. There $O(\log^*(R))$ ratio reduction rounds, but by our choice of ℓ and p in every round, the total number of adaptive rounds is $O(\log(n/\delta))$. When SUBSAMPLE-PREPROCESSING terminates, the interval containing OPT has a constant ratio, and we run EXHAUSTIVE-MAXIMIZATION modified to search over this new interval for the final solution.

Now we formally present SUBSAMPLE-PREPROCESSING and state the lemmas that are prerequisites for its guarantees. All proofs regarding preprocessing are deferred to Appendix B.2. We first show how the optimal solution in a subsampled ground set relates to OPT in terms of the subsampling probability.

LEMMA 5.1. *For any submodular function, sample each element in the ground set N independently with probability $1/\ell$. If the resulting subsampled set is N', denote the optimal solution in N' by OPT'. If Δ^* is an upper bound for the maximum marginal in N, then with probability at least $1-\delta$,*

$$\frac{1}{2}(\Delta^* + \text{OPT}') \leq \text{OPT} \leq \frac{2\ell}{\delta}(\Delta^* + \text{OPT}').$$

Next we show that in each round of SUBSAMPLE-PREPROCESSING, the current ratio R becomes polylogarithmically smaller until it drops below a constant lower bound threshold R^*. The adaptivity and query complexity of this iteration is sublinear as a function of R, so by summing over the $O(\log^*(k))$ rounds of SUBSAMPLE-PREPROCESSING, the overall number of adaptive rounds and expected number of queries are $O(\log(n/\delta))$ and $O(n)$, respectively.

Copyright © 2019
Copyright for this paper is retained by authors

Lemma 5.2. *For any monotone submodular function f, let $[L, U]$ be an interval containing* OPT *with $U/L = R$. For any ratio $R > 0$ and with probability at least $1-\delta$, we can compute a new feasible interval with ratio $(8/\delta)\log^3(R)$ such that:*

- *The number of adaptive rounds is $O\left(\frac{\log(n/\delta)}{\log\log(R)}\right)$.*
- *The number of queries needed is $O(n/\log(R))$.*

Algorithm 5 Subsample-Preprocessing

Input: evaluation oracle for $f : 2^N \to \mathbb{R}$, constraint k, error ε, constant failure probability δ

Set max marginal $\Delta^* \leftarrow \max\{f(x) : x \in N\}$
Set interval bounds $L \leftarrow \Delta^*$, $U \leftarrow k\Delta^*$, $R^* \leftarrow 2 \cdot 10^6/\delta^2$
while $U/L \geq R^*$ **do**
    Set sampling ratio $\ell \leftarrow \log^2(R)$
    Set imprecise decision accuracy $p \leftarrow 1/\log(R)$
    Set upper bound $m = \lceil \log_2(R) \rceil$
    Set smaller failure $\hat{\delta}_R \leftarrow \delta/(2(m+1)\log(R))$
    Set $N' \leftarrow$ Subsample$(N, 1/\ell)$
    for $i = 0$ to m in parallel **do**
        Set $\tau_i \leftarrow 2^i(L/k)$
        Set $S_i' \leftarrow$ Threshold-Sampling$(f', k, \tau_i, 1-p, \hat{\delta}_R)$
        Decide if $\text{OPT}_i' \leq 2k\tau_i$ or $\text{OPT}_i' \geq pk\tau_i$ using S_i' ▷ Imprecise binary search decision
    if $\text{OPT}_i' \leq 2k\tau_i$ for all $i = 0$ to m **then**
        Update $L \leftarrow (\Delta^* + L)/2$
        Update $U \leftarrow (4\ell/\hat{\delta}_R)(\Delta^* + L)$
    else if $\text{OPT}_i' \geq pk\tau_i$ for all $i = 0$ to m **then**
        Update $L \leftarrow (p/2)(\Delta^* + U)$
        Update $U \leftarrow (2\ell/\hat{\delta}_R)(\Delta^* + U)$
    else
        Set $i^* \leftarrow$ first $i = 0$ to m such that $\text{OPT}_i' \leq 2k(2^i/k)$ and $pk(2^{i+1}/k) \leq \text{OPT}_i'$
        Update $L \leftarrow (p/2)(\Delta^* + 2^{i^*+1})$
        Update $U \leftarrow (2\ell/\hat{\delta}_R)(\Delta^* + 2^{i^*+1})$
return $[L, U]$

Lemma 5.3. *For any monotone submodular function f and constant $0 < \delta \leq 1$, with probability at least $1-\delta$ the algorithm* Subsample-Preprocessing *returns an interval containing* OPT *with ratio $O(1/\delta^2)$ in $O(\log(n/\delta))$ adaptive rounds and uses $O(n)$ queries in expectation.*

Last, we show how to use the reduced interval returned by Subsample-Preprocessing with the Exhaustive-Maximization framework to get the Subsample-Maximization algorithm.

Theorem 5.1. *For any nonnegative, monotone submodular function f and constant $0 < \varepsilon \leq 1$, the algorithm* Subsample-Maximization *outputs a subset $S \subseteq N$ with $|S| \leq k$ in $O(\log(n)/\varepsilon^2)$ adaptive rounds such that $\mathbb{E}[f(S)] \geq (1 - 1/e - \varepsilon)\text{OPT}$ and with probability at least $1-\varepsilon$ the algorithm makes $O(n \log(1/\varepsilon)/\varepsilon^3)$ oracle queries in expectation.*

Proof. Set a smaller error $\hat{\varepsilon} = \varepsilon/4$ and run Subsample-Preprocessing$(f, k, \hat{\varepsilon})$ to obtain an interval with ratio $O(\varepsilon^{-2})$ that contains OPT with probability at least $1/\hat{\varepsilon}$. Next, modify and run Exhaustive-Maximization$(f, k, \hat{\varepsilon}, \hat{\varepsilon})$ so that it searches over the interval with ratio $O(\varepsilon^{-2})$. Both Subsample-Preprocessing and Exhaustive-Maximization succeed with probability at least $1 - 2\hat{\varepsilon}$ by a union bound. Therefore, conditioning on the success of both events we have

$$\begin{aligned}\mathbb{E}[f(S)] &\geq (1 - 1/e - \hat{\varepsilon})\text{OPT} \cdot (1 - 2\hat{\varepsilon}) \\ &\geq (1 - 1/e - \varepsilon)\text{OPT},\end{aligned}$$

as desired. The complexity guarantees of Subsample-Maximization follow from the guarantees of Lemma 5.3 and Theorem 4.1. ☐

6 Using the Threshold-Sampling Algorithm for Submodular Cover

In the submodular cover problem, we aim to find a minimum cardinality subset S such that $f(S)$ is at least some target goal L. In some sense, this problem can be viewed as the dual of submodular maximization with a cardinality constraint. To formalize the submodular cover problem, we want to solve $\min_{S \subseteq N} |S|$ subject to the value lower bound $f(S) \geq L$. We overload the notation S^* to denote the lexicographically least minimum size set satisfying the value lower bound. Therefore, the value of OPT is the cardinality $|S^*|$. To overcome granularity issues resulting from arbitrarily small marginal gains, a standard assumption is to work with integer-valued submodular functions.

The greedy algorithm provides the state-of-the-art approximation for submodular cover by outputting a set of size $O(\log(L)|S^*|)$. There have been recent attempts [MZK16] to develop distributed algorithms based on the greedy approach that achieve similar approximation factors, but these algorithms have suboptimal adaptivity complexity because the summarization algorithm of the centralized machine is sequential. Here, we show how to apply the ideas behind the Threshold-Sampling algorithm to submodular cover so that the

Copyright © 2019
Copyright for this paper is retained by authors

algorithm runs in a logarithmic number of adaptive rounds without losing the approximation guarantee.

We start by giving a high-level description of our algorithm. Similar to [MZK16], which attempts to imitate the greedy algorithm, we initialize $S = \emptyset$ and set the threshold τ to the highest marginal value Δ^* of elements in N. Then we repeatedly add sets of items to S whose average value to S is at least $(1-\varepsilon)\tau$. Whenever we run out of high value items, we lower the threshold from τ to $(1-\varepsilon)\tau$ and repeat this process. Unlike the cardinality constraint setting, the stopping condition of this algorithm is when the value of $f(S)$ reaches the lower bound L.

Specifically, for each threshold τ we run a variant of Threshold-Sampling called Threshold-Sampling-For-Cover as a subroutine to find a maximal set of valuable items in $O(\log(n))$ adaptive rounds. The first difference between this algorithm and Threshold-Sampling is that it takes the value lower bound L as part of its input instead of a cardinality constraint k. Therefore, we slightly modify the Threshold-Sampling algorithm as follows. Since we do not have an explicit constraint on the number of elements that we can to add, we set m such that it is possible to add all of the elements at once. This change is reflected in Lines 2 and 13 of Threshold-Sampling-For-Cover. We use $\lfloor (L - f(S))/((1-\hat{\varepsilon})\tau) \rfloor$ as an upper bound for the number of elements that can be added in each iteration. This is a consequence of our initial choice of $\tau = \Delta^*$ and our method for lowering the threshold as the algorithm progresses. We are guaranteed that for the current value of τ, no element has marginal gain more than $\tau/(1-\hat{\varepsilon})$ to the selected set S. Furthermore, the average contribution of elements in T satisfies $\mathbb{E}[\Delta(T,S)/|T|] \geq (1-\hat{\varepsilon})\tau$, justifying our choice for the upper bound. We are essentially applying Property 3 of Lemma 3.1 to the previous value of τ and using Property 2 of Lemma 3.1 for the current stage.

To give intuition for why this leads to an acceptable approximation factor, observe that for the current threshold τ, the optimum (conditioned on our current choice of S) must have at least $(L - f(S))/(\tau/(1-\hat{\varepsilon}))$ elements since the marginal gains are bounded. Our cardinality upper bound $(L - F(S))/((1-\hat{\varepsilon})\tau)$ implies that the algorithm does not add too many more elements than the optimum in each round. The final modification is in the stopping condition of Line 15, where we check whether or not we have reached the value lower bound L.

Algorithm 6 Threshold-Sampling-For-Cover

Input: evaluation oracle for $f : 2^N \to \mathbb{R}$, value goal L, threshold τ, error ε, failure probability δ

```
1: Set smaller error ε̂ ← ε/3
2: Set iteration bounds r ← ⌈log_{(1−ε̂)^{−1}}(2n/δ)⌉, m ← ⌈log(n)/ε̂⌉
3: Set smaller failure probability δ̂ ← δ/(2r(m+1))
4: Initialize S ← ∅, A ← N
5: for r rounds do
6:     Filter A ← {x ∈ A : Δ(x,S) ≥ τ}
7:     if |A| = 0 then
8:         break
9:     for i = 0 to m do
10:        Set t ← min{⌊(1+ε̂)^i⌋, |A|}
11:        if Reduced-Mean(D_t, ε̂, δ̂) then
12:            break
13:    Set T ∼ U(A, min{t, ⌊(L − f(S))/((1 − ε̂)τ)⌋})
14:    Update S ← S ∪ T
15:    if f(S) ≥ L then
16:        break
17: return S
```

Corollary 6.1. Threshold-Sampling-For-Cover *outputs* $S \subseteq N$ *in* $O(\log(n/\delta)/\varepsilon)$ *adaptive rounds such that the following properties hold with probability at least* $1-\delta$*:*

1. *The expected average marginal*

$$\mathbb{E}[f(S)/|S|] \geq (1-\varepsilon)\tau.$$

2. *If* $f(S) < L$*, then* $\Delta(x,S) < \tau$ *for all* $x \in N$*.*

Proof. The proof is a direct consequence of the proof for Lemma 3.1. □

As explained above, we iteratively use Threshold-Sampling-For-Cover as a subroutine starting from the highest threshold $\tau = \Delta^*$ to ensure that either we reach the value goal L or that there is no element with marginal value above τ to the current set S. If we have not reached the value lower bound L, we reduce the threshold by a factor of $(1-\varepsilon)$ and repeat. This idea is summarized in the Adaptive-Greedy-Cover algorithm. We note that by the integrality assumption on f, this algorithm is guaranteed to output a feasible solution in a deterministic amount of time, since the threshold can potentially be lowered enough such that any element can be added to the solution.

Copyright © 2019
Copyright for this paper is retained by authors

Algorithm 7 ADAPTIVE-GREEDY-COVER

Input: evaluation oracle for $f : 2^N \to \mathbb{R}$, value goal L

Set error $\varepsilon \leftarrow 1/2$
Set upper bounds $\Delta^* \leftarrow \arg\max\{f(x) : x \in N\}$, $m \leftarrow \lceil \log(\Delta^*)/\varepsilon \rceil$
Set failure probability $\delta \leftarrow 1/(n(m+1))$
Initialize $S \leftarrow \emptyset$
for $i = 0$ to m **do** ▷ Until $(1-\varepsilon)^i \Delta^* < 1$
  Set $\tau \leftarrow (1-\varepsilon)^i \Delta^*$
  Set $T \leftarrow$ THRESHOLD-SAMPLING-FOR-COVER($N, f_S, L - f(S), \tau, \varepsilon, \delta$)
  Update $S \leftarrow S \cup T$
  if $f(S) \geq L$ **then**
    break
return S

THEOREM 6.1. *For any integer-valued, nonnegative, monotone submodular function f, the algorithm* ADAPTIVE-GREEDY-COVER *outputs a set $S \subseteq N$ with $f(S) \geq L$ in $O(\log(n \log(L)) \log(L))$ adaptive rounds such that $\mathbb{E}[|S|] = O(\log(L)|S^*|)$.*

We defer the proof of Theorem 6.1 to Appendix C.1 and remark that it follows a similar line of reasoning to the analysis of the approximation factor for EXHAUSTIVE-MAXIMIZATION in Lemma 4.2. The main difference between these two proofs is that for Theorem 6.1 we need to show that ADAPTIVE-GREEDY-COVER makes geometric progress towards its value constraint L not only in expectation, but also with constant probability. We do this by considering the progress of the algorithm in intervals of $O(|S^*|)$ elements, which ultimately allows us to conveniently analyze $\mathbb{E}[|S|]$ using facts about the negative binomial distribution. Lastly, we have not optimized the constant in the approximation factor, but one could do this by more carefully considering how the error ε affects the lower bound for the constant probability term in our analysis.

References

[BENW15] Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1236–1244. JMLR Workshop and Conference Proceedings, 2015.

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

[BMKK14] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 671–680. ACM, 2014.

[BRS19] Eric Balkanksi, Aviad Rubinstein, and Yaron Singer. An exponential speedup in parallel running time for submodular maximization. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA). SIAM, 2019.

[BS06] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, pages 31–40. ACM, 2006.

[BS18] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *ACM Symposium on Theory of Computing (STOC)*, 2018.

[CQ19] Chandra Chekuri and Kent Quanrud. Submodular function maximization in parallel via the multilinear relaxation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA). SIAM, 2019.

[DF07] Delbert Dueck and Brendan J Frey. Non-metric affinity propagation for unsupervised image categorization. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[DK08] Abhimanyu Das and David Kempe. Algorithms for subset selection in linear regression. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 45–54, 2008.

[EAG11] Khalid El-Arini and Carlos Guestrin. Beyond keyword search: discovering relevant scientific literature. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 439–447. ACM, 2011.

[EN19] Alina Ene and Huy L. Nguyen. Submodular maximization in nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA). SIAM, 2019.

[KED+17] Rajiv Khanna, Ethan R. Elenberg, Alexandros G. Dimakis, Sahand Negahban, and Joydeep Ghosh. Scalable greedy feature selection via weak submodularity. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 1560–1568, 2017.

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

[KZK18] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In *International Conference on Machine Learning*, pages 2549–2558, 2018.

Copyright © 2019
Copyright for this paper is retained by authors

[LMSV11] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA*, pages 85–94, 2011.

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

[MBNF+17] Slobodan Mitrovic, Ilija Bogunovic, Ashkan Norouzi-Fard, Jakub M Tarnawski, and Volkan Cevher. Streaming robust submodular maximization: A partitioned thresholding approach. In *Advances in Neural Information Processing Systems*, pages 4557–4566, 2017.

[MKK17] Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Deletion-robust submodular maximization: Data summarization with the right to be forgotten. In *International Conference on Machine Learning*, pages 2449–2458, 2017.

[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 26: 27th Annual Conference on Neural Information Processing Systems*, pages 2049–2057, 2013.

[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*, (STOC), pages 153–162. ACM, 2015.

[MZK16] Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In *Advances in Neural Information Processing Systems*, pages 3594–3602, 2016.

[SSS07] Ian Simon, Noah Snavely, and Steven M Seitz. Scene summarization for online image collections. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[SSSJ12] Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 754–763. ACM, 2012.

[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.

A Missing Analysis from Section 3

A.1 Proof of Lemma 3.3

LEMMA 3.3. *In each round of* THRESHOLD-SAMPLING, *we have* $\mathbb{E}[I_1] \geq \mathbb{E}[I_2] \geq \cdots \geq \mathbb{E}[I_{|A|}]$.

Proof. Let $(n)_t = n(n-1)\dots(n-t+1)$ denote the falling factorial. By summing over all truncated permutations of the elements $x_1, x_2, \dots, x_m \in A$, we have

$$\begin{aligned}
&\mathbb{E}[I_{t+1}] \\
&= \frac{1}{(m)_{t+1}} \sum_{x_1,\dots,x_{t+1}} \mathbb{1}[\Delta(x_{t+1}, S \cup \{x_1,\dots,x_{t-1},x_t\}) \geq \tau] \\
&\leq \frac{1}{(m)_{t+1}} \sum_{x_1,\dots,x_{t+1}} \mathbb{1}[\Delta(x_{t+1}, S \cup \{x_1,\dots,x_{t-1}\}) \geq \tau] \\
&= \frac{m-t}{(m)_{t+1}} \sum_{x_1,\dots,x_{t-1},x_{t+1}} \mathbb{1}[\Delta(x_{t+1}, S \cup \{x_1,\dots,x_{t-1}\}) \geq \tau] \\
&= \frac{1}{(m)_t} \sum_{x_1,\dots,x_{t-1},x_t} \mathbb{1}[\Delta(x_t, S \cup \{x_1,\dots,x_{t-1}\}) \geq \tau] \\
&= \mathbb{E}[I_t].
\end{aligned}$$

Observe that the second to last equality is simply a change of variables. □

A.2 Analysis of Reduced-Mean

LEMMA A.1. (CHERNOFF BOUNDS, [BS06]) *Suppose* $X_1, \dots, X_n$ *are binary random variables such that* $\Pr(X_i = 1) = p_i$. *Let* $\mu = \sum_{i=1}^n p_i$ *and* $X = \sum_{i=1}^n X_i$. *Then for any* $a > 0$, *we have*

$$\Pr(X - \mu \geq a) \leq e^{-a \min\left(\frac{1}{5}, \frac{a}{4\mu}\right)}.$$

Moreover, for any $a > 0$, *we have*

$$\Pr(X - \mu \leq -a) \leq e^{-\frac{a^2}{2\mu}}.$$

LEMMA 3.2. *For any Bernoulli distribution* $\mathcal{D}$, REDUCED-MEAN *uses* $O(\log(\delta^{-1})/\varepsilon^2)$ *samples to correctly report one of the following properties with probability at least* $1 - \delta$*:*

1. *If the output is* `true`, *then the mean of* $\mathcal{D}$ *is* $\mu \leq 1 - \varepsilon$.
2. *If the output is* `false`, *then the mean of* $\mathcal{D}$ *is* $\mu \geq 1 - 2\varepsilon$.

Proof. By construction the number of samples used is $m = 16\lceil \log(2/\delta)/\varepsilon^2 \rceil$. To show the correctness of REDUCED-MEAN, it suffices to prove that $\Pr(|\overline{\mu} - \mu| \geq \varepsilon/2) \leq \delta$. Letting $X = \sum_{i=1}^m X_i$, this is equivalent to

$$\Pr\left(|X - m\mu| \geq \frac{\varepsilon m}{2}\right) \leq \delta.$$

Using the Chernoff bounds in Lemma A.1 and a union bound, for any $a > 0$ we have

$$\Pr(|X - m\mu| \geq a) \leq e^{-\frac{a^2}{2m\mu}} + e^{-a \min\left(\frac{1}{5}, \frac{a}{4m\mu}\right)}.$$

Copyright © 2019
Copyright for this paper is retained by authors

Let $a = \varepsilon m/2$ and consider the exponents of the two terms separately. Since $\mu \leq 1$, we bound the left term by

$$\frac{a^2}{2m\mu} = \frac{\varepsilon^2 m^2}{8m\mu} \geq \frac{\varepsilon^2}{8\mu} \cdot \frac{16\log(2/\delta)}{\varepsilon^2} \geq \log(2/\delta).$$

For the second term, first consider the case when $1/5 \leq a/(4m\mu)$. For any $\varepsilon \leq 1$, it follows that

$$a \min\left(\frac{1}{5}, \frac{a}{4m\mu}\right) = \frac{1}{5} \geq \frac{\varepsilon}{10} \cdot \frac{16\log(2/\delta)}{\varepsilon^2} \geq \log(2/\delta).$$

Otherwise, we have $a/(4m\mu) \leq 1/5$, and by previous analysis we have $a^2/(4m\mu) \geq \log(2\delta)$. Therefore, in all cases we have

$$\Pr\left(|X - m\mu| \geq \frac{\varepsilon m}{2}\right) \leq 2e^{-\log(2/\delta)} = \delta,$$

which completes the proof. □

B Missing Analysis from Section 5

B.1 Analysis of Binary-Search-Maximization

Corollary 5.1. *For any monotone, nonnegative submodular function f, the algorithm* Binary-Search-Maximization *outputs a subset $S \subseteq N$ with $|S| \leq k$ in $O(\log(n/\delta)/\varepsilon^2)$ adaptive rounds such that with probability at least $1-\delta$ the algorithm makes $O(n \log\log(k)/\varepsilon^3)$ oracle queries in expectation and $\mathbb{E}[f(S)] \geq (1 - 1/e - \varepsilon)\text{OPT}$.*

Proof. At the beginning of the algorithm, the interval $[L, U] = [\Delta^*, k\Delta^*]$ contains OPT by submodularity. In each step of the binary search we can choose $\tau \in [L, U]$ and use Threshold-Sampling to reduce the interval by some amount such that the updated interval contains OPT. This decision process is described in Section 5.1. Our goal is to run Exhaustive-Maximization on a smaller feasible interval with ratio $U/L = O(1/p)$ so that we can set $r = O(\log(1/p)/\varepsilon)$ instead of $O(\log(k)/\varepsilon)$. This objective stems from the fact that Exhaustive-Maximization grows $(1+\varepsilon)^i$-sized balls until the interval is covered to approximate τ^*. Therefore, at each step of the binary search we let

$$\tau = \operatorname*{arg\,min}_{\tau' \in [L,U]} \max\left\{\frac{2k\tau'}{L}, \frac{U}{pk\tau'}\right\},$$

by considering the worst ratio of both outcomes. Since one function is increasing in τ and the other is decreasing, we equate the two expressions to optimize τ, which gives us $\tau = \sqrt{UL/(2pk^2)}$. It follows that the ratio of the updated interval is at most $\sqrt{2U/(pL)}$.

To track the progress of the binary search, it is more convenient to analyze the logarithm of the ratio. Each step maps $\log(U/L) \mapsto (1/2)\log(U/L) + (1/2)\log(2/p)$, so after the i-th step the log of the ratio of the interval is at most

$$(1/2)^i \log(U/L) + \sum_{j=1}^{i} (1/2)^j \log(2/p) \leq (1/2)^i \log(U/L) + \log(2/p).$$

Letting m be the first step where $(1/2)^i \log(U/L) < 1$ and recalling that $U/L = k$, it follows that

$$m = \left\lfloor \frac{\log\log(k)}{\log(2)} \right\rfloor.$$

This means that after m steps of the imprecise binary search, the ratio of the remaining interval is at most $2e/p$. Therefore, if we run Exhaustive-Maximization on the preprocessed interval, it suffices to set $r = \lceil 2\log(2e/p)/\varepsilon \rceil = O(\log(1/p)/\varepsilon)$. Since each iteration of the binary search depends on the result of the last, we choose p such that the preprocessing adaptivity is $O(\log(n/\delta))$.

Observe that the adaptivity of Threshold-Sampling is actually $O(\log(n/\delta)/\log(1/p))$ by the proof of Lemma 3.1. Letting $p = 1/\log(k)$, it follows from the number of iterations m in the binary search of Binary-Search-Maximization that the adaptivity complexity of for preprocessing is

$$O\left(m \cdot \frac{\log(n/\delta)}{\log(1/p)}\right) = O\left(\log\log(k) \cdot \frac{\log(n/\delta)}{\log\log(k)}\right) = O(\log(n/\delta)).$$

Thus, the overall adaptivity of Binary-Search-Maximization is $O(\log(n/\delta)/\varepsilon^2)$ by Theorem 4.1.

Now we analyze the expected query complexity of Binary-Search-Maximization. For each call to Threshold-Sampling in the binary search, we expect to make $O(n/(1-p))$ oracle queries by Lemma 3.1. By our choice of $\hat{\delta}$ and a union bound, we can assume that all subroutines produce their guaranteed output. Therefore, the total expected query complexity for the binary search is

$$O\left(m \cdot \frac{n}{1-p}\right) = O\left(\log\log(k) \cdot \frac{n}{1 - \frac{1}{\log(k)}}\right) = O(n \log\log(k)).$$

Next, since the ratio of the updated interval $[U, L]$ after the binary search is $O(\log(k))$, it follows that by modifying the search for τ^* in Exhaustive-Maximization, the expected query complexity in this stage is $O\big(n \log\log(k)/\varepsilon^3\big)$ by Theorem 4.1. This term

Copyright © 2019
Copyright for this paper is retained by authors

dominates the query complexity of the binary search, so the result follows. Lastly, the approximation factor holds in expectation by Theorem 4.1 because the updated region contains τ^*. □

B.2 Analysis of Subsample-Maximization

LEMMA B.1. (CHEBYSHEV'S INEQUALITY) *Let $X_1, X_2, \ldots, X_n$ be independent random variables with $\mathbb{E}[X_i] = \mu_i$ and $\mathrm{Var}(X_i) = \sigma_i^2$. Then for any $a > 0$,*

$$\Pr\left(\left|\sum_{i=1}^n X_i - \sum_{i=1}^n \mu_i\right| \geq a\right) \leq \frac{1}{a^2}\sum_{i=1}^n \sigma_i^2.$$

LEMMA 5.1. *For any submodular function, sample each element in the ground set N independently with probability $1/\ell$. If the resulting subsampled set is N', denote the optimal solution in N' by OPT'. If Δ^* is an upper bound for the maximum marginal in N, then with probability at least $1-\delta$,*

$$\frac{1}{2}(\Delta^* + \mathrm{OPT}') \leq \mathrm{OPT} \leq \frac{2\ell}{\delta}(\Delta^* + \mathrm{OPT}').$$

Proof. Let $x_1, x_2, \ldots, x_k$ be the elements in S^* in lexicographic order. By summing the marginal gain for each element when they are added in lexicographic order, we have

$$f(S^*) = \sum_{x \in S^*} \Delta(x, \pi_x),$$

where π_x denotes the set of elements before x in the lexicographic order. Subsample the ground set N such that each element is included in the set N' independently with probability $1/\ell$, and let S' be the random set denoting the elements in S^* that remain after subsampling. It follows from submodularity that for any value that S' takes, we have

$$f(S') \geq \sum_{x \in S'} \Delta(x, \pi_x).$$

For each $x \in S^*$ define the random variable

$$Z_x = \begin{cases} \Delta(x, \pi_x) & \text{with probability } 1/\ell, \\ 0 & \text{otherwise.} \end{cases}$$

It follows from the mean and variance of a Bernoulli random variable that

$$\mathbb{E}[Z_x] = \Delta(x, \pi_x) \cdot \frac{1}{\ell}$$
$$\mathrm{Var}(Z_x) = \Delta(x, \pi_x)^2 \cdot \frac{1}{\ell}\left(1 - \frac{1}{\ell}\right).$$

Let $g(S^*)$ be the random variable

$$g(S^*) = \sum_{x \in S^*} Z_x,$$

which is always a lower bound for the optimal solution in N'. It follows that

$$\mathbb{E}[g(S^*)] = \frac{1}{\ell}\sum_{x \in S^*} \Delta(x, \pi_x) = \frac{1}{\ell} \cdot f(S^*) = \frac{\mathrm{OPT}}{\ell}.$$

Let OPT' denote the optimal solution in N'. Ultimately, we want to show that with probability at least $1-\delta$ we have

$$\frac{\ell\Delta^*}{\delta} + \ell \cdot \mathrm{OPT}' \geq \mathrm{OPT} \geq \mathrm{OPT}',$$

as this implies the lower and upper bounds

$$\frac{\ell}{\delta} \cdot (\Delta^* + \mathrm{OPT}') \geq \mathrm{OPT} \geq \frac{\mathrm{OPT}' + \Delta^*}{2}.$$

Consider the probability

$$\begin{aligned}
&\Pr(\ell\Delta^*/\delta + \mathrm{OPT}/2 + \ell g(S^*) \geq \mathrm{OPT}) \\
&\quad = \Pr(\ell g(S^*) - \mathrm{OPT} \geq -\ell\Delta^*/\delta - \mathrm{OPT}/2) \\
&\quad = \Pr(\mathrm{OPT} - \ell g(S^*) \leq \ell\Delta^*/\delta + \mathrm{OPT}/2) \\
&\quad = \Pr\left(\frac{\mathrm{OPT}}{\ell} - g(S^*) \leq \Delta^*/\delta + \frac{\mathrm{OPT}}{2\ell}\right).
\end{aligned}$$

Using Lemma B.1, the probability of the complementary event is

$$\begin{aligned}
&\Pr\left(\frac{\mathrm{OPT}}{\ell} - g(S^*) > \Delta^*/\delta + \frac{\mathrm{OPT}}{2\ell}\right) \\
&\leq \Pr\left(\frac{\mathrm{OPT}}{\ell} - g(S^*) \geq \Delta^*/\delta + \frac{\mathrm{OPT}}{2\ell}\right) \\
&\leq \Pr\left(\left|\frac{\mathrm{OPT}}{\ell} - g(S^*)\right| \geq \Delta^*/\delta + \frac{\mathrm{OPT}}{2\ell}\right) \\
&\leq \frac{1}{\left(\Delta^*/\delta + \frac{\mathrm{OPT}}{2\ell}\right)^2} \cdot \frac{1}{\ell}\left(1 - \frac{1}{\ell}\right)\sum_{x \in S^*} \Delta(x, \pi_x)^2 \\
&\leq \frac{1}{\left(\Delta^*/\delta + \frac{\mathrm{OPT}}{2\ell}\right)^2} \cdot \left(\frac{\ell - 1}{\ell^2}\right)\Delta^* \cdot \sum_{x \in S^*} \Delta(x, \pi_x) \\
&\leq \frac{4(\ell - 1)}{(2\ell\Delta^*/\delta + \mathrm{OPT})^2} \cdot \Delta^*\mathrm{OPT} \\
&\leq \frac{4\ell\Delta^*\mathrm{OPT}}{4\ell^2(\Delta^*)^2/\delta^2 + 4\ell\Delta^*\mathrm{OPT}/\delta + \mathrm{OPT}^2} \\
&\leq \frac{4\ell\Delta^*\mathrm{OPT}}{4\ell\Delta^*\mathrm{OPT}/\delta} \\
&= \delta.
\end{aligned}$$

Copyright © 2019
Copyright for this paper is retained by authors

Therefore, it follows that

$$\Pr(\ell\Delta^*/\delta + \mathrm{OPT}/2 + \ell g(S^*) \geq \mathrm{OPT}) \geq 1 - \delta.$$

Since in all instances we have $\mathrm{OPT}' \geq g(S^*)$ and $\mathrm{OPT} \geq \mathrm{OPT}'$, it follows that

$$\frac{\ell}{\delta}\cdot\Delta^* + \ell g(S^*) \geq \frac{\mathrm{OPT}}{2} \implies \frac{2\ell}{\delta}\left(\Delta^* + \mathrm{OPT}'\right) \geq \mathrm{OPT} \geq \frac{1}{2}\left(\Delta^* + \mathrm{OPT}'\right).$$

Therefore, if we query all marginals to compute Δ^* and then subsample by $1/\ell$, then with probability at least $1-\delta$ then we know that OPT lies within an interval of ratio $4\ell/\delta$. □

LEMMA 5.2. *For any monotone submodular function f, let $[L, U]$ be an interval containing* OPT *with $U/L = R$. For any ratio $R > 0$ and with probability at least $1-\delta$, we can compute a new feasible interval with ratio $(8/\delta)\log^3(R)$ such that:*

- *The number of adaptive rounds is $O\left(\frac{\log(n/\delta)}{\log\log(R)}\right)$.*
- *The number of queries needed is $O(n/\log(R))$.*

Proof. Subsample the ground set N with probability $1/\ell$ to get N'. We will choose the value of ℓ later as function of the ratio R. Let $i^* = \lceil\log(R)/\log(2)\rceil$ and set a smaller error probability $\hat{\delta} = \delta/(i^*+1)$. For a variable $p \in [0, 1)$ that we also set later, run THRESHOLD-SAMPLING$(f, k, \tau, 1-p, \hat{\delta})$ on N' in parallel for the values $\tau = L/k, 2L/k, 2^2L/k, \ldots, 2^{i^*}L/k$. For each call, we can determine if $\mathrm{OPT}' \leq 2k\tau$ or $pk\tau \leq \mathrm{OPT}'$ as explained in Section 5.1. There are three cases to consider:

- If we always have $\mathrm{OPT}' \leq 2k\tau$, then $\mathrm{OPT}' \leq 2L$.
- If we always have $\mathrm{OPT}' \geq pk\tau$, then $\mathrm{OPT}' \in [pU, U]$ since $\mathrm{OPT}' \leq \mathrm{OPT} \leq U$.
- Otherwise, find the least index i such that $\mathrm{OPT}' \leq 2k(2^i/k)$ and $pk(2^{i+1}/k) \leq \mathrm{OPT}'$. This implies that $\mathrm{OPT}' \in [p2^{i+1}, 2^{i+1}]$.

To analyze the first case, observe that with probability at least $1-\hat{\delta}$ we have

$$\frac{1}{2}(\Delta^* + L) \leq \mathrm{OPT} \leq \frac{2\ell}{\hat{\delta}}\left(\Delta^* + \mathrm{OPT}'\right) \leq \frac{2\ell}{\hat{\delta}}(2\Delta^* + 2L),$$

by Lemma 5.1. Therefore, we have a new interval containing OPT whose ratio is $4\ell/\hat{\delta}$. For the second case, it follows from the Lemma 5.1 and the case that

$$\frac{p}{2}(\Delta^* + U) \leq \mathrm{OPT} \leq \frac{2\ell}{\hat{\delta}}(\Delta^* + U).$$

Therefore, the ratio of the new feasible region is $4\ell/(\hat{\delta}p)$. Similarly, we achieve the same ratio in the third case. Therefore, in all cases the ratio R maps to a new ratio of size at most $8\ell/(\hat{\delta}p)$.

Now we assume $R \geq 100$ and consider the adaptivity and query complexity of this interval reduction procedure. The adaptivity is that of THRESHOLD-SAMPLING$(f, k, \tau, 1-p, \hat{\delta})$ because we try all values of τ in parallel. Therefore, by Lemma 3.1 the adaptivity complexity is

$$O\left(\frac{\log\left(\frac{n/\ell}{\hat{\delta}}\right)}{\log(1/p)}\right) = O\left(\frac{\log\left(\frac{n/\ell}{\delta}\right)}{\log(1/p)}\right).$$

Similarly, the expected number of queries is

$$O\left(i^*\cdot\frac{n/\ell}{1-p}\right) = O\left(\log(R)\cdot\frac{n/\ell}{1-p}\right).$$

Now we choose values for ℓ and p as a function of the ratio R so that by iterating this ratio reduction procedure until $R \leq 100$ the total query complexity is linear and the adaptivity complexity is logarithmic. Set $\ell = \log^2(R)$ and $p = 1/\log(R)$. It follows that the adaptivity complexity is

$$O\left(\frac{\log\left(\frac{n/\log^2(R)}{\hat{\delta}}\right)}{\log\log(R)}\right) = O\left(\frac{\log(n/\delta)}{\log\log(R)}\right),$$

and the query complexity is

$$O\left(\log(R)\cdot\frac{\frac{n}{\log^2(R)}}{1-\frac{1}{\log(R)}}\right) = O\left(\frac{n}{\log(R)}\right).$$

This completes the proof. □

LEMMA 5.3. *For any monotone submodular function f and constant $0 < \delta \leq 1$, with probability at least $1-\delta$ the algorithm* SUBSAMPLE-PREPROCESSING *returns an interval containing* OPT *with ratio $O(1/\delta^2)$ in $O(\log(n/\delta))$ adaptive rounds and uses $O(n)$ queries in expectation.*

Proof. Assume $\delta > 0$ is a constant failure probability over all stages of the preprocessing procedure. To guarantee an overall failure probability of at most δ,

Copyright © 2019
Copyright for this paper is retained by authors

we use the probability $\delta_R = \delta/(2\log(R))$ depending on the current ratio R. Let

$$R^* \stackrel{\text{def}}{=} \frac{2 \times 10^6}{\delta^2}$$

denote a lower bound for the ratio reduction procedure, and define the map

$$h(R) \stackrel{\text{def}}{=} (8/\delta_R)\log^3(R) = (16/\delta)\log^4(R).$$

For every $R \geq R^*$ we have $h(R) \leq R/2$, so reducing the ratio of the current feasible interval is sensible as long as $R \geq R^*$. To compute R^* first consider the equation $x/2 = 16\log^4(x)$ and observe that its largest positive real solution x^* is

$$x^* = e^{-4W_{-1}\left(-\frac{1}{8\cdot(2)^{1/4}}\right)} \leq 1.24015 \times 10^6,$$

where $W_k(z)$ is the analytic continuation of the product log function. From here, we observe that for any $\delta > 0$ and $x \geq x^*/\delta^2$, the inequality $x/2 \geq (16/\delta)\log^4(x)$ holds.

Now we consider the algorithm Subsample-Preprocessing that reduces the ratio of the feasible region to be upper bounded by R^*. Starting from the initial ratio $R_0 = k$, repeatedly apply the ratio reduction map and stop once the ratio falls below R^*. Formally, let $R_0 = k$ and then $R_i = h(R_{i-1})$ for $i \geq 1$ until $R_{m-1} \geq R^*$ and $R_m < R^*$. It follows that

$$R_m = h(R_{m-1}) \geq h(R^*) = \frac{16}{\delta}\log^4\left(\frac{2000000}{\delta^2}\right) \geq \log^4(2000000) \geq e^e.$$

By observing the number of adaptive rounds and number of queries in Lemma 5.2 for one ratio reduction from R to $h(R)$, it follows that if we upper bound

$$\sum_{i=0}^{m} \frac{1}{\log(R_i)} \leq \sum_{i=0}^{m} \frac{1}{\log\log(R_i)} \leq C_\delta$$

by a constant C_δ depending on δ for all initial values of $k \geq R^*$, then the preprocessing procedure reduces the ratio of the interval containing OPT to have ratio at most R^* using a total of $O(\log(n)/\delta)$ adaptive rounds and $O(n)$ queries in expectation. It can be verified that for $k \geq R^*$ and $0 \leq \delta \leq 1$,

$$\sum_{i=0}^{m} \frac{1}{\log\log(R_i)} \leq 2,$$

which proves the claim about the adaptivity and query complexities. Note also that by our choice of $\delta_R = \delta/(2\log(R))$, the subsampling preprocessing procedure fails with probability at most δ using a union bound and the fact $C_\delta \leq 2$ for $0 \leq \delta \leq 1$. □

C Missing Analysis from Section 6

C.1 Analysis of Adaptive-Greedy-Cover

Theorem C.1. *For any integer-valued, nonnegative, monotone submodular function f, the algorithm* Adaptive-Greedy-Cover *outputs a set $S \subseteq N$ with $f(S) \geq L$ in $O(\log(n\log(L))\log(L))$ adaptive rounds such that $\mathbb{E}[|S|] = O(\log(L)|S^*|)$.*

Proof. Assume that $\Delta^* < L$, since if we have $\Delta^* \geq L$ then the algorithm can trivially output the singleton with largest marginal value. Next, observe that we have $f(S) \geq L$ upon termination since we assumed f is integer-valued and the threshold can eventually reach $\tau < 1$. To bound the adaptivity complexity, observe that Threshold-Sampling-For-Cover runs in $O(\log(n\log(\Delta^*)))$ adaptive rounds by Corollary 6.1 and our choice of ε and δ in Lines 1 and 3 of Adaptive-Greedy-Cover. Furthermore, Adaptive-Greedy-Cover calls this subroutine $m = O(\log(\Delta^*))$ times, so the adaptivity complexity does not exceed $O(\log(n\log(L))\log(L))$ by our initial assumption.

For the approximation factor of Adaptive-Greedy-Cover, first assume that the output guarantees of Threshold-Sampling-For-Cover hold over all calls with probability at least $1-1/n$ by our choice of δ and a union bound. We begin by mirroring the analysis of the approximation factor for submodular maximization in Theorem 4.1 to setup our framework. Recall the value and marginal gain of the averaged process $\hat{f}(S_i)$ and $\hat{\Delta}(X_i, S_{i-1})$ defined in Section 4, and let $k^* = |S^*|$ denote the size of the optimal set S^*. Call the subsets that are added to S during the course of the algorithm $T_1, T_2, \ldots, T_m$ and let the remaining gap be $\delta_i = f(S^*) - \hat{f}(S_i)$. Following the proofs of Lemmas 4.1 and 4.2, and noticing that $f(S^*)/k^* \leq \Delta^*$ by submodularity, for all $i \geq 1$,

$$\mathbb{E}[\delta_i \mid H_{i-1}(h)] \leq \left(1 - \frac{(1-\varepsilon)^2}{k^*}\right) \cdot \mathbb{E}[\delta_{i-1} \mid H_{i-1}(h)].$$

While this expected inequality holds when conditioned on histories $h = (T_1, T_2, \ldots, T_{\ell-1})$, we show how to iterate it to show that the gap $L - f(S)$ decreases geometrically with constant probability.

We start by showing that the size of every subset T_i is at most $|T_i| \leq k^*/(1-\varepsilon)^2 = 4k^*$. Since f is a monotone submodular function and τ is reduced by a factor of $(1-\varepsilon)$ each time Threshold-Sampling-For-Cover is called starting from $\tau = \Delta^*$, it follows from Property 2 of Corollary 6.1 that

$$f(S^*) \leq f(S) + \sum_{x \in S^*} \Delta(x, S) \leq f(S) + k^* \cdot \frac{\tau}{1-\varepsilon}.$$

Copyright © 2019
Copyright for this paper is retained by authors

It follows from Line 13 in Threshold-Sampling-For-Cover that an upper bound for $|T_i|$ is

$$|T_i| \leq \left\lfloor \frac{L - f(S)}{(1-\varepsilon)\tau} \right\rfloor \leq \frac{f(S^*) - f(S)}{(1-\varepsilon)\tau} \leq \frac{k^*}{(1-\varepsilon)^2} = 4k^*.$$

Now we consider the progress of reducing the gap $L - f(S)$ after adding blocks of sets T_i. Define the first block $B_1 = T_1 \cup T_2 \cup \cdots \cup T_\ell$ such that $t_1 + t_2 + \cdots + t_\ell \geq 4k^*$ for the least possible value of ℓ. Similarly, define the blocks $B_2, B_3, \ldots$ to be the union of the sets T_i after the previous block such that the cardinality first exceeds $4k^*$. Because $|T_i| \leq 4k^*$, we have the upper bound $|B_i| \leq 8k^*$, which we use to ensure that the algorithm processes sufficiently many blocks. Since we analyze the algorithm by blocks, it is convenient to let $S_{B_i} = \bigcup_{j=1}^{i} B_j$ denote the union of the first i blocks. Lastly, observe that $\Delta(B_i, S_{B_{i-1}}) \leq 4(f(S^*) - f(S_{B_{i-1}}))$ for all $i \geq 1$, because the addition of each block never exceeds the previous gap by a factor of more than $1/(1-\varepsilon)^2$ and $L \leq f(S^*)$.

By analyzing the algorithm with blocks of size $O(k^*)$, we show that the addition of each block independently reduces the current gap $L - f(S_{B_{i-1}})$ by a constant factor with probability $p \geq 0.05$. This allows us to analyze the expected output size $\mathbb{E}[|S|]$ via the negative binomial distribution. Using an analogous block indexing for the gap δ_i, observe that our earlier upper bound on $\mathbb{E}[\delta_i \mid H_{i-1}(h)]$ implies

$$\begin{aligned}\mathbb{E}\left[\delta_{B_i} \mid S_{B_{i-1}}\right] &\leq \left(1 - \frac{1}{4k^*}\right)^{4k^*} \cdot \mathbb{E}\left[\delta_{B_{i-1}} \mid S_{B_{i-1}}\right] \\ &\leq (1/e) \cdot \mathbb{E}\left[\delta_{B_{i-1}} \mid S_{B_{i-1}}\right].\end{aligned}$$

Since the blocks are unions of complete sets T_j, the averaged process $\hat{f}(S_{B_i})$ and true value $f(S_{B_i})$ always agree, conditioned on the previous state $S_{B_{i-1}}$. Unpacking the expected inequality above and using the lower bound $L \leq f(S^*)$, we have

$$\mathbb{E}\left[\Delta(B_i, S_{B_{i-1}}) \mid S_{B_{i-1}}\right] \geq (1 - 1/e)\mathbb{E}\left[\delta_{B_{i-1}} \mid S_{B_{i-1}}\right].$$

This means that the addition of each block B_i decreases the current gap $L - f(S_{B_{i-1}})$ by a constant factor in expectation. However, we can make a stronger claim since $\Delta(B_i, S_{B_{i-1}})$ is upper bounded.

Let X_i be the indicator random variable conditioned on $S_{B_{i-1}}$ such that

$$X_i = \begin{cases} 0 & \text{if } \Delta(B_i, S_{B_{i-1}}) < (1 - 2/e) \cdot \delta_{B_{i-1}}, \\ 1 & \text{otherwise.} \end{cases}$$

We claim that $X_i = 1$ with probability $p \geq 0.05$, for otherwise we would have

$$\begin{aligned}\mathbb{E}\left[\Delta(B_i, S_{B_{i-1}}) \mid S_{B_{i-1}}\right] & \\ < (1-p)(1-2/e) \cdot \mathbb{E}\left[\delta_{B_{i-1}} \mid S_{B_{i-1}}\right] & \\ + 4p \cdot \mathbb{E}\left[\delta_{B_{i-1}} \mid S_{B_{i-1}}\right] & \\ < (1 - 1/e) \cdot \mathbb{E}\left[\delta_{B_{i-1}} \mid S_{B_{i-1}}\right], &\end{aligned}$$

which is a contradiction. Therefore, for each block B_i we have

$$\Pr\left(\delta_{B_i} \geq (2/e) \cdot \delta_{B_{i-1}} \mid S_{B_{i-1}}\right) \geq 0.05.$$

In other words, with probability $p \geq 0.05$, the addition of each block independently decreases the remaining gap to $f(S^*)$ by a constant factor.

Thus, if after the addition of ℓ blocks there are $a = \lceil \log(L)/\log(e/2) \rceil$ events such that $X_i = 1$, then the current gap to $f(S^*)$ satisfies

$$\delta_{B_\ell} \leq (2/e)^a \cdot \delta_{B_0} \leq \frac{1}{L} \cdot \delta_{B_0}.$$

By the definition of δ_{B_i} and the assumption that f is nonnegative, this implies that

$$\begin{aligned}& f(S^*) - f(S_{B_\ell}) \leq \frac{1}{L} \cdot f(S^*) \\ & \qquad \implies L\left(1 - \frac{1}{L}\right) = L - 1 \leq f(S_{B_\ell}).\end{aligned}$$

Because we assumed that f is integer-valued, the algorithm then reaches the value lower bound L after the addition of the next item.

It follows that we can upper bound $\mathbb{E}[|S|]$ by the expected number of blocks needed to have a successful events plus one more block to ensure that we exceed the target value L (conditioned on all calls to Threshold-Sampling-For-Cover succeeding, which by our choice of δ happens with probability at least $1 - 1/n$). Since each block has at most $8k^*$ elements, noticing that this stopping criterion is given by the negative binomial distribution yields

$$\begin{aligned}\mathbb{E}[|S|] &\leq 8k^*\left(1 + \sum_{\ell=0}^{\infty}(\ell + a)\binom{\ell + a - 1}{\ell}(1-p)^\ell p^a\right) \\ &= 8k^*\left(1 + a + \sum_{\ell=0}^{\infty}\ell\binom{\ell + a - 1}{\ell}(1-p)^\ell p^a\right) \\ &= 8k^*\left(1 + a + \frac{(1-p)a}{p}\right) \\ &\leq 8k^*(20a + 1).\end{aligned}$$

Here we use the fact that the expected value of a negative binomial distribution parameterized by a successes and failure probability $1 - p$ is $(1-p)a/p$. Since

Copyright © 2019
Copyright for this paper is retained by authors

$a = O(\log(L))$, it follows that we have the conditional expectation $\mathbb{E}[|S|] = O(k^* \log(L))$ with probability at least $1-1/n$. Conditioned on the algorithm failing (with probability at most $1/n$), we have $|S| \leq n$. Thus, in total we have

$$\begin{aligned}\mathbb{E}[|S|] &= (1 - 1/n) \cdot O(k^* a) + (1/n) \cdot n \\ &= O(k^* \log(L)),\end{aligned}$$

as desired. This completes the proof of the expected approximation factor. □

Copyright © 2019
Copyright for this paper is retained by authors