Non-monotone Submodular Maximization with Nearly Optimal Adaptivity and Query Complexity

Matthew Fahrbach [1] **Vahab Mirrokni** [2] **Morteza Zadimoghaddam** [2]

Abstract

Submodular maximization is a general optimization problem with a wide range of applications in machine learning (e.g., active learning, clustering, and feature selection). In large-scale optimization, the parallel running time of an algorithm is governed by its adaptivity, which measures the number of sequential rounds needed if the algorithm can execute polynomially-many independent oracle queries in parallel. While low adaptivity is ideal, it is not sufficient for an algorithm to be efficient in practice—there are many applications of distributed submodular optimization where the number of function evaluations becomes prohibitively expensive. Motivated by these applications, we study the adaptivity and query complexity of submodular maximization. In this paper, we give the first constant-factor approximation algorithm for maximizing a non-monotone submodular function subject to a cardinality constraint k that runs in $O(\log(n))$ adaptive rounds and makes $O(n \log(k))$ oracle queries in expectation. In our empirical study, we use three real-world applications to compare our algorithm with several benchmarks for non-monotone submodular maximization. The results demonstrate that our algorithm finds competitive solutions using significantly fewer rounds and queries.

1. Introduction

Submodular set functions are a powerful tool for modeling real-world problems because they naturally exhibit the property of diminishing returns. Several well-known examples of submodular functions include graph cuts, entropy-based clustering, coverage functions, and mutual information. As a result, submodular functions have been increasingly used in applications of machine learning such as data summarization (Simon et al., 2007; Sipos et al., 2012; Tschiatschek et al., 2014), feature selection (Das & Kempe, 2008; Khanna et al., 2017), and recommendation systems (El-Arini & Guestrin, 2011). While some of these applications involve maximizing monotone submodular functions, the more general problem of non-monotone submodular maximization has also been used extensively (Feige et al., 2011; Buchbinder et al., 2014; Mirzasoleiman et al., 2016; Balkanski et al., 2018; Norouzi-Fard et al., 2018). Some specific applications of non-monotone submodular maximization include image summarization and movie recommendation (Mirzasoleiman et al., 2016), and revenue maximization in viral marketing (Hartline et al., 2008). Two compelling uses of non-monotone submodular maximization algorithms are:

- Optimizing objectives that are a monotone submodular function minus a linear cost function that penalizes the addition of more elements to the set (e.g., the coverage and diversity trade-off). This appears in facility location problems where opening centers is expensive and in exemplar-based clustering (Dueck & Frey, 2007).
- Expressing learning problems such as feature selection using weakly submodular functions (Das & Kempe, 2008; Khanna et al., 2017; Elenberg et al., 2018; Qian & Singer, 2019). One possible source of non-monotonicity in this context is overfitting to training data by selecting too many representative features (e.g., Section 1.6 and Corollary 3.19 in (Mohri et al., 2018)). Although most of these learning problems have not yet been rigorously modeled as non-monotone submodular functions, there has been a recent surge of interest and a substantial amount of momentum in this direction.

The literature on submodular optimization typically assumes access to an oracle that evaluates the submodular function. In practice, however, oracle queries may take a long time to process. For example, the log-determinant of submatrices of a positive semi-definite matrix is a submodular function that is notoriously expensive to compute (Kazemi et al., 2018). Therefore, our goal when designing distributed algorithms is to minimize the number of rounds where the algorithm communicates with the oracle. This motivates the notion of

[1]Georgia Institute of Technology [2]Google. Correspondence to: Matthew Fahrbach <matthew.fahrbach@gatech.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

the *adaptivity complexity* of submodular optimization, first investigated in (Balkanski & Singer, 2018). In this model of computation, the algorithm can ask polynomially-many independent oracle queries all together in each round.

In a wide range of machine learning optimization problems, the objective functions can only be estimated through oracle access to the function. In many instances, these oracle evaluations are a new time-consuming optimization problem that we treat as a black box (e.g., hyperparameter optimization). Since our goal is to optimize the objective function using as few rounds of interaction with the oracle as possible, insights and algorithms developed in this adaptivity complexity framework can have a deep impact on distributed computing for machine learning applications in practice. Further motivation for the importance of this computational model is given in (Balkanski & Singer, 2018).

While the number of adaptive rounds is an important quantity to minimize, the computational complexity of evaluating oracle queries also motivates the design of algorithms that are efficient in terms of the total number of oracle queries. An algorithm typically needs to make at least a constant number of queries per element in the ground set to achieve a constant-factor approximation. In this paper, we study the adaptivity complexity and the total number of oracle queries that are needed to guarantee a constant-factor approximation when maximizing a non-monotone submodular function.

Results and Techniques. Our main result is a distributed algorithm for maximizing a non-monotone submodular function subject to a cardinality constraint k that achieves an expected $(0.039 - \varepsilon)$-approximation in $O(\log(n)/\varepsilon)$ adaptive rounds using $O(n \log(k)/\varepsilon^2)$ expected function evaluation queries. To the best of our knowledge, this is the first constant-factor approximation algorithm with nearly optimal adaptivity for the general problem of maximizing non-monotone submodular functions. The adaptivity complexity of our algorithm is optimal up to a $\Theta(\log\log(n))$ factor by the lower bound in (Balkanski & Singer, 2018).

The building blocks of our algorithm are the THRESHOLD-SAMPLING subroutine in (Fahrbach et al., 2019), which returns a subset of high-valued elements in $O(\log(n)/\varepsilon)$ adaptive rounds, and the unconstrained submodular maximization algorithm in (Chen et al., 2018) that gives a $(1/2 - \varepsilon)$-approximation in $O(\log(1/\varepsilon)/\varepsilon)$ adaptive rounds. We modify THRESHOLD-SAMPLING to terminate early if its pool of candidate elements becomes too small, which ensures that each element is not chosen with at least constant probability. This property has been shown to be useful for obtaining constant-factor approximations for non-monotone submodular function maximization (Buchbinder et al., 2014). Next, we run unconstrained maximization on the remaining set of high-valued candidates if its size is at most $3k$, downsample accordingly, and output the better of the two solutions. Our analysis shows how to optimize the constant parameters to balance between these two behaviors. Last, since THRESHOLD-SAMPLING requires an input close to OPT/k, we find an interval containing OPT, try logarithmically-many input thresholds in parallel, and return the solution with maximum value. We note that improving the bounds for OPT via low-adaptivity preprocessing can reduce the total query complexity as shown in (Fahrbach et al., 2019).

Related Works. Submodular maximization has garnered a significant amount of attention in the distributed and streaming literature because of its role in large-scale data mining (Lattanzi et al., 2011; Mirzasoleiman et al., 2013; Badanidiyuru et al., 2014; Kumar et al., 2015; Mirrokni & Zadimoghaddam, 2015; Barbosa et al., 2015; 2016; Fahrbach et al., 2018; Liu & Vondrak, 2018). However, in many distributed models (e.g., the Massively Parallel Computation model), round complexity often captures a different notion than adaptivity complexity. For example, a constant-factor approximation is achievable in two rounds of computation (Mirrokni & Zadimoghaddam, 2015), but it is impossible to compute a constant-factor approximation in $O(\log(n)/\log\log(n))$ adaptive rounds (Balkanski & Singer, 2018). Since adaptivity measures the communication complexity with a function evaluation oracle, a round in most distributed models can have arbitrarily high adaptivity.

The first set of related works with low adaptivity focus on maximizing *monotone* submodular functions subject to a cardinality constraint k. In (Balkanski & Singer, 2018), the authors show that a $(1/3 - \varepsilon)$-approximation is achievable in $O(\log(n))$ rounds. In terms of parallel running time, this is exponentially faster than the celebrated greedy algorithm which gives a $(1 - 1/e)$-approximation in $O(k)$ rounds (Nemhauser et al., 1978). Subsequently, (Balkanski et al., 2019; Ene & Nguyen, 2019; Fahrbach et al., 2019) independently designed $(1 - 1/e - \varepsilon)$-approximation algorithms with $O(\log(n))$ adaptivity. These works also show that only $O(n)$ oracle queries are needed in expectation. Recent works have also investigated the adaptivity of the multilinear relaxation of monotone submodular functions subject to packing constraints (Chekuri & Quanrud, 2019) and the submodular cover problem (Agarwal et al., 2019).

While the general problem of maximizing a (not necessarily monotone) submodular function has been studied extensively (Lee et al., 2010; Feige et al., 2011; Gharan & Vondrák, 2011; Buchbinder et al., 2014), noticeably less progress has been made. For example, the best achievable approximation for the centralized maximization problem is unknown but in the range $[0.385, 0.491]$ (Buchbinder & Feldman, 2016; Gharan & Vondrák, 2011). However, some progress has been made for the adaptive complexity of this problem, all which has been done independently and concurrently with an earlier version of this paper. Recently,

Table 1. Independent and concurrent works for low-adaptivity non-monotone submodular maximization subject to a cardinality constraint.

ALGORITHM	APPROXIMATION	ADAPTIVITY	QUERIES
(BUCHBINDER ET AL., 2016)	$1/e - \varepsilon$	$O(k)$	$O(n)$
(BALKANSKI ET AL., 2018)	$1/(2e) - \varepsilon$	$O(\log^2(n))$	$O(\text{OPT}^2 n \log^2(n) \log(k))$
(CHEKURI & QUANRUD, 2018)	$3 - 2\sqrt{2} - \varepsilon$	$O(\log^2(n))$	$O(nk^4 \log^2(n))$
(ENE ET AL., 2018B)	$1/e - \varepsilon$	$O(\log^2(n))$	$O(nk^2 \log^2(n))$
THIS PAPER	$0.039 - \varepsilon$	$O(\log(n))$	$O(n \log(k))$

(Balkanski et al., 2018) designed a parallel algorithm for non-monotone submodular maximization subject to a cardinality constraint that gives a $(1/(2e) - \varepsilon)$-approximation in $O(\log^2(n))$ adaptive rounds. Their algorithm estimates the expected marginal gain of random subsets, and therefore the number of function evaluations it needs to achieve provable guarantees is $O(\text{OPT}^2 n \log^2(n) \log(k))$. We acknowledge that the query complexity can likely be improved via normalization or estimating an indicator random variable instead. The works of (Chekuri & Quanrud, 2018; Ene et al., 2018b) give constant-factor approximation algorithms with $O(\log^2(n))$ adaptivity for maximizing non-monotone submodular functions subject to matroid constraints. Their approaches use multilinear extensions and thus require $\Omega(nk^2 \log^2(n))$ function evaluations to simulate an oracle for ∇f with high enough accuracy. There have also been significant advancements in low-adaptivity algorithms for the problem of unconstrained submodular maximization (Chen et al., 2018; Ene et al., 2018a).

2. Preliminaries

For any set function $f : 2^N \to \mathbb{R}_{\geq 0}$ and subsets $S, T \subseteq N$, let $\Delta(T, S) \stackrel{\text{def}}{=} f(S \cup T) - f(S)$ denote the *marginal gain* of f at T with respect to S. We refer to N as the ground set and let $|N| = n$. A set function $f : 2^N \to \mathbb{R}_{\geq 0}$ is *submodular* if for all $S \subseteq T \subseteq N$ and any $x \in N \setminus T$ we have $\Delta(x, S) \geq \Delta(x, T)$, where the marginal gain notation is overloaded for singletons. A set function is *monotone* if for all subsets $S \subseteq T \subseteq N$ we have $f(S) \leq f(T)$. In this paper, we investigate distributed algorithms for maximizing submodular functions subject to a cardinality constraint, including those that are *non-monotone*. 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 every 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 needed such that in each round the algorithm makes $O(\text{poly}(n))$ independent queries to the evaluation oracle. Queries in a given round may depend on the answers of queries from previous rounds but not the current round. We measure the parallel running time of an algorithm by its adaptivity.

One of the inspirations for our algorithm is the following lemma, which is remarkably useful for achieving a constant-factor approximation for general submodular functions.

Lemma 2.1. *(Buchbinder et al., 2014) Let $f : 2^N \to \mathbb{R}_{\geq 0}$ be submodular. Denote by $A(p)$ a random subset of A where each element appears with probability at most p (not necessarily independently). Then $\mathbb{E}[f(A(p))] \geq (1 - p) f(\emptyset)$.*

In our case, if S is the output of the algorithm and the probability of any element appearing in S is bounded away from 1, we can analyze the submodular function $g : 2^N \to \mathbb{R}_{\geq 0}$ defined by $g(S) = f(S \cup S^*)$ to lower bound $\mathbb{E}[f(S \cup S^*)]$ in terms of $\text{OPT} = f(S^*)$ since $g(\emptyset) = f(S^*)$.

2.1. The THRESHOLD-SAMPLING Algorithm

We start with a high-level description of the THRESHOLD-SAMPLING algorithm in (Fahrbach et al., 2019), which after a slight modification is the main subroutine of our non-monotone maximization algorithm. For an input threshold τ, THRESHOLD-SAMPLING iteratively builds a solution set S over $O(\log(n)/\varepsilon)$ adaptive rounds and maintains a set of unchosen candidate elements A. Initially, the solution set is empty and all elements are candidates (i.e., $S = \emptyset$ and $A = N$). In each round, the algorithm starts by discarding elements in A whose marginal gain to the current solution S is less than the threshold τ. Then the algorithm efficiently finds the largest cardinality t^* such that for $T \sim \mathcal{U}(A, t^*)$ uniformly at random we have $\mathbb{E}[\Delta(T, S)/|T|] \geq (1 - \varepsilon)\tau$. At the end of a round, the algorithm samples $T \sim \mathcal{U}(A, t^*)$ and updates the current solution to be $S \cup T$.

The random choice of T in THRESHOLD-SAMPLING 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 elements that the algorithm considers in each round decreases geometrically in expectation. It follows that $O(\log(n)/\varepsilon)$ rounds suffice to guarantee that when the algorithm terminates, we either have $|S| = k$ or the marginal gains of all the elements are below the threshold.

Before presenting THRESHOLD-SAMPLING, we define the distribution $\mathcal{D}_t$ from which THRESHOLD-SAMPLING samples when estimating the maximum cardinality t^*. Sampling from this Bernoulli distribution can be simulated with two calls to the evaluation oracle.

Definition 2.2. 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 $I_t = \mathbb{1}[\Delta(x, S \cup T) \geq \tau]$.

We can view $\mathbb{E}[I_t]$ 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 random permutation.

Algorithm 1 THRESHOLD-SAMPLING

Input: oracle for $f : 2^N \to \mathbb{R}_{\geq 0}$, constraint k, threshold τ, error ε, failure probability δ

Set smaller error $\hat{\varepsilon} \leftarrow \varepsilon/3$
Set $r \leftarrow \lceil \log_{(1-\hat{\varepsilon})^{-1}}(2n/\delta) \rceil$, $m \leftarrow \lceil \log(k)/\hat{\varepsilon} \rceil$
Set smaller failure probability $\hat{\delta} \leftarrow \delta/(2r(m+1))$
Initialize $S \leftarrow \emptyset$, $A \leftarrow N$
for r rounds **do**
    Filter $A \leftarrow \{x \in A : \Delta(x, S) \geq \tau\}$
    if $|A| = 0$ **then**
        break
    for $i = 0$ to m **do**
        Set $t \leftarrow \min\{\lfloor (1+\hat{\varepsilon})^i \rfloor, |A|\}$
        Set $\ell \leftarrow 16\lceil \log(2/\hat{\delta})/\hat{\varepsilon}^2 \rceil$
        Sample $X_1, X_2, \ldots, X_\ell \sim \mathcal{D}_t$
        Set $\overline{\mu} \leftarrow \frac{1}{\ell}\sum_{j=1}^{\ell} X_j$
        if $\overline{\mu} \leq 1 - 1.5\hat{\varepsilon}$ **then**
            break
    Sample $T \sim \mathcal{U}(A, \min\{t, k - |S|\})$
    Update $S \leftarrow S \cup T$
    if $|S| = k$ **then**
        break
return (S, A)

Lemma 2.3. *(Fahrbach et al., 2019) 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 marginal $\mathbb{E}[f(S)/|S|] \geq (1-\varepsilon)\tau$.*
3. *If $|S| < k$, then $\Delta(x, S) < \tau$ for all $x \in N$.*

2.2. Unconstrained Submodular Maximization

The second subroutine in our non-monotone maximization algorithm is a constant-approximation algorithm for unconstrained submodular maximization that runs in a constant number of rounds depending on ε. While the focus of this paper is submodular maximization subject to a cardinality constraint, we show how calling UNCONSTRAINED-MAX on a new ground set $A \subseteq N$ of size $O(k)$ can be used with (Buchbinder et al., 2014) to achieve a constant-approximation for the constrained maximization problem.

Lemma 2.4. *(Feige et al., 2011) For any nonnegative submodular function f, denote the solution to the unconstrained maximization problem by $\mathrm{OPT} = \max_{S \subseteq N} f(S)$. If R is a uniformly random subset of S, then $\mathbb{E}[f(R)] \geq (1/4)\mathrm{OPT}$.*

The guarantees for the UNCONSTRAINED-MAX algorithm in Lemma 2.5 are standard consequences of Lemma 2.4.

Algorithm 2 UNCONSTRAINED-MAX

Input: oracle for $f : 2^N \to \mathbb{R}_{\geq 0}$, ground subset $A \subseteq N$, error ε, failure probability δ

Set iteration bound $t \leftarrow \lceil \log(1/\delta)/\log(1 + (4/3)\varepsilon) \rceil$
for $i = 1$ to t in parallel **do**
    Let R_i be a uniformly random subset of A
Set $S \leftarrow \arg\max_{X \in \{R_1, \ldots, R_t\}} f(X)$
return S

Lemma 2.5. *For any nonnegative submodular function f and subset $A \subseteq N$,* UNCONSTRAINED-MAX *outputs a set $S \subseteq A$ in one adaptive round using $O(\log(1/\delta)/\varepsilon)$ oracle queries such that with probability at least $1-\delta$ we have $f(S) \geq (1/4 - \varepsilon)\mathrm{OPT}_A$, where $\mathrm{OPT}_A = \max_{T \subseteq A} f(T)$.*

An essentially optimal algorithm for unconstrained submodular maximization was recently given in (Chen et al., 2018), which allows us to slightly improve the approximation factor of our non-monotone maximization algorithm.

Theorem 2.6. *(Chen et al., 2018) There is an algorithm that achieves a $(1/2-\varepsilon)$-approximation for unconstrained submodular maximization using $O(\log(1/\varepsilon)/\varepsilon)$ adaptive rounds and $O(n\log^3(1/\varepsilon)/\varepsilon^4)$ evaluation oracle queries.*

3. Non-monotone Submodular Maximization

In this section we show how to combine THRESHOLD-SAMPLING and UNCONSTRAINED-MAX to achieve the first constant-factor approximation algorithm for non-monotone submodular maximization subject to a cardinality constraint k that uses $O(\log(n))$ adaptive rounds. Moreover, this algorithm makes $O(n\log(k))$ expected oracle queries. While the approximation factor is only 0.039, we demonstrate that THRESHOLD-SAMPLING can readily be extended to non-monotone settings without increasing its adaptivity.

We start by describing ADAPTIVE-NONMONOTONE-MAX and the analysis of its approximation factor at a high level. One inspiration for this algorithm is Lemma 2.1, which allows us to lower bound the expected value of the returned set $\mathbb{E}[f(R)]$ by OPT as long as every element has at most a constant probability less than 1 of being in the output. With this property in mind, ADAPTIVE-NONMONOTONE-MAX starts by trying different thresholds in parallel, one of which is sufficiently close to $c_1\text{OPT}/k$. For each threshold, it runs THRESHOLD-SAMPLING modified to break if the number of candidates in A falls below c_3k. For all values of $c_3 > 1$, this guarantees that each element appears in S with probability at most $1/c_3$. In the event that THRESHOLD-SAMPLING breaks because $|A| < c_3k$, it then runs unconstrained submodular maximization on A and downsamples the solution so that it has cardinality at most k. In the end, the algorithm returns the set with maximum value over all thresholds. Our analysis shows how we optimize the constants c_1 and c_3 to balance the expected trade-offs between the two events and thus give the best approximation factor. We present the algorithm and its guarantees below.

Algorithm 3 ADAPTIVE-NONMONOTONE-MAX

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

1: Set smaller error $\hat{\varepsilon} \leftarrow \varepsilon/6$
2: Set $\Delta^* \leftarrow \max\{f(x) : x \in N\}$, $r \leftarrow \lceil 2\log(k)/\hat{\varepsilon}\rceil$
3: Set smaller failure probability $\hat{\delta} \leftarrow \delta/(2(r+1))$
4: Set optimized constants $c_1 \leftarrow 1/7, c_3 \leftarrow 3$
5: Initialize $R \leftarrow \emptyset$
6: **for** $i = 0$ to r in parallel **do**
7:     Set $\tau \leftarrow c_1(1+\hat{\varepsilon})^i\Delta^*/k$
8:     Set $(S, A) \leftarrow$ THRESHOLD-SAMPLING$(f, k, \tau, \hat{\varepsilon}, \hat{\delta})$ modified to break on Line 7 if $|A| < c_3k$
9:     Initialize $U \leftarrow \emptyset, U' \leftarrow \emptyset, U'' \leftarrow \emptyset$
10:     **if** $|A| < c_3k$ **then**
11:         Set $U \leftarrow$ UNCONSTRAINED-MAX$(f, A, \hat{\varepsilon}, \hat{\delta})$
12:         **if** $|U| > k$ **then**
13:             Sample $D \sim \mathcal{U}(U, k)$
14:             Update $U' \leftarrow D$
15:         **else**
16:             Update $U' \leftarrow U$
17:     Permute the elements of U' uniformly at random
18:     Set $U'' \leftarrow$ highest-valued prefix of the permutation
19:     Update $R \leftarrow \arg\max_{X \in \{R, S, U''\}} f(X)$
20: **return** R

Theorem 3.1. *For any nonnegative submodular function f,* ADAPTIVE-NONMONOTONE-MAX *outputs a set $R \subseteq N$ with $|R| \le k$ in $O(\log(n/\delta)/\varepsilon)$ adaptive rounds such that with probability at least $1-\delta$ it makes $O(n\log(k)/\varepsilon^2)$ queries in expectation and $\mathbb{E}[f(S)] \ge (0.026 - \varepsilon)$OPT.*

Since the quality of our approximation relies on the approximation factor of a low-adaptivity algorithm for unconstrained submodular maximization, we can use Theorem 2.6 instead of UNCONSTRAINED-MAX to improve our approximation without a loss in adaptivity or query complexity.

Theorem 3.2. *There is an algorithm for nonnegative submodular maximization subject to a cardinality constraint k that achieves a $(0.039 - \varepsilon)$-approximation in expectation using $O(\log(n)/\varepsilon)$ adaptive rounds and $O(n\log(k)/\varepsilon^2)$ expected queries to the evaluation oracle.*

3.1. Prerequisite Notation and Lemmas

We start by defining notation that is useful for analyzing THRESHOLD-SAMPLING as the subroutine progresses. Let $T_1, T_2, \dots, T_r$ be the sequences of randomly generated sets used to build the output set S. Similarly, let the corresponding sequences of partial solutions be $S_i = \bigcup_{j=1}^{i} T_j$ and candidate sets be $A_0, A_1, \dots, A_r$. To analyze the approximation factor of ADAPTIVE-NONMONOTONE-MAX, we consider a threshold τ sufficiently close to $\tau^* = \text{OPT}/k$ and then analyze the resulting sets S, U, U', and U''. Lastly, we use ALG as an alias for the final output set R.

Next, we present several simple lemmas that are helpful for analyzing the approximation factor. The following lemma is an equation in the proof of Lemma 2.3, and we use this lemma to show that the elements in any partial solution S_i have an average marginal gain exceeding the input threshold.

Lemma 3.3. *(Fahrbach et al., 2019) At each step $i \ge 0$ of* THRESHOLD-SAMPLING*, we have*

$$\mathbb{E}[\Delta(T_{i+1}, S_i)] \ge (1 - 2\hat{\varepsilon})\tau \cdot \mathbb{E}[|T_{i+1}|].$$

Corollary 3.4. *At each step of* THRESHOLD-SAMPLING *we have $\mathbb{E}[f(S_i)] \ge (1-2\hat{\varepsilon})\tau \cdot \mathbb{E}[|S_i|]$.*

The following lemmas allow us show that (1) every element has at least a constant probability of not appearing in the output set, and (2) that the quality of a solution of size greater than k degrades at worst by its downsampling rate. The first property is motivated by Lemma 2.1 and allows us to achieve a lower bound in terms of OPT in Lemma 3.9. The second property is useful for analyzing Line 13 of the ADAPTIVE-NONMONOTONE-MAX algorithm.

Lemma 3.5. *For any element $x \in N$, $\Pr(x \in S) \le 1/c_3$.*

Proof. Let X_i be an indicator random variable for the event $x \in T_i$. It follows that

$$\Pr(x \in S) = \sum_{i=1}^{r} \mathbb{E}[X_i] \le \sum_{i=1}^{r} \mathbb{E}\left[\frac{|T_i|}{|A_i|}\right] \le \frac{1}{c_3k}\sum_{i=1}^{r}\mathbb{E}[|T_i|]$$
$$= \frac{1}{c_3k}\cdot \mathbb{E}[|S|] \le \frac{1}{c_3k}\cdot k = \frac{1}{c_3}. \qquad \square$$

Lemma 3.6. *For any subset $S \subseteq N$ and $0 \le k \le |S|$, if $T \sim \mathcal{U}(S, k)$ then $\mathbb{E}[f(T)] \ge k/|S| \cdot f(S)$.*

We defer the proofs of Corollary 3.4 and Lemma 3.6 to the supplementary manuscript.

3.2. Analysis of the Approximation Factor

The main idea behind our analysis is to capture two different behaviors of ADAPTIVE-NONMONOTONE-MAX and balance the worst of the two outcomes by optimizing constants.

Definition 3.7. Let $A_{<c_3k}$ denote the event that the subroutine THRESHOLD-SAMPLING breaks because $|A| < c_3k$. Similarly, let $A_{\geq c_3k}$ denote the complementary event.

The following two key lemmas lower bound the expected solution in terms of OPT and $\Pr(A_{<c_3k})$. The goal is to average these inequalities so that the probability terms disappear, giving us with a lower bound only in terms of OPT.

Lemma 3.8. *For any τ such that $\tau \leq c_1\tau^* \leq \tau(1+\hat{\varepsilon})$, we have $\mathbb{E}[\text{ALG}] \geq \mathbb{E}[f(S)] \geq (1-\varepsilon)c_1 \Pr(A_{\geq c_3k}) \cdot \text{OPT}$.*

Proof. Observing that $\mathbb{E}[|S| \mid A_{\geq c_3k}] = k$, it follows from Corollary 3.4 and the law of total expectation that

$$\begin{aligned}\mathbb{E}[f(S)] &\geq (1-2\hat{\varepsilon})\tau \cdot \mathbb{E}[|S|] \\ &\geq (1-2\hat{\varepsilon})\tau \cdot \mathbb{E}[|S| \mid A_{\geq c_3k}] \cdot \Pr(A_{\geq c_3k}) \\ &= (1-2\hat{\varepsilon})\tau k \cdot \Pr(A_{\geq c_3k}) \\ &\geq (1-\varepsilon)c_1 \Pr(A_{\geq c_3k}) \cdot \text{OPT}.\end{aligned}$$

The result follows from the fact $\mathbb{E}[\text{ALG}] \geq \mathbb{E}[f(S)]$. □

The core of the analysis is devoted to proving the following lower bound and intricately uses the conditional expectation of nonnegative random variables.

Lemma 3.9. *Let α denote the approximation factor for an unconstrained submodular maximization algorithm. For any threshold τ such that $\tau \leq c_1\tau^* \leq \tau(1+\hat{\varepsilon})$, we have*

$$\mathbb{E}[\text{ALG}] \geq \frac{(1-\varepsilon)}{c_3\alpha^{-1}}\left[\frac{\text{OPT}\cdot\Pr(A_{<c_3k})}{(1-c_1)^{-1}} - \frac{\text{OPT}}{c_3} - 2\mathbb{E}[f(S)]\right].$$

Proof. For any pair of subsets $A, S \subseteq N$ returned by THRESHOLD-SAMPLING, we can partition the optimal S^* into $S_1^* = S^* \cap A$ and $S_2^* = S^* \setminus A$. Let U_A be the output of a call to UNCONSTRAINED-MAX. By Lemma 2.5, we have $f(U_A) \geq (\alpha - \hat{\varepsilon})f(S_1^*)$. Submodularity and the definition of A also imply that $f(S_2^* \cup S) \leq f(S) + k\tau$. Let $\text{Gap}(A, S) = \max\{f(S_2^*) - f(S_2^* \cup S), 0\}$. By subadditivity and the previous inequalities, it follows that

$$\begin{aligned}f(S^*) &\leq f(S_1^*) + f(S_2^*) - f(S_2^* \cup S) + f(S_2^* \cup S) \\ &\leq (\alpha-\hat{\varepsilon})^{-1}f(U_A) + \text{Gap}(A,S) + f(S) + k\tau. \quad (1)\end{aligned}$$

Using (1) and the assumption on τ, we have

$$f(U_A) \geq (\alpha-\hat{\varepsilon})((1-c_1)\text{OPT} - \text{Gap}(A,S) - f(S)). \quad (2)$$

Our next goal is to upper bound $\text{Gap}(A, S)$ as a function of S^* so that we have a bound that is independent of A. Specifically, we prove in the supplementary material that for all sets $A \subseteq N$, $f(S_2^*) - f(S_2^* \cup S) \leq f(S^*) - f(S^* \cup S)$. This is a consequence of submodularity. Therefore, we have $\text{Gap}(\emptyset, S) \leq f(S^*) - f(S^* \cup S) + f(S)$ by subadditivity since f is nonnegative.

Next, define a new submodular function $g : 2^N \to \mathbb{R}$ such that $g(S) = f(S^* \cup S)$, and consider a random set S returned by THRESHOLD-SAMPLING. Each element appears in S with probability at most $1/c_3$ by Lemma 3.5. Applying Lemma 2.1 to g gives us $\mathbb{E}[f(S \cup S^*)] \geq (1-1/c_3)f(S^*)$. It follows that

$$\begin{aligned}\mathbb{E}[\text{Gap}(\emptyset, S)] &\leq \mathbb{E}[f(S^*) - f(S^* \cup S) + f(S)] \\ &\leq (1/c_3)\text{OPT} + \mathbb{E}[f(S)]. \quad (3)\end{aligned}$$

Now we are prepared to give the lower bound for $\mathbb{E}[\text{ALG}]$ in terms of $\text{OPT} \cdot \Pr(A_{<c_3k})$. From our earlier analysis, if the algorithm calls UNCONSTRAINED-MAX, we can use the inequality (2) to lower bound $\mathbb{E}[f(U) \mid A_{<c_3k}]$. Since $\mathbb{E}[f(U)] \geq \mathbb{E}[f(U) \mid A_{<c_3k}] \cdot \Pr(A_{<c_3k})$, the claim follows from (3), the law of total expectation, and the nonnegativity of $\text{Gap}(\emptyset, S)$ and $f(S)$. Last, it is possible that the unconstrained solution U exceeds the cardinality constraint, but by construction $|U| \leq c_3k$. Therefore, it follows from Lemma 3.6 that $\mathbb{E}[f(U'')] \geq \mathbb{E}[f(U')] \geq (1/c_3)\mathbb{E}[f(U)]$, which gives us the desired lower bound for $\mathbb{E}[\text{ALG}]$. □

Equipped with these two complementary lower bounds, we can now prove our main results.

Proof of Theorem 3.1. First assume that all subroutines behave as desired with probability at least $1-\delta$ by our choice of $\hat{\delta}$ and a union bound. Since ADAPTIVE-NONMONOTONE-MAX necessarily tries a τ such that $\tau \leq c_1\tau^* \leq \tau(1+\hat{\varepsilon})$, the analysis that follows considers this particular threshold.

We start with the proof of the approximation factor. Suppose $\mathbb{E}[f(S)] > c_4\text{OPT}$ for a constant $c_4 \geq 0$ that we later optimize. This leads to a c_4-approximation for OPT. Otherwise, $\mathbb{E}[f(S)] \leq c_4\text{OPT}$, so it follows from Lemma 3.9 that

$$\mathbb{E}[\text{ALG}] \geq \frac{(1-\varepsilon)}{c_3\alpha^{-1}}\left[\frac{\Pr(A_{<c_3k})}{(1-c_1)^{-1}} - \frac{1}{c_3} - 2c_4\right]\text{OPT}. \quad (4)$$

Taking a weighted average of Lemma 3.8 and (4) gives us

$$\frac{\mathbb{E}[\text{ALG}]}{\text{OPT}} \geq \frac{(1-\varepsilon)}{(1+\beta)c_3\alpha^{-1}} \cdot \left[\frac{\beta c_1}{c_3^{-1}\alpha}\Pr(A_{\geq c_3k}) + \frac{\Pr(A_{<c_3k})}{(1-c_1)^{-1}} - \frac{1}{c_3} - 2c_4\right].$$

To bound the approximation factor, we solve the optimization problem

$$\max_{c_1,c_3,c_4,\beta} \min\left\{c_4, \frac{(1-\varepsilon)}{(1+\beta)c_3\alpha^{-1}}\left(1-c_1-\frac{1}{c_3}-2c_4\right)\right\}$$

subject to the constraint $\beta\alpha^{-1}c_1c_3 = 1 - c_1$, which effectively balances the two complementary probabilities.

Now we optimize the constants in the algorithm. The equality constraint implies that $c_1 = (1 + \beta\alpha^{-1}c_3)^{-1}$. Next, we set the two expressions in the maximin problem to be equal since one is increasing in c_4 and the other is decreasing, which implies that

$$c_4 = \frac{(1-\varepsilon)\big(1 - c_1 - c_3^{-1}\big)}{(1+\beta)c_3\alpha^{-1} + 2(1-\varepsilon)}.$$

Using the expressions above for c_4 and c_1, it follows that

$$\frac{\mathbb{E}[\text{ALG}]}{\text{OPT}} \geq \frac{(1-\varepsilon)\big(1 - (1+\beta c_3\alpha^{-1})^{-1} + c_3^{-1}\big)}{(1+\beta)c_3\alpha^{-1} + 2}. \quad (5)$$

Lemma 2.5 implies that $\alpha = 1/4$. Setting $c_3 = 3$, $\beta = 1/2$, it follows that $c_1 = 1/7$, which gives us an approximation factor of $0.026 - \varepsilon$ by (5).

The proof of the adaptivity and query complexities follow from Lemma 2.3 and Lemma 2.5 since all $O(\log(k)/\varepsilon)$ thresholds are run in parallel. This completes the analysis for the ADAPTIVE-NONMONOTONE-MAX algorithm. □

Proof of Theorem 3.2. The proof is analogous to the proof of Theorem 3.1 except that Theorem 2.6 implies $\alpha = 1/2$. Setting $c_3 = 3.556, \beta = 0.5664$, we have $c_1 = 0.198989$ and an approximation factor of $0.0395 - \varepsilon$ by (5). Running the same non-monotone maximization algorithm with failure probability $\delta = 1/n$ and error $\varepsilon/2$ proves the claim. □

4. Experiments

In this section, we evaluate ADAPTIVE-NONMONOTONE-MAX on three real-world applications introduced in (Mirzasoleiman et al., 2016). We compare our algorithm with several benchmarks for non-monotone submodular maximization and demonstrate that it consistently finds competitive solutions using significantly fewer rounds and queries. Our experiments build on those in (Balkanski et al., 2018), which plot function values at each round as the algorithms progress. Additionally, we include plots of $\max_{|S|\leq k} f(S)$ for different constraints k and plots of the cumulative number of queries an algorithm has used after each round. For algorithms that rely on a $(1 \pm \varepsilon)$-approximation of OPT, we run all guesses in parallel and record statistics for the approximation that maximizes the objective function. We defer the implementation details to the supplementary manuscript.

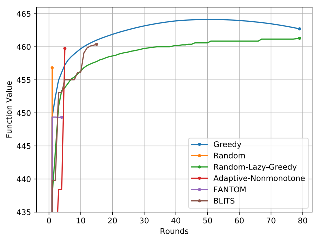
Next, we briefly describe the benchmark algorithms. The GREEDY algorithm builds a solution by choosing an element with the maximum positive marginal gain in each round. This requires $O(k)$ adaptive rounds and $O(nk)$ oracle queries, and it does not guarantee a constant approximation. The RANDOM algorithm randomly permutes the ground set and returns the highest-valued prefix of elements. It uses a constant number of rounds, makes $O(k)$ queries, and also fails to give a constant approximation. The RANDOM-LAZY-GREEDY-IMPROVED algorithm (Buchbinder et al., 2016) lazily builds a solution by randomly selecting one of the k elements with highest marginal gain in each round. This gives a $(1/e - \varepsilon)$-approximation in $O(k)$ adaptive rounds using $O(n)$ queries. The FANTOM algorithm (Mirzasoleiman et al., 2016) is similar to GREEDY and robust to intersecting matroid and knapsack constraints. For a cardinality constraint, it gives a $(1/6 - \varepsilon)$-approximation using $O(k)$ adaptive rounds and $O(nk)$ queries. The BLITS algorithm (Balkanski et al., 2018) constructs a solution by randomly choosing blocks of high-valued elements, giving a $(1/(2e) - \varepsilon)$-approximation in $O(\log^2(n))$ rounds. While BLITS is exponentially faster than the previous algorithms, it requires $O(\text{OPT}^2 n \log^2(n)\log(k))$ oracle queries.

Image Summarization. The goal of image summarization is to find a small, representative subset from a large collection of images that accurately describes the entire dataset. The quality of a summary is typically modeled by two contrasting requirements: coverage and diversity. Coverage measures the overall representation of the dataset, and diversity encourages succinctness by penalizing summaries that contain similar images. For a collection of images N, the objective function we use for image summarization is

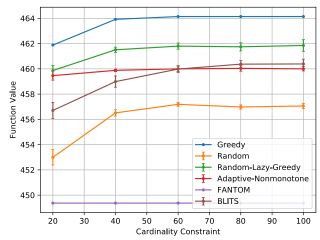
$$f(S) = \sum_{i\in N} \max_{j\in S} s_{i,j} - \frac{1}{|N|}\sum_{i\in S}\sum_{j\in S} s_{i,j},$$

where $s_{i,j}$ is the similarity between image i and image j. The trade-off between coverage and diversity naturally gives rise to non-monotone submodular functions. We perform our image summarization experiment on the CIFAR-10 test set (Krizhevsky & Hinton, 2009), which contains 10,000 32×32 color images. The image similarity $s_{i,j}$ is measured by the cosine similarity of the 3,072-dimensional pixel vectors for images i and j. Following (Balkanski et al., 2018), we randomly select 500 images to be our subsampled ground set since this experiment is throttled by the number and cost of oracle queries.

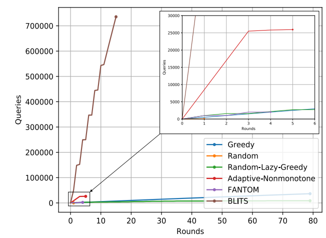
We set $k = 80$ in Figure 1a and track the progress of the algorithms in each round. Figure 1b compares the solution quality for different constraints $k \in (20, 40, 60, 80, 100)$ and demonstrates that ADAPTIVE-NONMONOTONE-MAX and BLITS find substantially better solutions than RANDOM. We use 10 trials for each stochastic algorithm and plot the mean and standard deviation of the solutions. We note that FANTOM performs noticeably worse than the others because it stops choosing elements when their (possibly positive) marginal gain falls below a fixed threshold. We give a picture-in-picture plot of the query complexities in Figure 1c to highlight the difference in overall cost of the estimators for ADAPTIVE-NONMONOTONE-MAX and BLITS.



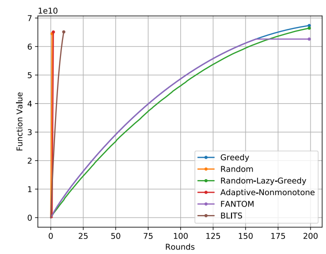
(a) Image Summarization



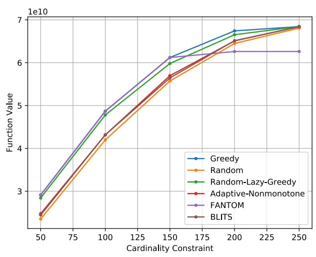
(b) Image Summarization



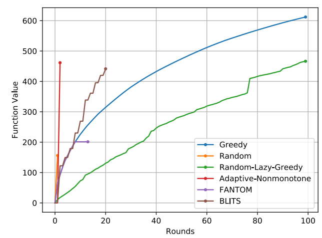
(c) Image Summarization



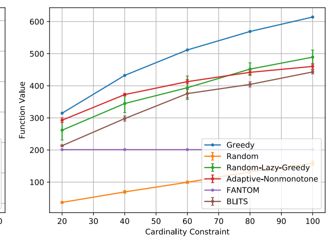
(d) Movie Recommendation



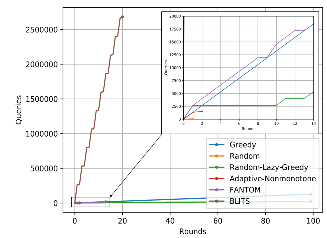
(e) Movie Recommendation



(f) Revenue Maximization



(g) Revenue Maximization



(h) Revenue Maximization

Figure 1. Performance of ADAPTIVE-NONMONOTONE-MAX compared to several benchmarks for image summarization on the CIFAR-10 dataset, movie recommendation on the MovieLens 20M dataset, and revenue maximization on the top 5,000 communities of YouTube.

Movie Recommendation. Personalized movie recommendation systems aim to provide short, comprehensive lists of high-quality movies for a user based on the ratings of similar users. In this experiment, we randomly sample 500 movies from the MovieLens 20M dataset (Harper & Konstan, 2016), which contains 20 million ratings for 26,744 movies by 138,493 users. We use SOFT-IMPUTE (Mazumder et al., 2010) to predict the rating vector for each movie via low-rank matrix completion, and we define the similarity of two movies $s_{i,j}$ as the inner product of the rating vectors for movies i and j. Following (Mirzasoleiman et al., 2016), we use the objective function

$$f(S) = \sum_{i \in N} \sum_{j \in S} s_{i,j} - \lambda \sum_{i \in S} \sum_{j \in S} s_{i,j},$$

with $\lambda = 0.95$. Note that if $\lambda = 1$ we have the cut function.

We remark that experiment is similar to solving max-cut on an Erdös-Rényi graph. In Figure 1d we set $k = 200$, and in Figure 1e we consider $k \in (50, 100, 150, 200, 250)$. The GREEDY algorithm performs moderately better than RANDOM as the constraint approaches $k = 250$, and all other algorithms except FANTOM are sandwiched between these benchmarks. The query complexities are similar to Figure 1c, so we exclude this plot to keep Figure 1 compact.

Revenue Maximization. In this application, our goal is to choose a subset of users in a social network to advertise a product in order to maximize its revenue. We consider the top 5,000 communities of the YouTube network (Leskovec & Krevl, 2014) and subsample the graph by restricting to 25 randomly chosen communities (Balkanski et al., 2018). The resulting network has 1,329 nodes and 3,936 edges. We assign edge weights according to the continuous uniform distribution $\mathcal{U}(0, 1)$, and we measure influence using the non-monotone function

$$f(S) = \sum_{i \in N \setminus S} \sqrt{\sum_{j \in S} w_{i,j}}.$$

In Figure 1f, we set $k = 100$ and observe that ADAPTIVE-NONMONOTONE-MAX significantly outperforms FANTOM and RANDOM. Figure 1g shows a stratification of the algorithms for $k \in (20, 40, 60, 80, 100)$, and Figure 1h is similar to the image summarization experiment. We note that the inner plot in Figure 1h shows that for the optimal threshold of ADAPTIVE-NONMONOTONE-MAX, the number of candidates instantly falls below $3k$ and the algorithm outputs a random prefix of high-valued elements in the next round.

5. Conclusions

We give the first algorithm for maximizing a non-monotone submodular function subject to a cardinality constraint that achieves a constant-factor approximation with nearly optimal adaptivity complexity. The query complexity of this algorithm is also nearly optimal and considerably less than in previous works. While the approximation guarantee is only $0.039 - \varepsilon$, our empirical study shows that for several real-world applications ADAPTIVE-NONMONOTONE-MAX finds solutions that are competitive with the benchmarks for non-monotone submodular maximization and requires significantly fewer rounds and oracle queries.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. Matthew Fahrbach was supported in part by an NSF Graduate Research Fellowship under grant DGE-1650044. Part of this work was done while he was a summer intern at Google Research, Zürich.

References

Agarwal, A., Assadi, S., and Khanna, S. Stochastic submodular cover with limited adaptivity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 323–342. SIAM, 2019.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 671–680. ACM, 2014.

Balkanski, E. and Singer, Y. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1138–1151. ACM, 2018.

Balkanski, E., Breuer, A., and Singer, Y. Non-monotone submodular maximization in exponentially fewer iterations. In *Advances in Neural Information Processing Systems*, pp. 2359–2370, 2018.

Balkanski, E., Rubinstein, A., and Singer, Y. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 283–302. SIAM, 2019.

Barbosa, R., Ene, A., Nguyen, H., and Ward, J. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning*, pp. 1236–1244, 2015.

Barbosa, R. d. P., Ene, A., Nguyen, H. L., and Ward, J. A new framework for distributed submodular maximization. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 645–654. IEEE, 2016.

Buchbinder, N. and Feldman, M. Constrained submodular maximization via a non-symmetric technique. *arXiv preprint arXiv:1611.03253*, 2016.

Buchbinder, N., Feldman, M., Naor, J. S., and Schwartz, R. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1433–1452. SIAM, 2014.

Buchbinder, N., Feldman, M., and Schwartz, R. Comparing apples and oranges: Query trade-off in submodular maximization. *Mathematics of Operations Research*, 42(2): 308–329, 2016.

Chekuri, C. and Quanrud, K. Parallelizing greedy for submodular set function maximization in matroids and beyond. *arXiv preprint arXiv:1811.12568*, 2018.

Chekuri, C. and Quanrud, K. Submodular function maximization in parallel via the multilinear relaxation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 303–322. SIAM, 2019.

Chen, L., Feldman, M., and Karbasi, A. Unconstrained submodular maximization with constant adaptive complexity. *arXiv preprint arXiv:1811.06603*, 2018.

Das, A. and Kempe, D. Algorithms for subset selection in linear regression. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pp. 45–54. ACM, 2008.

Dueck, D. and Frey, B. J. Non-metric affinity propagation for unsupervised image categorization. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8. IEEE, 2007.

El-Arini, K. and Guestrin, C. Beyond keyword search: Discovering relevant scientific literature. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 439–447. ACM, 2011.

Elenberg, E. R., Khanna, R., Dimakis, A. G., Negahban, S., et al. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568, 2018.

Ene, A. and Nguyen, H. L. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 274–282. SIAM, 2019.

Ene, A., Nguyen, H. L., and Vladu, A. A parallel double greedy algorithm for submodular maximization. *arXiv preprint arXiv:1812.01591*, 2018a.

Ene, A., Nguyen, H. L., and Vladu, A. Submodular maximization with matroid and packing constraints in parallel. *arXiv preprint arXiv:1808.09987*, 2018b.

Fahrbach, M., Miller, G. L., Peng, R., Sawlani, S., Wang, J., and Xu, S. C. Graph sketching against adaptive adversaries applied to the minimum degree algorithm. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 101–112. IEEE, 2018.

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

Feige, U., Mirrokni, V. S., and Vondrak, J. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Gharan, S. O. and Vondrák, J. Submodular maximization by simulated annealing. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1098–1116. SIAM, 2011.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):19, 2016.

Hartline, J., Mirrokni, V., and Sundararajan, M. Optimal marketing strategies over social networks. In *Proceedings of the 17th international conference on World Wide Web*, pp. 189–198. ACM, 2008.

Kazemi, E., Zadimoghaddam, M., and Karbasi, A. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In *International Conference on Machine Learning*, pp. 2549–2558, 2018.

Khanna, R., Elenberg, E. R., Dimakis, A. G., Negahban, S., and Ghosh, J. Scalable greedy feature selection via weak submodularity. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1560–1568, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing (TOPC)*, 2(3):14, 2015.

Lattanzi, S., Moseley, B., Suri, S., and Vassilvitskii, S. Filtering: A method for solving graph problems in mapreduce. In *Proceedings of the Twenty-Third Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 85–94. ACM, 2011.

Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Liu, P. and Vondrak, J. Submodular optimization in the mapreduce model. *arXiv preprint arXiv:1810.01489*, 2018.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.

Mirrokni, V. and Zadimoghaddam, M. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pp. 153–162. ACM, 2015.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

Mirzasoleiman, B., Badanidiyuru, A., and Karbasi, A. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*, pp. 1358–1367, 2016.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. MIT Press, 2018.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

Norouzi-Fard, A., Tarnawski, J., Mitrović, S., Zandieh, A., Mousavifar, A., and Svensson, O. Beyond $1/2$-approximation for submodular maximization on massive data streams. *arXiv preprint arXiv:1808.01842*, 2018.

Qian, S. and Singer, Y. Fast parallel algorithms for feature selection. *arXiv preprint arXiv:1903.02656*, 2019.

Simon, I., Snavely, N., and Seitz, S. M. Scene summarization for online image collections. In *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8. IEEE, 2007.

Sipos, R., Swaminathan, A., Shivaswamy, P., and Joachims, T. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pp. 754–763. ACM, 2012.

Tschiatschek, S., Iyer, R. K., Wei, H., and Bilmes, J. A. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems*, pp. 1413–1421, 2014.