Nearly Linear-Time, Parallelizable Algorithms for Non-Monotone Submodular Maximization

Alan Kuhnle
Department of Computer Science, Florida State University, Tallahassee, Florida
kuhnle@cs.fsu.edu

Abstract

We study combinatorial, parallelizable algorithms for maximization of a submodular function, not necessarily monotone, with respect to a cardinality constraint k. We improve the best approximation factor achieved by an algorithm that has optimal adaptivity and query complexity, up to logarithmic factors in the size of the ground set, from 0.039 to nearly 0.193. Heuristic versions of our algorithms are empirically validated to use a low number of adaptive rounds and total queries while obtaining solutions with high objective value in comparison with state-of-the-art approximation algorithms, including continuous algorithms that use the multilinear extension.

1 Introduction

A nonnegative set function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, defined on all subsets of a ground set $\mathcal{N}$ of size n, is *submodular* if for all $A, B \subseteq \mathcal{N}$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. Submodular set functions naturally arise in many learning applications, including data summarization (Simon, Snavely, and Seitz 2007; Sipos et al. 2012; Tschiatschek et al. 2014; Libbrecht, Bilmes, and Stafford 2017), viral marketing (Kempe, Kleinberg, and Tardos 2003; Hartline, Mirrokni, and Sundararajan 2008), and recommendation systems (El-Arini and Guestrin 2011). Some applications yield submodular functions that are not monotone (a set function is monotone if $A \subseteq B$ implies $f(A) \leq f(B)$): for example, image summarization with diversity (Mirzasoleiman, Badanidiyuru, and Karbasi 2016) or revenue maximization on a social network (Hartline, Mirrokni, and Sundararajan 2008). In this work, we study the maximization of a (not necessarily monotone) submodular function subject to a cardinality constraint; that is, given submodular function f and integer k, determine $\arg\max_{|S| \leq k} f(S)$ (SMCC). Access to f is provided through a value query oracle, which when queried with the set S returns the value $f(S)$.

As the amount of data in applications has exhibited exponential growth in recent years (*e.g.* the growth of social networks (Mislove et al. 2008) or genomic data (Libbrecht, Bilmes, and Stafford 2017)), it is necessary to design algorithms for submodular optimization that can scale to these large datasets (Badanidiyuru and Vondrák 2014; Mirzasoleiman et al. 2015; Kuhnle 2019; Crawford 2019, 2020).

One aspect of algorithmic efficiency is the *query complexity*, the total number of queries to the oracle for f; since evaluation of f is often expensive, the queries to f often dominate the runtime of an algorithm. In addition to low query complexity, it is necessary to design algorithms that parallelize well to take advantage of modern computer architectures. To quantify parallelization, the *adaptivity* or *adaptive complexity* of an algorithm is the minimum number of rounds such that in each round the algorithm makes $O(\text{poly}(n))$ independent queries to the evaluation oracle. The lower the adaptive complexity of an algorithm, the more suited the algorithm is to parallelization, as within each adaptive round, the queries to f are independent and may be easily parallelized.

The design of algorithms with nontrivial adaptivity for SMCC when f is monotone has been recently initiated by Balkanski and Singer (2018), who also prove a lower bound of $\Omega(\log n / \log \log n)$ adaptive rounds to achieve a constant ratio. Recently, much work has focused on the design of adaptive algorithms for SMCC with (not necessarily monotone) submodular functions, as summarized in Table 1. However, although many algorithms with low adaptivity have been proposed, most of these algorithms exhibit at least a quadratic dependence of the query complexity on the size n of the ground set, for $k = \Omega(n)$. For many applications, instances have grown too large for quadratic query complexity to be practical. Therefore, it is necessary to design adaptive algorithms that also have nearly linear query complexity. The only algorithm in prior literature that meets this requirement is the $(0.039 - \varepsilon)$-approximation algorithm developed by Fahrbach, Mirrokni, and Zadimoghaddam (2019a), which has $O(n \log k)$ query complexity and $O(\log n)$ adaptivity.

Contributions In this work, we improve the best approximation factor for nearly linear-time algorithms that are highly parallelizable to $0.193 - \varepsilon$. Specifically, we propose two algorithms: the $(1/6 - \varepsilon)$-approximation algorithm ADAPTIVESIMPLETHRESHOLD (AST) with adaptivity $O(\log n)$ and query complexity $O(n \log k)$; and the $(0.193 - \varepsilon)$-approximation algorithm ADAPTIVETHRESHOLDGREEDY (ATG) with adaptivity $O(\log^2 n)$ and query complexity $O(n \log k)$.

Our algorithm AST uses a novel double-thresholding procedure to obtain its ratio of $1/6 - \varepsilon$. Our second algorithm ATG is a low-adaptivity modification of the algorithm of

Reference	Approximation	Adaptivity	Queries
Buchbinder, Feldman, and Schwartz (2015)	$1/e - \varepsilon$	$O(k)$	$O(n)$
Balkanski, Breuer, and Singer (2018)	$1/(2e) - \varepsilon$	$O\left(\log^2(n)\right)$	$O\left(OPT^2 n \log^2(n) \log(k)\right)$
Chekuri and Quanrud (2019)	$3 - 2\sqrt{2} - \varepsilon$	$O(\log^2(n))$	$O\left(nk^4 \log^2(n)\right)$
Ene and Nguyên (2020)	$1/e - \varepsilon$	$O(\log(n))$	$O\left(nk^2 \log^2(n)\right)$
Fahrbach, Mirrokni, and Zadimoghaddam (2019a)	$0.039 - \varepsilon$	$O(\log(n))$	$O(n \log(k))$
Theorem 2 (AST)	$1/6 - \varepsilon$	$O(\log(n))$	$O(n \log(k))$
Theorem 3 (ATG)	$0.193 - \varepsilon$	$O(\log^2(n))$	$O(n \log(k))$

Table 1: Adaptive algorithms for SMCC where objective f is not necessarily monotone

Gupta et al. (2010), for which we improve the ratio from $1/6$ to ≈ 0.193 through a novel analysis. Both of our algorithms use a low-adaptivity, threshold sampling procedure (Fahrbach, Mirrokni, and Zadimoghaddam 2019b,a; Kazemi et al. 2019) and a subroutine for unconstrained maximization of a submodular function (Feige, Mirrokni, and Vondrák 2011; Chen, Feldman, and Karbasi 2019) as components. More details are given in the related work discussion below and in Section 3.

Empirically, we demonstrate that heuristic versions of both of our algorithms achieve superior objective value to current state-of-the-art algorithms while using a small number of queries and adaptive rounds on two applications of SMCC.

Throughout this work, references are made to the full version of the manuscript, which contains complete proofs, additional empirical evaluations, and additional details. The full version is available at https://arxiv.org/abs/2009.01947.

1.1 Related Work

Adaptive Algorithms Since the study of parallelizable algorithms for submodular optimization was initiated by Balkanski and Singer (2018), there have been a number of $O(\log n)$-adaptive algorithms designed for SMCC. When f is monotone, adaptive algorithms that obtain the optimal ratio (Nemhauser and Wolsey 1978) of $1 - 1/e - \varepsilon$ have been designed by Balkanski, Rubinstein, and Singer (2019); Fahrbach, Mirrokni, and Zadimoghaddam (2019b); Ene and Nguyen (2019). Of these, the algorithms of Fahrbach, Mirrokni, and Zadimoghaddam (2019b); Ene and Nguyen (2019) also have nearly optimal query complexity; that is, they have query complexity $O(n \log k)$.

However, when the function f is not monotone, the best approximation ratio with polynomial query complexity for SMCC is unknown, but falls within the range $[0.385, 0.491]$ (Buchbinder and Feldman 2016; Gharan and Vondrák 2011). For SMCC, algorithms with nearly optimal adaptivity have been designed by Balkanski, Breuer, and Singer (2018); Chekuri and Quanrud (2019); Ene, Nguyên, and Vladu (2019); Fahrbach, Mirrokni, and Zadimoghaddam (2019a); for the query complexity and approximation factors of these algorithms, see Table 1. Of these, the best approximation ratio of $(1/e - \varepsilon) \approx 0.368$ is obtained by the algorithm of Ene and Nguyên (2020). However, this algorithm requires access to an oracle for the gradient of the continuous extension of a submodular set function, which requires $\Omega(nk^2 \log^2(n))$ queries to sufficiently approximate; the practical performance of the algorithm of Ene and Nguyên (2020) is investigated in our empirical evaluation of Section 4. Other than the algorithm AdaptiveNonmonotoneMax of Fahrbach, Mirrokni, and Zadimoghaddam (2019a), all parallelizable algorithms exhibit a runtime at least quadratic dependence on n; in contrast, our algorithms have query complexity of $O(n \log k)$ and have $O(\log n)$ or $O(\log^2 n)$ adaptivity.

The IteratedGreedy Algorithm Although the standard greedy algorithm performs arbitrarily badly for SMCC, Gupta et al. (2010) showed that multiple repetitions of the greedy algorithm, combined with an approximation for the unconstrained maximization problem, yields an approximation for SMCC. Specifically, Gupta et al. (2010) provided the IteratedGreedy algorithm, which achieves approximation ratio of $1/6$ for SMCC when the $1/2$-approximation of Buchbinder et al. (2012) is used for the unconstrained maximization subproblems. Our algorithm AdaptiveThresholdGreedy uses ThresholdSample combined with the descending thresholds technique of Badanidiyuru and Vondrák (2014) to obtain an adaptive version of IteratedGreedy, as described in Section 3. Pseudocode for IteratedGreedy is given in the full version, where an improved ratio of ≈ 0.193 is proven for this algorithm; we also prove the ratio of nearly 0.193 for our adaptive algorithm ATG in Section 3.

1.2 Preliminaries

Notation A submodular set function defined on all subsets of ground set $\mathcal{N}$ is denoted by f. The marginal gain of adding an element s to a set S is denoted by $f_s(S) = f(S \cup \{s\}) - f(S)$. The restriction of f to all subsets of a set $S \subseteq \mathcal{N}$ is denoted by $f\restriction_S$.

Next, we describe two subproblems both of our algorithms need to solve: namely, unconstrained maximization subproblems and a threshold sampling subproblem. For both of these subproblems, procedures with low adaptivity are needed.

Unconstrained Maximization The first subproblem is unconstrained maximization of a submodular function. When the function f is non-monotone, the problem of maximizing f without any constraints is NP-hard (Feige, Mirrokni, and Vondrák 2011). Recently, Chen, Feldman, and Karbasi

Algorithm 1 The ADAPTIVESIMPLETHRESHOLD Algorithm

procedure AST($f, k, \varepsilon, \delta$)
  Input: evaluation oracle $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, constraint k, accuracy parameter $\varepsilon > 0$, failure probability $\delta > 0$
  $M \leftarrow \max_{x \in \mathcal{N}} f(x)$; $c \leftarrow 4 + \alpha$, where α^{-1} is ratio of UNCONSTRAINEDMAX
  for $i \leftarrow 0$ to $\log_{1-\varepsilon}(1/(ck))$ in parallel **do**
    $\tau_i \leftarrow M\,(1-\varepsilon)^i$
    $A_i \leftarrow$ THRESHOLDSAMPLE $(f, k, \tau_i, \varepsilon, \delta/2)$
    $B_i \leftarrow$ THRESHOLDSAMPLE $\left(f\restriction_{\mathcal{N} \setminus A_i}, k, \tau_i, \varepsilon, \delta/2\right)$
    $A_i' \leftarrow$ UNCONSTRAINEDMAX(A_i)
    $C_i \leftarrow \arg\max\{f(A_i), f(A_i'), f(B_i)\}$
  return $C \leftarrow \arg\max_i\{f(C_i)\}$

(2019) developed an algorithm that achieves nearly the optimal ratio of $1/2$ with constant adaptivity, as summarized in the following theorem.

Theorem 1 (Chen, Feldman, and Karbasi (2019))**.** *For each $\varepsilon > 0$, 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.*

To achieve the approximation factor listed for our algorithms in Table 1, the algorithm of Chen, Feldman, and Karbasi (2019) is employed for unconstrained maximization subproblems.

THRESHOLDSAMPLE The second subproblem is the following: given a threshold τ, choose a set S such that 1) $f(S) \geq \tau|S|$; 2) if $|S| < k$, then for any $x \notin S$, $f_x(S) < \tau$. Multiple algorithms in the literature satisfy this requirement, including those in Fahrbach, Mirrokni, and Zadimoghaddam (2019b) and Kazemi et al. (2019). In Sections 2 and 3, we analyze our algorithms using the THRESHOLDSAMPLE procedure of Fahrbach, Mirrokni, and Zadimoghaddam (2019b). In brief, THRESHOLDSAMPLE ensures the marginal gain of any singleton falls below a given threshold τ, while the average contribution of elements added is roughly τ with probability $1 - \delta$. THRESHOLDSAMPLE is $O(\log n)$ adaptive and requires linearly many queries. Pseudocode for THRESHOLDSAMPLE is given in the full version. Below, we use the following lemma of Fahrbach, Mirrokni, and Zadimoghaddam (2019b).

Lemma 1 ((Fahrbach, Mirrokni, and Zadimoghaddam 2019b))**.** *The algorithm* THRESHOLDSAMPLE *outputs $S \subseteq \mathcal{N}$ 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) According to a randomly uniformly chosen permutation of S, the expected marginal $\mathbb{E}\left[f_{s_{i+1}}(S_i)\right] \geq (1-\varepsilon)\tau$. 3) If $|S| < k$, then $f_x(S) < \tau$ for all $x \in \mathcal{N}$.*

2 The ADAPTIVESIMPLETHRESHOLD Algorithm

In this section, we present the algorithm ADAPTIVESIMPLETHRESHOLD (AST, Alg. 1) and show it obtains ratio of $1/6 - \varepsilon$ with nearly optimal query and adaptive complexity. Procedures for threshold sampling and unconstrained maximization are required; see Section 1.2 for a discussion of these subproblems.

Overview of Algorithm Algorithm AST works as follows. First, the **for** loop guesses a value of τ close to $\frac{\text{OPT}}{(4+\alpha)k}$, where $1/\alpha$ is the ratio of the algorithm used for the unconstrained maximization subproblem. Next, THRESHOLDSAMPLE is called with parameter τ to yield set A; followed by a second call to THRESHOLDSAMPLE with f restricted to $\mathcal{N} \setminus A$ to yield set B. Next, an unconstrained maximization is performed with f restricted to A to yield set A'; finally, the best of the three candidate sets A, B, A' is returned.

We prove the following theorem concerning the performance of AST.

Theorem 2. *Suppose there exists an $(1/\alpha)$-approximation for* UNCONSTRAINEDMAX *with adaptivity Θ and query complexity Ξ, and let $\varepsilon, \delta > 0$. Then there exists an algorithm for SMCC with expected approximation ratio $\frac{1}{4+\alpha} - \varepsilon$ with probability at least $1 - \delta$, expected query complexity $O\left(\log_{1-\varepsilon}(1/(6k)) \cdot (n/\varepsilon + \Xi)\right)$, and adaptivity $O\left(\log(n/\delta)/\varepsilon + \Theta\right)$.*

If the algorithm of Chen, Feldman, and Karbasi (2019) is used for UNCONSTRAINEDMAX, AST achieves ratio $1/6 - \varepsilon$ with adaptive complexity is $O\left(\log(n/\delta)/\varepsilon + \log(1/\varepsilon)/\varepsilon\right)$ and query complexity $O\left(\log_{1-\varepsilon}(1/(6k)) \cdot \left(n/\varepsilon + n\log^3(1/\varepsilon)/\varepsilon^4\right)\right)$.

Overview of Proof The proof splits into two cases: the case that one of A or B is of size k and the case that both are of size less than k. Since every element added has gain roughly $\tau = \text{OPT}/(ck)$, the first case holds trivially by the number of elements added. To gain intution for the second case, let O be an optimal solution. By submodularity, $f(O) \leq f(O \cap A) + f(O \setminus A)$. The first term is bounded by the unconstrained maximization, and the second term is bounded by an application of submodularity and the fact that the maximum marginal gain of adding an element into A or B is below τ. The specific choice of c balances the trade-off between the two cases of the proof.

Proof of Theorem 2. Let (f, k) be an instance of SMCC, and let $\varepsilon, \delta > 0$. Suppose algorithm AST uses a procedure for UNCONSTRAINEDMAX with expected ratio $1/\alpha$. We will show that the set C returned by algorithm AST$(f, k, \varepsilon, \delta)$ satisfies $\mathbb{E}\left[f(C)\right] \geq \left(\frac{1}{4+\alpha} - \varepsilon\right)\text{OPT}$ with probability at least $(1 - \delta)$, where OPT is the optimal solution value on the instance (f, k).

Observe that $\tau_0 = M = \max_{x \in \mathcal{N}} f(x) \geq \text{OPT}/k$ by submodularity of f. Let $c = 4 + \alpha$. If $j = \lceil \log_{1-\varepsilon}(1/(ck)) \rceil$, then $\tau_j = M(1-\varepsilon)^j \leq \text{OPT}/(ck)$ since $M \leq \text{OPT}$. Hence,

there exists i_0 such that $\frac{(1-\varepsilon)\text{OPT}}{ck} \leq \tau_{i_0} \leq \frac{\text{OPT}}{ck}$. Let A, B, A' denote $A_{i_0}, B_{i_0}, A'_{i_0}$, respectively. For the rest of the proof, we assume that the properties of Lemma 1 hold for the calls to THRESHOLDSAMPLE with threshold τ_{i_0}, which happens with at least probability $1-\delta$ by the union bound.

Case $|A| = k$ or $|B| = k$. Let $S \in \{A, B\}$ satisfy $|S| = k$. By Lemma 1 and the value of τ_{i0}, we have

$$\mathbb{E}\left[f(S)/k\right] = \mathbb{E}\left[f(S)/|S|\right] \geq (1-\varepsilon)\tau_{i_0} \geq \frac{(1-\varepsilon)^2\text{OPT}}{ck} \geq \frac{(1/c-\varepsilon)\text{OPT}}{k}.$$

Then $\mathbb{E}\left[f(C)\right] \geq \mathbb{E}\left[f(S)\right] \geq (1/c-\varepsilon)\text{OPT}$.

Case $|A| < k$ and $|B| < k$. Let O be a set such that $f(O) = \text{OPT}$ and $|O| \leq k$. Since $|A| < k$, by Lemma 1 it holds that for any $x \in \mathcal{N}$, $f_x(A) < \tau_{i_0}$. Similarly, for any $x \in \mathcal{N} \setminus A$, $f_x(B) < \tau_{i_0}$. Hence, by submodularity

$$f(O \cup A) - f(A) \leq \sum_{o \in O} f_o(A) < k\tau_{i_0} \leq \text{OPT}/c, \quad (1)$$

$$f((O \setminus A) \cup B) - f(B) \leq \sum_{o \in O \setminus A} f_o(B) < k\tau_{i_0} \leq \text{OPT}/c. \quad (2)$$

Next, from (1), (2), submodularity, nonnegativity, and the fact that $A \cap B = \emptyset$, we have that

$$\begin{aligned} f(A) + f(B) + 2\text{OPT}/c &\geq f(O \cup A) + f((O \setminus A) \cup B) \\ &\geq f(O \setminus A) + f(O \cup A \cup B) \\ &\geq f(O \setminus A). \end{aligned} \quad (3)$$

Since UNCONSTRAINEDMAX is an α-approximation, we have

$$\alpha\mathbb{E}\left[f(A')\right] \geq f(O \cap A). \quad (4)$$

From Inequalities (3), (4), and submodularity, we have

$$\begin{aligned} \text{OPT} = f(O) &\leq f(O \cap A) + f(O \setminus A) \\ &\leq \alpha\mathbb{E}\left[f(C)\right] + 2\mathbb{E}\left[f(C)\right] + 2\text{OPT}/c, \end{aligned}$$

from which it follows that $\mathbb{E}\left[f(C)\right] \geq \text{OPT}/c$.

Adaptive and query complexities. The adaptivity of AST is twice the adaptivity of THRESHOLDSAMPLE plus the adaptivity of UNCONSTRAINEDMAX plus a constant. Further, the total query complexity is $\log_{1-\varepsilon}(1/(6k))$ times the sum of twice the query complexity of THRESHOLDSAMPLE and the query complexity of UNCONSTRAINEDMAX. □

3 The ADAPTIVETHRESHOLDGREEDY Algorithm

In this section, we present the algorithm ADAPTIVETHRESHOLDGREEDY (ATG, Alg. 2), which achieves ratio $\approx 0.193 - \varepsilon$ in nearly optimal query and adaptive complexities. The price of improving the ratio of the preceding section is an extra $\log(n)$ factor in the adaptivity.

Algorithm 2 The ADAPTIVETHRESHOLDGREEDY Algorithm

procedure ATG($f, k, \varepsilon, \delta$)
Input: evaluation oracle $f : 2^{\mathcal{N}} \to \mathbb{R}^+$, constraint k, accuracy parameter $\varepsilon > 0$, failure probability $\delta > 0$
  $c \leftarrow 8/\varepsilon$, $\varepsilon' \leftarrow (1 - 1/e)\varepsilon/8$, $\delta' \leftarrow \delta/(2\log_{1-\varepsilon'}(1/(ck)))$
  $M \leftarrow \max_{x \in \mathcal{N}} f(x)$, $A \leftarrow \emptyset$, $B \leftarrow \emptyset$
  for $i \leftarrow 0$ to $\log_{1-\varepsilon'}(1/(ck))$ **do**
    $\tau \leftarrow M(1-\varepsilon')^i$
    $S \leftarrow$ THRESHOLDSAMPLE($f_A, k - |A|, \tau, \varepsilon', \delta'$)
    $A \leftarrow A \cup S$
  for $i \leftarrow 0$ to $\log_{1-\varepsilon'}(1/(ck))$ **do**
    $\tau \leftarrow M(1-\varepsilon')^i$
    $S \leftarrow$ THRESHOLDSAMPLE($f_B\restriction_{\mathcal{N}\setminus A}, k - |B|, \tau, \varepsilon', \delta'$)
    $B \leftarrow B \cup S$
  $A' \leftarrow$ UNCONSTRAINEDMAX(A, ε')
  $C \leftarrow \arg\max\{f(A), f(A'), f(B)\}$
  return C

Overview of Algorithm Our algorithm (pseudocode in Alg. 2) works as follows. Each **for** loop corresponds to a low-adaptivity greedy procedure using THRESHOLDSAMPLE with descending thresholds. Thus, the algorithm is structured as two, iterated calls to a greedy algorithm, where the second greedy call is restricted to select elements outside the set A returned by the first. Finally, an unconstrained maximization procedure is used within the first greedily-selected set, and the best of the three candidate sets is returned. In the pseudocode for ATG, Alg. 2, THRESHOLDSAMPLE is called with functions of the form f_S, which is defined to be the submodular function $f_S(\cdot) = f(S \cup \cdot)$.

At a high level, our approach is the following: the ITERATEDGREEDY framework of Gupta et al. (2010) runs two standard greedy algorithms followed by an unconstrained maximization, which yields an algorithm with $O(nk)$ query complexity and $O(k)$ adaptivity. We adopt this gramework but replace the standard greedy algorithm with a novel greedy approach with low adaptivity and query complexity. To design this novel greedy approach, we modify the descending thresholds algorithm of Badanidiyuru and Vondrák (2014), which has query complexity $O(n \log k)$ but very high adaptivity of $\Omega(n \log k)$. We use THRESHOLDSAMPLE to lower the adaptivity of the descending thresholds greedy algorithm (see the full version for pseudocode and a detailed discussion).

For the resulting algorithm ATG, we prove a ratio of $0.193 - \varepsilon$ (Theorem 3), which improves the $1/6$ ratio for ITERATEDGREEDY proven in Gupta et al. (2010). This novel analysis requires that partial solutions returned by the greedy algorithms are analyzed together, which creates difficulties since the output of the second greedy algorithm must be conditioned on the random set A chosen by the first greedy algorithm. If the output of the first greedy approach is conditioned upon selecting set A, a lower bound on the expected marginal gain of choosing elements into A cannot be ob-

tained. Therefore, a careful analysis is required to prove the ratio for the ATG algorithm.

A simpler form of our arguments shows that the improved ratio also holds for the original IteratedGreedy of Gupta et al. (2010); this analysis is given in the full version. We prove the following theorem concerning the performance of ATG.

Theorem 3. *Suppose there exists an $(1/\alpha)$-approximation for* UnconstrainedMax *with adaptivity Θ and query complexity Ξ, and let $\varepsilon, \delta > 0$. Then the algorithm* AdaptiveThresholdGreedy *for SMCC has expected approximation ratio $\frac{e-1}{e(2+\alpha)-\alpha} - \varepsilon$ with probability at least $(1-\delta)$, adaptive complexity of $O\left(\log_{1-\varepsilon}(1/k)\log(n/\delta)/\varepsilon + \Theta\right)$ and expected query complexity of $O\left(\log_{1-\varepsilon}(1/k)\cdot(n/\varepsilon) + \Xi\right)$.*

If the algorithm of Chen, Feldman, and Karbasi (2019) is used for UnconstrainedMax, ATG achieves approximation ratio $\approx 0.193 - \varepsilon$ with adaptive complexity $O(\log(k)\log(n))$ and query complexity $O(n\log(k))$, wherein the ε dependence has been suppressed.

Proof of Theorem 3. In this proof, we assume that the guarantees of Lemma 1 hold for each call to ThresholdSample made by ATG; this occurs with probability at least $(1-\delta)$ by the union bound and the choice of δ'.

Overview of Proof For the proof, a substantial amount of machinery is necessary to lower bound the marginal gain. The necessary definitions are made first; then, in Lemmas 2 – 5, we formulate the necessary lower bounds on the marginal gains for the first and second greedy procedures. For each respective greedy procedure, this is accomplished by considering respective events ω, ω' that fix a subset that defines the choices the algorithm has made up until the call to ThresholdSample that adds a batch of elements that includes that i-th marginal gain; then, the set of all possible events ω, ω' are marginalized over. This allows us to formulate a recurrence on the sum of the expected marginal gains (Lemma 6). Finally, the recurrence allows us to proceed similarly to our proof in for IteratedGreedy after a careful analysis of the error introduced (see the full version).

Definitions of Random Variables Consider the probability space of all possible sequences of sets returned by the successive calls to ThresholdSample on line 6 and line 10 and the call to UnconstrainedMax on line 12. Further, after each call to ThresholdSample, suppose the elements of S returned by ThresholdSample are added to A or B in uniformly random order. Let $\mathcal{A}_i$ be the random variable defined as the value of the set A during the execution of ATG when $|A| = i$; define $\mathcal{B}_i$ analogously. Let $\mathbf{t} = (t_1, t_2, \ldots, t_l)$ and $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_l)$ be the random variables recording the sizes of nonempty sets returned by ThresholdSample and the corresponding values of τ, respectively, during the **for** loop on line 4. Similarly, let $\mathbf{t}' = (t'_1, t'_2, \ldots, t'_{l'})$ and $\boldsymbol{\tau}' = (\tau'_1, \tau'_2, \ldots, \tau'_{l'})$ record the analogous values during the **for** loop on line 8. For $0 \le i \le k$, let $j(i)$ be the maximum value of the following set: $\{\sum_{h=1}^{j} t_h = s_j : j \ge 1 \wedge s_j \le i\}$ or 0 if $t_1 > i$. and let $j'(i)$ be defined analogously for the sequence $(t'_1, \ldots, t'_{l'})$. Finally, let $O \subseteq \mathcal{N}$ have $f(O) = \mathrm{OPT}$, $|O| \le k$.

Let $l \le k$, $\boldsymbol{\alpha} \in \mathbb{N}^l$ and $\boldsymbol{\beta} \in \mathbb{R}^l$ be sequences of length l. Let $i \le k$, and let $E \subseteq \mathcal{N}$. Let ω be the event that $\mathbf{t} = \boldsymbol{\alpha}$ and $\boldsymbol{\tau} = \boldsymbol{\beta}$ and $\mathcal{A}_{j(i)} = E$; let Ω be the collection of all events ω of this form.

The next four lemmas are proven in the full version.

Lemma 2. *Let $\omega \in \Omega$. Then* $\mathbb{E}\left[f(\mathcal{A}_{i+1}) - f(\mathcal{A}_i) \mid \omega\right] + \frac{M}{ck} \ge \frac{(1-\varepsilon')^2}{k} \cdot \left(f(O \cup E) - f(E)\right)$.

Lemma 3. *Let $0 \le i < k$.* $\mathbb{E}\left[f\left(\mathcal{A}_{i+1}\right) - f\left(\mathcal{A}_i\right)\right] + \frac{M}{ck} \ge \frac{(1-\varepsilon')^2}{k}\mathbb{E}\left[f\left(O \cup \mathcal{A}_{j(i)}\right) - f\left(\mathcal{A}_{j(i)}\right)\right]$.

The next two lemmas show an analogous result for the expected gain of $\mathcal{B}$.

Let $l \le k$, and let $\boldsymbol{\alpha} \in \mathbb{N}^l$ and $\boldsymbol{\beta} \in \mathbb{R}^l$ be sequences of length l. Let $i \le k$, let A be a subset of U of size at most k, and let F be a subset of $U \setminus A$. Let ω' be the event that $\mathcal{A} = A \wedge \mathcal{B}_{j'(i)} = F \wedge \mathbf{t}' = \boldsymbol{\alpha} \wedge \boldsymbol{\tau}' = \boldsymbol{\beta}$. Finally, let Ω' be the collection of all events ω' of this form.

Lemma 4. *Let $\omega' \in \Omega'$. Then* $\mathbb{E}\left[f(\mathcal{B}_{i+1}) - f(\mathcal{B}_i) \mid \omega'\right] + \frac{M}{ck} \ge \frac{(1-\varepsilon')^2}{k} \cdot \left(f((O \setminus A) \cup F) - f(F)\right)$.

Lemma 5. *Let $0 \le i < k$.* $\mathbb{E}\left[f\left(\mathcal{B}_{i+1}\right) - f\left(\mathcal{B}_i\right)\right] + \frac{M}{ck} \ge \frac{(1-\varepsilon')^2}{k}\mathbb{E}\left[f\left(O \setminus \mathcal{A} \cup \mathcal{B}_{j'(i)}\right) - f\left(\mathcal{B}_{j'(i)}\right)\right]$.

The next lemma establishes the main recurrence.

Lemma 6. *Let $\Gamma_i = f\left(\mathcal{A}_i\right) + f\left(\mathcal{B}_i\right)$. Then* $\mathbb{E}\left[\Gamma_{i+1}\right] - \mathbb{E}\left[\Gamma_i\right] + \frac{2M}{ck} \ge \frac{(1-\varepsilon')^2}{k}\left(\mathbb{E}\left[f\left(O \setminus \mathcal{A}\right)\right] - \mathbb{E}\left[\Gamma_i\right]\right)$.

Proof of Lemma 6.

$$
\begin{aligned}
&\mathbb{E}\left[\Gamma_{i+1}\right] - \mathbb{E}\left[\Gamma_i\right] + \frac{2M}{ck} \\
&\overset{(a)}{\ge} \frac{(1-\varepsilon')^2}{k}\left(\mathbb{E}\left[f\left(O \setminus \mathcal{A} \cup \mathcal{B}_{j'(i)}\right) - f\left(\mathcal{B}_{j'(i)}\right)\right]\right. \\
&\qquad \left. + \mathbb{E}\left[f\left(O \cup \mathcal{A}_{j(i)}\right) - f\left(\mathcal{A}_{j(i)}\right)\right]\right) \\
&\overset{(b)}{=} \frac{(1-\varepsilon')^2}{k}\left(\mathbb{E}\left[f\left(O \cup \mathcal{A}_{j(i)}\right) + f\left(O \setminus \mathcal{A} \cup \mathcal{B}_{j'(i)}\right)\right]\right. \\
&\qquad \left. - \mathbb{E}\left[f\left(\mathcal{A}_{j(i)}\right) - f\left(\mathcal{B}_{j'(i)}\right)\right]\right) \\
&\overset{(c)}{\ge} \frac{(1-\varepsilon')^2}{k}\mathbb{E}\left[f\left(O \setminus \mathcal{A}\right) - f\left(\mathcal{A}_{j(i)}\right) - f\left(\mathcal{B}_{j'(i)}\right)\right] \\
&\overset{(d)}{\ge} \frac{(1-\varepsilon')^2}{k}\left(\mathbb{E}\left[f\left(O \setminus \mathcal{A}\right)\right] - \mathbb{E}\left[\Gamma_i\right]\right),
\end{aligned}
$$

where (a) and (b) follow from linearity of expectation and Lemmas 3 and 5.

Inequality (c) follows from the submodularity and nonnegativity of f and the definition of expected value; indeed,

$$
\begin{aligned}
f\left((O \setminus A) \cup B'\right) + f\left(O \cup A'\right) - f\left(A'\right) - f\left(B'\right) \\
\ge f\left(O \setminus A\right) - f\left(A'\right) - f\left(B'\right),
\end{aligned}
$$

from submodularity and nonnegativity of f, for any sets satisfying $A' \subseteq A$ and $B' \cap A = \emptyset$; the values of the random

variables during any single run of the algorithm satisfy these conditions.

Inequality (d) follows from linearity of expectation and the facts that during any run of the algorithm, $j(i) \leq i$, $j'(i) \leq i$; and $\mathbb{E}\left[f\left(\mathcal{A}_{j+1}\right) - f\left(\mathcal{A}_j\right)\right] \geq 0$ and $\mathbb{E}\left[f\left(\mathcal{B}_{j+1}\right) - f\left(\mathcal{B}_j\right)\right] \geq 0$ for any j; these latter two inequalities hold by Lemma 1 and imply $\mathbb{E}\left[f\left(\mathcal{A}_i\right)\right] \geq \mathbb{E}\left[f\left(\mathcal{A}_{j(i)}\right)\right]$ and $\mathbb{E}\left[f\left(\mathcal{B}_i\right)\right] \geq \mathbb{E}\left[f\left(\mathcal{B}_{j'(i)}\right)\right]$. □

Lemma 6 yields a recurrence of the form $u_{i+1} \geq au_i + b$, $u_0 = 0$, and has the solution $u_i \geq \frac{b}{1-a}(1-a^i)$. Consequently, we have

$$\mathbb{E}\left[f(A_k)\right] + \mathbb{E}\left[f(B_k)\right] \geq \left[\mathbb{E}\left[f(O \setminus A)\right] - \frac{2M}{c(1-\varepsilon')^2}\right] \cdot \left(1 - e^{-(1-\varepsilon')^2}\right). \quad (5)$$

Let $\beta = 1 - e^{-(1-\varepsilon')^2}$. From the choice of C on line 13, we have $2f(C) \geq f(A) + f(B)$ and so from (5), we have

$$\begin{aligned}\mathbb{E}\left[f(O \setminus A)\right] &\leq \frac{2}{\beta}\mathbb{E}\left[f(C)\right] + \frac{2M}{c(1-\varepsilon')^2} \\ &\leq \frac{2}{\beta}\mathbb{E}\left[f(C)\right] + \frac{2f(O)}{c(1-\varepsilon')^2}. \quad (6)\end{aligned}$$

For any set A, $f(O) \leq f(O \cap A) + f(O \setminus A)$ by submodularity and nonnegativity. Therefore,

$$f(O) \leq \mathbb{E}\left[f(O \cap A)\right] + \mathbb{E}\left[f(O \setminus A)\right]. \quad (7)$$

Since an $(1/\alpha)$-approximation is used for UNCONSTRAINEDMAX, for any A, $f(O \cap A)/\alpha \leq \mathbb{E}\left[f(C)|A\right]$; therefore,

$$\mathbb{E}\left[f(O \cap A)\right] \leq \alpha\mathbb{E}\left[f(C)\right]. \quad (8)$$

From (6), (7), and (8) and the choices of c, ε' on line 2, we have from Lemma 7 in the full version

$$\begin{aligned}\mathbb{E}\left[f(C)\right] &\geq \left(\frac{1 - \frac{2}{c(1-\varepsilon')^2}}{\alpha + \frac{2}{\beta}}\right) f(O) \\ &\geq \left(\frac{(e-1)}{\alpha(e-1) + 2e} - \varepsilon\right) f(O). \quad \square\end{aligned}$$

4 Empirical Evaluation

In this section, we evaluate our algorithm in comparison with the state-of-the-art parallelizable algorithms: ADAPTIVENONMONOTONEMAX of Fahrbach, Mirrokni, and Zadimoghaddam (2019a) and the algorithm of Ene and Nguyên (2020). Our results are summarized as follows.

- Our algorithm ATG obtains the best objective value of any of the parallelizable algorithms; obtaining an improvement of up to 18% over the next algorithm, our AST. Both Fahrbach, Mirrokni, and Zadimoghaddam (2019a) and Ene and Nguyên (2020) exhibit a large loss of objective value at small k values; see Figs. 1(a) and 1(c).
- Both our algorithm AST and ADAPTIVENONMONOTONEMAX use a very small number of adaptive rounds. Both ATG and the algorithm of Ene and Nguyên (2020) use roughly an order of magnitude more adaptive rounds; see Figs. 1(b) and 1(e).
- The algorithm of Ene and Nguyên (2020) is the most query efficient if access is provided to an exact oracle for the multilinear extension of a submodular function and its gradient[1]. However, if these oracles must be approximated with the set function, their algorithm becomes very inefficient and does not scale beyond small instances ($n \leq 100$).

  Our algorithms used fewer queries to the submodular set function than the linear-time algorithm of Buchbinder, Feldman, and Schwartz (2015).

Algorithms In addition to the algorithms discussed in the preceding paragraphs, we evaluate the following baselines: the ITERATEDGREEDY algorithm of Gupta et al. (2010), and the linear-time $(1/e - \varepsilon)$-approximation algorithm FASTRANDOMGREEDY of Buchbinder, Feldman, and Schwartz (2015). These algorithms are both $O(k)$-adaptive, where k is the cardinality constraint.

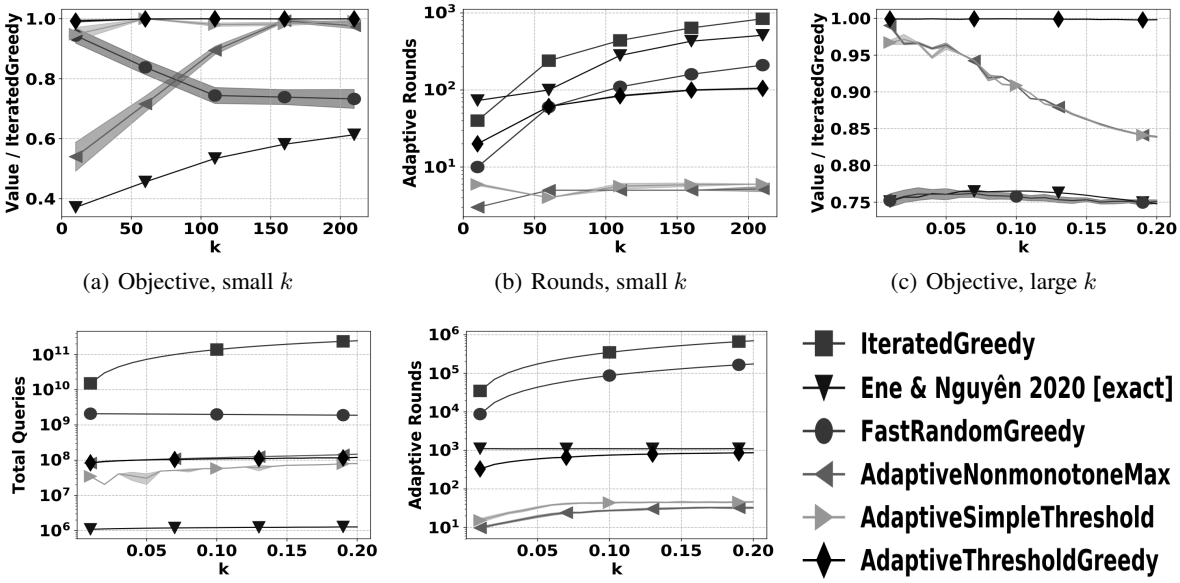
For all algorithms, accuracy parameter ε was set to 0.1; 100 samples were used to evaluate expectations in all adaptive algorithms (thus, these algorithms were run as heuristics with no performance guarantee). Randomized algorithms are averaged over 20 independent repetitions, and the mean is reported. The standard deviation is indicated by a shaded region in the plots. Any algorithm that requires a subroutine for UNCONSTRAINEDMAX is implemented to use a random set, which is a $(1/4)$-approximation by Feige, Mirrokni, and Vondrák (2011).

Applications All combinatorial algorithms are evaluated on two applications of SMCC: the cardinality-constrained maximum cut application and revenue maximization on social networks, a variant of the influence maximization problem in which k users are selected to maximize revenue. We evaluate on a variety of network technologies from the Stanford Large Network Dataset Collection (Leskovec and Krevl 2020).

The algorithm of Ene and Nguyên (2020) requires access to an oracle for the multilinear extension and its gradient. In the case of maximum cut, the multilinear extension and its gradient can be computed in closed form in time linear in the size of the graph, as described in the full version. This fact enables us to evaluate the algorithm of Ene and Nguyên (2020) using direct oracle access to the multilinear extension and its gradient on the maximum cut application. However, no closed form exists for the multilinear extension of the revenue maximization objective. In this case, we found (see the full version) that sampling to approximate the multilinear extension is exorbitant in terms of runtime; hence, we were unable to evaluate Ene and Nguyên (2020) on revenue maximization. For more details on the applications and datasets, see the full version.

Results on cardinality-constrained maximum cut. In Fig. 1, we show representative results for cardinality-constrained maximum cut on web-Google ($n = 875713$)

[1]The definition of the multilinear extension is given in the full version



(a) Objective, small k (b) Rounds, small k (c) Objective, large k

(d) Queries, large k (e) Rounds, large k (f) Legend

Figure 1: Comparison of objective value (normalized by the ITERATEDGREEDY objective value), total queries, and adaptive rounds on web-Google for the maxcut application for both small and large k values. The large k values are given as a fraction of the number of nodes in the network. The algorithm of Ene and Nguyên (2020) is run with oracle access to the multilinear extension and its gradient; total queries reported for this algorithm are queries to these oracles, rather than the original set function.

for both small and large k values. Results on other datasets and revenue maximization are given in the full version. In addition, results for Ene and Nguyên (2020) when the multilinear extension is approximated via sampling are given in the full version. The algorithms are evaluated by objective value of solution, total queries made to the oracle, and the number of adaptive rounds (lower is better). Objective value is normalized by that of ITERATEDGREEDY.

In terms of objective value (Figs. 1(a) and 1(c)), our algorithm ATG maintained better than 0.99 of the ITERATEDGREEDY value, while all other algorithms fell below 0.95 of the ITERATEDGREEDY value on some instances. Our algorithm AST obtained similar objective value to ADAPTIVENONMONOTONEMAX on larger k values, but performed much better on small k values. Finally, the algorithm of Ene and Nguyên (2020) obtained poor objective value for $k \leq 100$ and about 0.75 of the ITERATEDGREEDY value on larger k values. It is interesting to observe that the two algorithms with the best approximation ratio of $1/e$, Ene and Nguyên (2020) and FASTRANDOMGREEDY, returned the worst objective values on larger k (Fig. 1(c)).

For total queries (Fig. 1(d)), the most efficient is Ene and Nguyên (2020), although it does not query the set function directly, but the multilinear extension and its gradient. The most efficient of the combinatorial algorithms was AST, followed by ATG. Finally, with respect to the number of adaptive rounds (Fig. 1(e)), the best was ADAPTIVENONMONOTONEMAX, closely followed by AST; the next lowest was AST, followed by Ene and Nguyên (2020).

5 Conclusions and Future Work

We have provided two combinatorial algorithms for efficient and parallelizable maximization of submodular functions; both algorithms are within polylogarithmic factors of optimal adaptivity and query complexity. The approximation factor of such a nearly optimal algorithm is improved by our analysis from $0.039 - \varepsilon$ of Fahrbach, Mirrokni, and Zadimoghaddam (2019a) to $0.193 - \varepsilon$. An empirical evaluation of heuristic versions of our algorithms demonstrate that our simpler algorithm AST nearly matches the adaptivity of Fahrbach, Mirrokni, and Zadimoghaddam (2019a) while improving on solution quality, while our second algorithm sacrifices a small amount of adaptivity for even better solution quality. Finally, an empirical evaluation of the state-of-the-art algorithm (Ene and Nguyên 2020) for parallelizable maximization of continous DR-submodular functions on the multilinear extension of the maximum cut objective shows that if an oracle to the multilinear extension is unavailable, combinatorial algorithms are much more efficient than approximation of the multilinear extension by sampling.

Although we improved the ratio of fast, parallelizable algorithms for SMCC, there is still a theoretical gap between our ratio of $0.193 - \varepsilon$ for nearly linear-time combinatorial algorithms and the best known approximation factor for this problem (0.385 of Buchbinder and Feldman (2016).) Hence, future work includes narrowing this gap. Also, future work includes investigating if modifications of our algorithms could be run with theoretical guarantees on quality of solution, instead of as heuristics.

Acknowledgments

The work of Alan Kuhnle was partially supported by Florida State University. Yixin Chen, Victoria G. Crawford, and the anonymous reviewers provided helpful feedback on earlier versions of the manuscript.

References

Badanidiyuru, A.; and Vondrák, J. 2014. Fast algorithms for maximizing submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Balkanski, E.; Breuer, A.; and Singer, Y. 2018. Non-monotone Submodular Maximization in Exponentially Fewer Iterations. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Balkanski, E.; Rubinstein, A.; and Singer, Y. 2019. An Exponential Speedup in Parallel Running Time for Submodular Maximization without Loss in Approximation. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Balkanski, E.; and Singer, Y. 2018. The adaptive complexity of maximizing a submodular function. In *ACM SIGACT Symposium on Theory of Computing (STOC)*.

Buchbinder, N.; and Feldman, M. 2016. Constrained Submodular Maximization via a Non-symmetric Technique. *Mathematics of Operations Research* 44(3).

Buchbinder, N.; Feldman, M.; Naor, J. S.; and Schwartz, R. 2012. A Tight Linear Time (1 / 2)-Approximation for Unconstrained Submodular Maximization. In *Symposium on Foundations of Computer Science (FOCS)*.

Buchbinder, N.; Feldman, M.; and Schwartz, R. 2015. Comparing Apples and Oranges: Query Tradeoff in Submodular Maximization. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Chekuri, C.; and Quanrud, K. 2019. Parallelizing greedy for submodular set function maximization in matroids and beyond. *Proceedings of the Annual ACM Symposium on Theory of Computing* 78–89.

Chen, L.; Feldman, M.; and Karbasi, A. 2019. Unconstrained submodular maximization with constant adaptive complexity. In *STOC*, 102–113.

Crawford, V. G. 2019. An Efficient Evolutionary Algorithm for Minimum Cost Submodular Cover. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Crawford, V. G. 2020. Faster Guarantees of Pareto Optimization for Submodular Maximization. In *arxiv preprint arXiv:1908:01230*.

El-Arini, K.; and Guestrin, C. 2011. Beyond Keyword Search: Discovering Relevant Scientific Literature. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Ene, A.; and Nguyen, H. L. 2019. Submodular Maximization with Nearly-optimal Approximation and Adaptivity in Nearly-linear Time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Ene, A.; and Nguyên, H. L. 2020. Parallel Algorithm for Non-Monotone DR-Submodular Maximization. In *International Conference on Machine Learning (ICML)*.

Ene, A.; Nguyên, H. L.; and Vladu, A. 2019. Submodular maximization with matroid and packing constraints in parallel. *Proceedings of the Annual ACM Symposium on Theory of Computing* 90–101.

Fahrbach, M.; Mirrokni, V.; and Zadimoghaddam, M. 2019a. Non-monotone Submodular Maximization with Nearly Optimal Adaptivity Complexity. In *International Conference on Machine Learning (ICML)*.

Fahrbach, M.; Mirrokni, V.; and Zadimoghaddam, M. 2019b. Submodular Maximization with Nearly Optimal Approximation, Adaptivity, and Query Complexity. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 255–273.

Feige, U.; Mirrokni, V.; and Vondrák, J. 2011. Maximizing non-monotone submodular functions. *SIAM Journal on Computing* .

Gharan, S. O.; and Vondrák, J. 2011. Submodular maximization by simulated annealing. *ACM-SIAM Symposium on Discrete Algorithms (SODA)* .

Gupta, A.; Roth, A.; Schoenebeck, G.; and Talwar, K. 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics (WINE)*.

Hartline, J.; Mirrokni, V. S.; and Sundararajan, M. 2008. Optimal marketing strategies over social networks. *International Conference on World Wide Web (WWW)* 189–198.

Kazemi, E.; Mitrovic, M.; Zadimoghaddam, M.; Lattanzi, S.; and Karbasi, A. 2019. Submodular Streaming in All its Glory: Tight Approximation, Minimum Memory and Low Adaptive Complexity. In *International Conference on Machine Learning (ICML)*.

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Kuhnle, A. 2019. Interlaced Greedy Algorithm for Maximization of Submodular Functions in Nearly Linear Time. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Leskovec, J.; and Krevl, A. 2020. SNAP Datasets: Stanford Large Network Dataset Collection. URL http://snap.stanford.edu/data. Accessed 2021-03-17.

Libbrecht, M. W.; Bilmes, J. A.; and Stafford, W. 2017. Choosing non-redundant representative subsets of protein sequence data sets using submodular optimization. *Proteins: Structure, Function, and Bioinformatics* (July 2017): 454–466.

Mirzasoleiman, B.; Badanidiyuru, A.; and Karbasi, A. 2016. Fast Constrained Submodular Maximization : Personalized Data Summarization. In *International Conference on Machine Learning (ICML)*.

Mirzasoleiman, B.; Badanidiyuru, A.; Karbasi, A.; Vondrak, J.; and Krause, A. 2015. Lazier Than Lazy Greedy. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Mislove, A.; Koppula, H. S.; Gummadi, K. P.; Druschel, P.; and Bhattacharjee, B. 2008. Growth of the Flickr Social Network. In *First Workshop on Online Social Networks*.

Nemhauser, G. L.; and Wolsey, L. A. 1978. Best Algorithms for Approximating the Maximum of a Submodular Set Function. *Mathematics of Operations Research* 3(3): 177–188.

Simon, I.; Snavely, N.; and Seitz, S. M. 2007. Scene summarization for online image collections. In *IEEE International Conference on Computer Vision (ICCV)*.

Sipos, R.; Swaminathan, A.; Shivaswamy, P.; and Joachims, T. 2012. Temporal corpus summarization using submodular word coverage. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

Tschiatschek, S.; Iyer, R.; Wei, H.; and Bilmes, J. 2014. Learning Mixtures of Submodular Functions for Image Cxollection Summarization. In *Advances in Neural Information Processing Systems (NeurIPS)*.