Guarantees of Stochastic Greedy Algorithms for Non-monotone Submodular Maximization with Cardinality Constraint

Shinsaku Sakaue
NTT Communication Science Laboratories

Abstract

Submodular maximization with a cardinality constraint can model various problems, and those problems are often very large in practice. For the case where objective functions are monotone, many fast approximation algorithms have been developed. The stochastic greedy algorithm (SG) is one such algorithm, which is widely used thanks to its simplicity, efficiency, and high empirical performance. However, its approximation guarantee has been proved only for monotone objective functions. When it comes to non-monotone objective functions, existing approximation algorithms are inefficient relative to the fast algorithms developed for the case of monotone objectives. In this paper, we prove that SG (with slight modification) can achieve almost 1/4-approximation guarantees in expectation in linear time even if objective functions are non-monotone. Our result provides a constant-factor approximation algorithm with the fewest oracle queries for non-monotone submodular maximization with a cardinality constraint. Experiments validate the performance of (modified) SG.

1 INTRODUCTION

We consider the following submodular function maximization problem with a cardinality constraint:

$$\underset{S\subseteq V}{\text{maximize}} \quad f(S) \qquad \text{subject to} \quad |S| \leq k, \qquad (1)$$

where V is a finite ground set of n elements, $f : 2^V \to \mathbb{R}$ is a non-negative submodular function, and k ($\leq n$) is a positive integer. As is conventionally done, we assume the value oracle model (i.e., $f(\cdot)$ is a black-box function) and discuss the complexity of algorithms in terms of the number of oracle queries, which we call the oracle complexity. Since the evaluation of f is often expensive, to develop oracle-efficient algorithms has been an important research subject.

For the case where f is monotone, the standard greedy algorithm can achieve a $(1-1/\mathrm{e})$-approximation guarantee with $O(kn)$ queries (Nemhauser et al., 1978); this, however, is often too costly when applied to practical large-size instances. To deal with such large instances, various fast algorithms have been developed (Badanidiyuru and Vondrák, 2014; Wei et al., 2014). The stochastic greedy algorithm (SG) (Mirzasoleiman et al., 2015) is one such algorithm: In each iteration, instead of finding the element with the maximum marginal gain at the cost of up to n queries, we sample (roughly) $\frac{n}{k}\ln\frac{1}{\epsilon}$ elements uniformly at random, where $\epsilon \in (\mathrm{e}^{-k}, 1)$, and choose the element with the largest marginal gain out of the sampled elements. SG requires about $n\ln\frac{1}{\epsilon}$ oracle queries in total, and it is known to achieve a $(1-1/\mathrm{e}-\epsilon)$-approximation guarantee if f is monotone. Thanks to its simplicity, efficiency, strong guarantee, and high empirical performance, SG has been used in various studies (Song et al., 2017; Hashemi et al., 2018).

Non-monotone submodular functions also appear in many practical scenarios: sensor placement (Krause et al., 2008), document summarization (Lin and Bilmes, 2010), feature selection (Iyer and Bilmes, 2012), and recommendation (Mirzasoleiman et al., 2016). Unfortunately, the problem becomes much harder if f is non-monotone; for example, the approximation ratio of the greedy algorithm can become arbitrarily poor (at most $1/k$-approximation) in general as in (Pan et al., 2014, Appendix H.1). Although various constant-factor approximation algorithms for non-monotone objectives have been developed (Buchbinder et al., 2014, 2017; Kuhnle, 2019), they often require much more oracle queries than the aforementioned fast algorithms developed for monotone objectives, including SG. Therefore, non-monotone submodular maximization with a cardi-

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

Table 1: Comparison of Fast Algorithms for Non-monotone Submodular Maximization with Cardinality Constraint.

	Approximation ratio	Oracle complexity		Remark
Our result	$\frac{1}{4}(1-\delta)^2$	$n\ln 2 + n\delta\frac{k}{k-1}$ $\max\{n, k+\frac{2k}{\delta}\}\times\ln 2 + k$	(expectation) (worst case)	Randomized
Buchbinder et al. (2017)	$1/\mathrm{e}-\epsilon$	$O\left(\frac{n}{\epsilon^2}\ln\frac{1}{\epsilon}\right)$		Randomized
Kuhnle (2019)	$1/4-\epsilon$	$O\left(\frac{n}{\epsilon}\ln\frac{n}{\epsilon}\right)$		Deterministic

nality constraint is currently awaiting oracle-efficient constant-factor approximation algorithms.

1.1 Our Contribution

We prove approximation guarantees of (modified) SG for non-monotone objective functions, thus providing oracle-efficient approximation algorithms for non-monotone submodular maximization with a cardinality constraint. Below we detail our contributions:

- Assuming $n \geq 3k$, we prove that SG can achieve a $\frac{1}{4}\left(1-2\cdot\frac{k-1}{n-k}\right)^2$-approximation guarantee in expectation by setting ϵ at $\frac{1}{2}+\frac{k-1}{n-k}$. Namely, if $n \gg k$, SG can achieve an approximation ratio close to $1/4$ with about $n\ln 2$ queries.

- We develop modified SG such that the sample size in each iteration is also stochastic. The resulting algorithm achieves a $\frac{1}{4}(1-\delta)^2$-approximation guarantee in expectation. The expected and worst-case oracle complexities are bounded by $n\ln 2 + n\delta\frac{k}{k-1} = O(n)$ and $\max\{n, k+\frac{2k}{\delta}\}\times\ln 2 + k \leq O(n/\delta)$, respectively. Namely, modified SG is a randomized linear-time constant-factor approximation algorithm. As will be discussed in Section 1.2, this result provides a constant-factor approximation algorithm with the fewest oracle queries.

- Experiments confirm the efficiency and high performance of (modified) SG; they run much faster and require far fewer queries than existing algorithms while achieving comparable objective values. The results demonstrate that we can use (modified) SG as practical and theoretically guaranteed algorithms even for non-monotone objectives.

Note, however, that the approximation guarantees are required to hold only in expectation; the worst-case approximation ratio can be arbitrarily bad. This is why it is possible to achieve the constant-factor approximation guarantee with oracle queries possibly fewer than n, which may be counter-intuitive at first glance.

1.2 Related Work

SG was proposed by Mirzasoleiman et al. (2015) as an accelerated version of the well-known greedy algorithm (Nemhauser et al., 1978) for monotone submodular maximization with a cardinality constraint. Hassidim and Singer (2017) studied a variant of SG for monotone objectives and proved a guarantee that holds with a high probability. Guarantees of SG for monotone set functions with approximate submodularity have also been widely studied (Khanna et al., 2017; Hashemi et al., 2018; de Veciana et al., 2019). Harshaw et al. (2019) studied SG for maximizing set functions written as $f = g - c$, where g is monotone weakly submodular and c is non-negative modular; while f can be non-monotone, they do not consider the whole class of non-monotone submodular functions and their approximation guarantee cannot be written with a multiplicative factor unlike our results.

Constrained non-monotone submodular maximization has been extensively studied (Lee et al., 2010; Gupta et al., 2010; Feldman et al., 2011). For the cardinality-constrained case, Buchbinder et al. (2014) proposed the random greedy algorithm, which behaves differently than SG. Specifically, it chooses an element uniformly at random from the top-k most beneficial elements in each iteration. While it achieves a 1/e-approximation guarantee, its oracle complexity is $O(kn)$, which is as costly as the standard greedy algorithm. They also achieved the best approximation ratio, $1/\mathrm{e}+0.004$, by combining the random greedy and continuous double greedy algorithms. Buchbinder and Feldman (2018) derandomized the random greedy algorithm and achieved a 1/e-approximation guarantee with $O(k^2n)$ oracle queries; 1/e is the best ratio achieved by deterministic algorithms. As regards hardness results, Vondrák (2013) proved that to improve a 1/2-approximation guarantee requires exponentially many queries when $k = n/2$. For the case of $k = o(n)$, Gharan and Vondrák (2011) proved a stronger hardness of 0.491-approximation.

Regarding oracle-efficient algorithms, Buchbinder et al. (2017) proposed the random sampling algorithm (RS), which achieves a $(1/\mathrm{e}-\epsilon)$-approximation with

$O\left(\frac{n}{\epsilon^2}\ln\frac{1}{\epsilon}\right)$ oracle queries; to the best of our knowledge, this is the only existing linear-time constant-factor approximation algorithm. More precisely, RS requires at least $\frac{8n}{\epsilon^2}\ln\frac{2}{\epsilon}$ queries; hence, to obtain a non-negative approximation ratio, we need at least $8\mathrm{e}^2 n\ln(2\mathrm{e}) \geq 100n$ queries. On the other hand, the expected and worst-case oracle complexities of the modified SG are at most $n\ln 2 + n\delta\frac{k}{k-1}$ and $\max\{n, k+\frac{2k}{\delta}\}\times\ln 2 + k$, respectively. Therefore, taking the constant factors into account, SG is far faster than RS. In Section 4, we experimentally confirm that this gap is crucial in practice. Buchbinder et al. (2017) also developed another algorithm that achieves a $(1/\mathrm{e}-\epsilon)$-approximation guarantee with $O\left(k\sqrt{\frac{n}{\epsilon}\ln\frac{k}{\epsilon}} + \frac{n}{\epsilon}\ln\frac{k}{\epsilon}\right)$ oracle queries in expectation. Since $k = \Theta(n)$ in general, it is more costly than SG. Recently, Kuhnle (2019) proposed a deterministic $(1/4-\epsilon)$-approximation algorithm with $O(\frac{n}{\epsilon}\ln\frac{n}{\epsilon})$ queries, which is the best oracle complexity among those of deterministic algorithms. Note that it is also slower than SG due to the presence of the $\ln n$ factor. Table 1 compares the above results and ours.

We remark that our work is different from (Qian et al., 2018; Ji et al., 2020), which are seemingly overlapping with ours. Their algorithms for non-monotone objectives are not SG-style ones but variants of the aforementioned random greedy algorithm. Hence, unlike SG and the above efficient algorithms, their algorithms generally require $O(kn)$ queries. Approximation algorithms for non-monotone submodular maximization with more general constraints have also been studied (Mirzasoleiman et al., 2016; Feldman et al., 2017). If those algorithms are applied to the cardinality-constrained case, we need $\Omega(kn)$ queries in general.

Recently, parallel non-monotone submodular maximization algorithms have been widely studied (Balkanski et al., 2018; Ene et al., 2019; Fahrbach et al., 2019). Unlike us, they are interested in a different complexity framework called the adaptive complexity, which is defined with the number of sequential rounds required when polynomially many oracle queries can be executed in parallel. As summarized in (Fahrbach et al., 2019), such parallel algorithms require more than $\Omega(n)$ oracle queries; among them, a $(0.039-\epsilon)$-approximation algorithm of (Fahrbach et al., 2019) requires the fewest queries, $O(\frac{n}{\epsilon^2}\ln k)$, in expectation. Unlike those algorithms, SG requires only $O(n)$ queries in expectation.

1.3 Notation and Definitions

Given a set function $f : 2^V \to \mathbb{R}$, we define $f_X(Y) \coloneqq f(X\cup Y) - f(X)$ for any $X, Y \subseteq V$. We sometimes abuse the notation and regard $v \in V$ as a subset (e.g., we use $f_X(v)$ instead of $f_X(\{v\})$). We say f is non-negative if $f(X) \geq 0$ for any $X \subseteq V$, monotone if $f_X(v) \geq 0$ for any $X \subseteq V$ and $v \notin X$, normalized if $f(\emptyset) = 0$, and submodular if $f(X) + f(Y) \geq f(X\cup Y) + f(X\cap Y)$ for any $X, Y \subseteq V$, which is also equivalently characterized by the following diminishing return property: $f_X(v) \geq f_Y(v)$ for any $X \subseteq Y$ and $v \notin Y$. In this paper, all set functions are assumed to be non-negative and submodular (not necessarily monotone and normalized) unless otherwise specified. In what follows, we use A^* to denote an optimal solution to problem (1).

Algorithm 1 Stochastic Greedy (SG)

1: $A_0 \leftarrow \emptyset$
2: **for** $i = 1, \ldots, k$ **do**
3: Get R by sampling $\lceil s\rceil$ elements from $V\backslash A_{i-1}$
4: $a_i \leftarrow \mathrm{argmax}_{a\in R} f_{A_{i-1}}(a)$
5: **if** $f_{A_{i-1}}(a_i) > 0$ **then** $A_i \leftarrow A_{i-1}\cup\{a_i\}$
6: **else** $A_i \leftarrow A_{i-1}$
7: **return** A_k

1.4 Organization

Section 2 reviews the details of SG and the proof for the case of monotone objectives. In Section 3 we prove the approximation guarantees of (modified) SG for the case of non-monotone objectives. Section 4 presents the experimental results. Section 5 concludes this paper. All missing proofs are presented in the appendix.

2 STOCHASTIC GREEDY AND PROOF FOR MONOTONE CASE

We here review the details of SG and the proof for the case of monotone objectives (Mirzasoleiman et al., 2015), which will help us to understand the main discussion presented in Section 3.

Let $s \coloneqq \frac{n}{k}\ln\frac{1}{\epsilon}$. In each iteration of SG (Algorithm 1), we choose the best element from $\lceil s\rceil$ elements sampled uniformly at random from $V\backslash A_{i-1}$ without replacement. We remark that Algorithm 1 is slightly different from the original SG (Mirzasoleiman et al., 2015): since the marginal gain can be negative due to the lack of monotonicity, we let Algorithm 1 to reject elements with non-positive marginal gains as in Steps 5 and 6 (elements with zero gains are rejected to simplify the discussion in Section 3.2). As is usual with the proofs of greedy-style algorithms, we first consider lower bounding the marginal gain of each iteration as follows:

Lemma 1 (cf. (Mirzasoleiman et al., 2015))**.** *If f is non-negative and submodular, for $i = 1, \ldots, k$, we have*

$$\mathbb{E}[f(A_i) - f(A_{i-1})] \geq \frac{1-\epsilon}{k}\mathbb{E}[f_{A_{i-1}}(A^*)].$$

While Mirzasoleiman et al. (2015) proved this lemma implicitly relying on the monotonicity of f, we can prove it even if f is non-monotone due to the non-negativity of $\mathbb{E}[f(A_i) - f(A_{i-1})]$, which is obtained from Steps 5 and 6 (see, Appendix A for the proof). This non-monotone version of the lemma will play an important role in the proof of our main result presented in Section 3.

We now see how to prove the $(1-1/\mathrm{e}-\epsilon)$-approximation guarantee of SG for the case of monotone objectives. Assume that f is monotone and normalized. We have $\mathbb{E}[f(A^* \cup A_{i-1})] \geq f(A^*)$ due to the monotonicity, and thus Lemma 1 implies

$$\mathbb{E}[f(A_i) - f(A_{i-1})] \geq \frac{1-\epsilon}{k}(f(A^*) - \mathbb{E}[f(A_{i-1})]).$$

By using this inequality for $i = 1, \ldots, k$ and $f(\emptyset) = 0$, we obtain the desired result as follows:

$$\begin{aligned}\mathbb{E}[f(A_k)] &\geq f(A^*) - \left(1 - \frac{1-\epsilon}{k}\right)^k (f(A^*) - f(\emptyset)) \\ &\geq \left(1 - \frac{1}{\mathrm{e}^{1-\epsilon}}\right) f(A^*) \geq \left(1 - \frac{1}{\mathrm{e}} - \epsilon\right) f(A^*).\end{aligned}$$

In the above proof, the inequality, $\mathbb{E}[f(A^* \cup A_{i-1})] \geq f(A^*)$, obtained with the monotonicity, played a key role. As will be shown in Section 3.1, we can derive a variant of the inequality for non-monotone f by using the randomness of SG, which enables us to prove approximation guarantees without the monotonicity.

3 PROOF FOR NON-MONOTONE CASE

We present approximation guarantees of (modified) SG for non-monotone objectives. In Section 3.1, we prove the $\frac{1}{4}\left(1 - 2 \cdot \frac{k-1}{n-k}\right)^2$-approximation guarantee of SG, and in Section 3.2 we prove the $\frac{1}{4}(1-\delta)^2$-approximation guarantee of modified SG.

3.1 $\frac{1}{4}\left(1 - 2 \cdot \frac{k-1}{n-k}\right)^2$-approximation of SG

We here make the following assumption:

Assumption 1. *We assume that $k \geq 2$ and $n \geq 3k$ hold and that ϵ is set so as to satisfy $1/\mathrm{e} \leq \epsilon < 1$.*

The first assumption, $k \geq 2$, is natural since, if $k = 1$, an α-approximation guarantee ($\forall \alpha \in [0, 1]$) can be achieved in expectation by examining $\lceil \alpha n \rceil$ elements, which means any approximation ratio can be achieved in linear time. Hence we assume $k \geq 2$ in what follows. The second assumption, $n \geq 3k$, will be removed in Section 3.2. The third assumption, $1/\mathrm{e} \leq \epsilon < 1$, can be easily satisfied since ϵ is a controllable input.

We derive a variant of $\mathbb{E}[f(A^* \cup A_{i-1})] \geq f(A^*)$ for non-monotone f. To this end, we use the following lemma:

Lemma 2 (Buchbinder et al. (2014), Lemma 2.2)**.** *Let $g : 2^V \to \mathbb{R}$ be submodular. Denote by $A(p)$ a random subset of $A \subseteq V$ where each element appears with a probability of at most p (not necessarily independently). Then, $\mathbb{E}[g(A(p))] \geq (1-p)g(\emptyset)$.*

Namely, if A_{i-1} includes each $a \in V$ with a probability of at most p, then $\mathbb{E}[f(A^* \cup A_{i-1})] \geq (1-p)f(A^*)$ holds. Below we upper bound p by leveraging the randomness of SG and prove the following lemma:

Lemma 3. *Assume that $1/\mathrm{e} \leq \epsilon < 1$ holds. Then, for $i = 0, \ldots, k$, we have*

$$\mathbb{E}[f(A^* \cup A_i)] \geq \left(1 - \frac{1}{k}\ln\frac{1}{\epsilon} - \frac{2}{n-k}\right)^i f(A^*). \quad (2)$$

Proof of Lemma 3. If $i = 0$, the lemma holds since $A_0 = \emptyset$. Below we assume $i \geq 1$. In the i-th iteration, conditioned on A_{i-1}, each $a \in V \backslash A_{i-1}$ stays outside of A_i with a probability of at least $1 - \frac{\lceil s \rceil}{|V \backslash A_{i-1}|}$. Hence, after i iterations ($i = 1, \ldots, k$), each $a \in V$ stays outside of A_i with a probability of at least

$$\begin{aligned}\prod_{j=1}^{i}\left(1 - \frac{\lceil s \rceil}{|V \backslash A_{j-1}|}\right) &\geq \prod_{j=1}^{i}\left(1 - \frac{s+1}{n-k}\right) \\ &= \left(1 - \frac{1}{k}\ln\frac{1}{\epsilon} - \frac{1 + \ln\frac{1}{\epsilon}}{n-k}\right)^i.\end{aligned}$$

Therefore, from $\epsilon \geq 1/\mathrm{e}$, we obtain

$$\Pr[a \in A_i] \leq 1 - \left(1 - \frac{1}{k}\ln\frac{1}{\epsilon} - \frac{2}{n-k}\right)^i.$$

We define $g(A) \coloneqq f(A \cup A^*)$, which we can easily confirm to be submodular. From Lemma 2, we obtain

$$\begin{aligned}\mathbb{E}[f(A^* \cup A_i)] &= \mathbb{E}[g(A_i)] \\ &\geq \left(1 - \frac{1}{k}\ln\frac{1}{\epsilon} - \frac{2}{n-k}\right)^i \mathbb{E}[g(\emptyset)] \\ &= \left(1 - \frac{1}{k}\ln\frac{1}{\epsilon} - \frac{2}{n-k}\right)^i f(A^*).\end{aligned}$$

Hence the lemma holds. □

We then consider lower bounding the RHS of (2) for $i = k-1$. Intuitively, if $n \gg k$ and the $\frac{2}{n-k}$ term is ignorably small, the RHS can be lower bounded by $\epsilon f(A^*)$ since $\left(1 - \frac{1}{k}\ln\frac{1}{\epsilon}\right)^{k-1} \approx \mathrm{e}^{-\ln\frac{1}{\epsilon}} = \epsilon$. By evaluating the RHS more carefully using Assumption 1, we can obtain the following lemma (proof is provided in Appendix B):

Lemma 4. *If Assumption 1 holds, we have*

$$\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{k-1}\geq\epsilon-2\cdot\frac{k-1}{n-k}.$$

We are now ready to prove the approximation guarantee of SG for the case of non-monotone objectives.

Theorem 1. *Let A be the output of Algorithm 1. If Assumption 1 holds, we have*

$$\mathbb{E}[f(A)]\geq\left(\epsilon-2\cdot\frac{k-1}{n-k}\right)(1-\epsilon)f(A^*).$$

By setting $\epsilon=\frac{1}{2}+\frac{k-1}{n-k}$, we obtain

$$\mathbb{E}[f(A)]\geq\frac{1}{4}\left(1-2\cdot\frac{k-1}{n-k}\right)^2 f(A^*).$$

Note that $1/\mathrm{e}\leq\frac{1}{2}+\frac{k-1}{n-k}<1$ holds since $n\geq 3k$. The following proof is partly inspired by the technique used in (Buchbinder et al., 2014).

Proof of Theorem 1. We prove

$$\frac{\mathbb{E}[f(A_i)]}{f(A^*)}\geq\frac{i}{k}\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{i-1}(1-\epsilon)\quad(3)$$

for $i=0,\dots,k$ by induction. If $i=0$, the RHS of (3) becomes 0, and so the inequality holds due to the non-negativity of f. Assume that (3) holds for every $i'=0,\dots,i-1$. Then we have

$$\begin{aligned}
&\mathbb{E}[f(A_i)]\\
&=\mathbb{E}[f(A_{i-1})]+\mathbb{E}[f(A_i)-f(A_{i-1})]\\
&\geq\mathbb{E}[f(A_{i-1})]+\frac{1-\epsilon}{k}\mathbb{E}\left[f(A^*\cup A_{i-1})-f(A_{i-1})\right]\\
&\qquad\qquad\text{(Lemma 1)}\\
&=\left(1-\frac{1-\epsilon}{k}\right)\mathbb{E}[f(A_{i-1})]\\
&\quad+\frac{1-\epsilon}{k}\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{i-1}f(A^*)\\
&\qquad\qquad\text{(Lemma 3)}\\
&\geq\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)\mathbb{E}[f(A_{i-1})]\\
&\quad+\frac{1-\epsilon}{k}\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{i-1}f(A^*)\\
&\qquad\qquad(1-\epsilon\leq\ln\tfrac{1}{\epsilon}\leq\ln\tfrac{1}{\epsilon}+\tfrac{2k}{n-k})\\
&\geq\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)\\
&\quad\times\frac{i-1}{k}\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{i-2}(1-\epsilon)f(A^*)\\
&\quad+\frac{1-\epsilon}{k}\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{i-1}f(A^*)\\
&\qquad\qquad\text{(Assumption of induction)}\\
&=\frac{i}{k}\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{i-1}(1-\epsilon)f(A^*).
\end{aligned}$$

Hence (3) holds for $i=0,\dots,k$; for $i=k$, we have

$$\mathbb{E}[f(A_k)]\geq\left(1-\frac{1}{k}\ln\frac{1}{\epsilon}-\frac{2}{n-k}\right)^{k-1}(1-\epsilon)f(A^*).$$

Finally, by using Lemma 4, we can lower bound the approximation ratio by $\left(\epsilon-2\cdot\frac{k-1}{n-k}\right)(1-\epsilon)$. □

Note that, while Lemma 1 motivates us to let ϵ to be small, the opposite is true regarding Lemma 3. Thus we set $\epsilon\approx 1/2$ to balance the effects of the two inequalities.

3.2 $\frac{1}{4}(1-\delta)^2$-approximation of Modified SG

As shown in Section 3.1, the approximation ratio of SG becomes close to $1/4$ if $n\gg k$. In this section, we first consider improving the ratio by adding sufficiently many dummy elements to V. We then present modified SG that can achieve a $\frac{1}{4}(1-\delta)^2$-approximation guarantee without using dummy elements explicitly.

Let D be a set of dummy elements and $\overline{V}=V\cup D$; i.e., we have $f_A(a)=0$ for any $A\subseteq\overline{V}$ and $a\in D$. We add sufficiently many dummy elements to V so that $N\coloneqq|\overline{V}|$ becomes equal to $\max\{n,k+\lceil(2k-1)/\delta\rceil\}$, where $\delta\in(0,1)$ is an input parameter; smaller δ means that we add more dummy elements. Note that we have $N\geq k+2(k-1)/\delta$ and $N\geq n$. We can also easily prove $N\geq 3k$ by induction; this enables us to remove the second assumption of Assumption 1.

We now consider performing SG on $\overline{V}$. Let $\overline{A}$ be the output of SG and $\overline{A^*}=\operatorname{argmax}_{S\subseteq\overline{V}:|S|\leq k}f(S)$. Thanks to Theorem 1, we have

$$\mathbb{E}[f(\overline{A})]\geq\left(\epsilon-2\cdot\frac{k-1}{N-k}\right)(1-\epsilon)f(\overline{A^*}).$$

If we set $\epsilon=\frac{1}{2}+\frac{k-1}{N-k}$, we obtain

$$\begin{aligned}
\mathbb{E}[f(\overline{A})]&\geq\frac{1}{4}\left(1-2\cdot\frac{k-1}{N-k}\right)^2 f(\overline{A^*})\\
&\geq\frac{1}{4}(1-\delta)^2 f(\overline{A^*})\geq\frac{1}{4}(1-\delta)^2 f(A^*),
\end{aligned}$$

where the second inequality comes from $N\geq k+2(k-1)/\delta$ and the last inequality comes from $V\subseteq\overline{V}$. Furthermore, since no elements with zero marginal gains are added to the current solution in each iteration, $\overline{A}$ includes no elements in D (i.e., $\overline{A}\subseteq V$). Therefore, $\overline{A}$ is a feasible $\frac{1}{4}(1-\delta)^2$-approximate solution.

Algorithm 2 Modified SG

$N \leftarrow \max\{n, k + \lceil (2k-1)/\delta \rceil\}$ and $\overline{s} \leftarrow \frac{N}{k} \ln \frac{1}{\epsilon}$
$A_0 \leftarrow \emptyset$
for $i = 1, \ldots, k$ **do**
  Draw $r \sim \mathrm{H}\left(\lceil \overline{s} \rceil, |V \backslash A_{i-1}|, N - |A_{i-1}|\right)$
  Get R by sampling r elements from $V \backslash A_{i-1}$
  $a_i \leftarrow \mathrm{argmax}_{a \in R} f_{A_{i-1}}(a)$
  if $f_{A_{i-1}}(a_i) > 0$ **then** $A_i \leftarrow A_{i-1} \cup \{a_i\}$
  else $A_i \leftarrow A_{i-1}$
return A_k

We then discuss the oracle complexity of performing SG on $\overline{V}$. In each iteration, we sample $\lceil \overline{s} \rceil$ elements to get R, where $\overline{s} := \frac{N}{k} \ln \frac{1}{\epsilon}$, and then we compute $a_i = \mathrm{argmax}_{a \in R} f_{A_{i-1}}(a)$. Note that, if $a \in D$, we do not need to compute $f_{A_{i-1}}(a)$ since the value is always equal to 0; i.e., any $a \in D$ is taken out of consideration. Therefore, only the number of elements belonging to $R \cap V$ matters to the oracle complexity, which conforms to the hypergeometric distribution with a population of size $|\overline{V} \backslash A_{i-1}|$, $\lceil \overline{s} \rceil$ draws, and $|V \backslash A_{i-1}|$ targets. We denote the distribution by $\mathrm{H}(\lceil \overline{s} \rceil, |V \backslash A_{i-1}|, |\overline{V} \backslash A_{i-1}|)$. Note that its mean is bounded as follows:

$$\lceil \overline{s} \rceil \frac{|V \backslash A_{i-1}|}{|\overline{V} \backslash A_{i-1}|} \leq \left(\frac{N}{k} \ln \frac{1}{\epsilon} + 1 \right) \frac{n}{N-k} \leq \frac{n}{k} \ln \frac{1}{\epsilon} + \frac{n}{k-1} \delta,$$

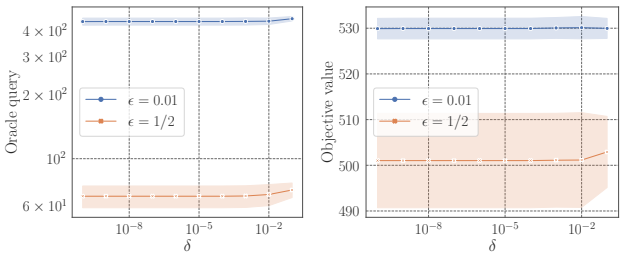
where the last inequality comes from $N \geq k + 2(k-1)/\delta$ and $\epsilon \geq 1/\mathrm{e}$. Thus, the total oracle complexity is at most $n \ln \frac{1}{\epsilon} + n\delta \frac{k}{k-1}$ in expectation, which can be arbitrarily close to $n \ln \frac{1}{\epsilon}$ if δ is sufficiently small. Therefore, by setting $\epsilon = \frac{1}{2} + \frac{k-1}{N-k} \geq \frac{1}{2}$, we can achieve a $\frac{1}{4}(1-\delta)^2$-approximation guarantee with at most $n \ln 2 + n\delta \frac{k}{k-1}$ oracle queries in expectation. Furthermore, the worst-case oracle complexity is also bounded by

$$k \lceil \overline{s} \rceil = N \ln \frac{1}{\epsilon} + k \leq \max \left\{ n, k + \frac{2k}{\delta} \right\} \times \ln \frac{1}{\epsilon} + k.$$

Finally, we see that no dummy elements are needed explicitly. As mentioned above, only the elements in $R \cap V$ affects the behavior of SG performed on $\overline{V}$, and so an algorithm with the same behavior can be obtained by sampling $R \subseteq V$ as follows: Draw $r \in [0, \lceil \overline{s} \rceil]$ from the hypergeometric distribution $\mathrm{H}\left(\lceil \overline{s} \rceil, |V \backslash A_{i-1}|, N - |A_{i-1}|\right)$ and get R by sampling r elements uniformly at random from $V \backslash A_{i-1}$ without replacement. Algorithm 2, called modified SG, presents the details. To conclude, we obtain the following result:

Theorem 2. *Fix ϵ and δ so that $\epsilon \in [1/\mathrm{e}, 1)$ and $\delta \in (0, \epsilon)$ hold, respectively. Then, Algorithm 2 outputs solution A that satisfies*

$$\mathbb{E}[f(A)] \geq (\epsilon - \delta)(1 - \epsilon) f(A^*).$$



(a) Oracle Query (semi-log) (b) Objective Value

Figure 1: MSG Performance with Various δ Values.

The expected and worst-case oracle complexities are at most $n \ln \frac{1}{\epsilon} + n\delta \frac{k}{k-1}$ and $\max\left\{n, k + \frac{2k}{\delta}\right\} \times \ln \frac{1}{\epsilon} + k$, respectively. If we set $\epsilon = \frac{1}{2} + \frac{k-1}{N-k}$, we have

$$\mathbb{E}[f(A)] \geq \frac{1}{4}(1-\delta)^2 f(A^*).$$

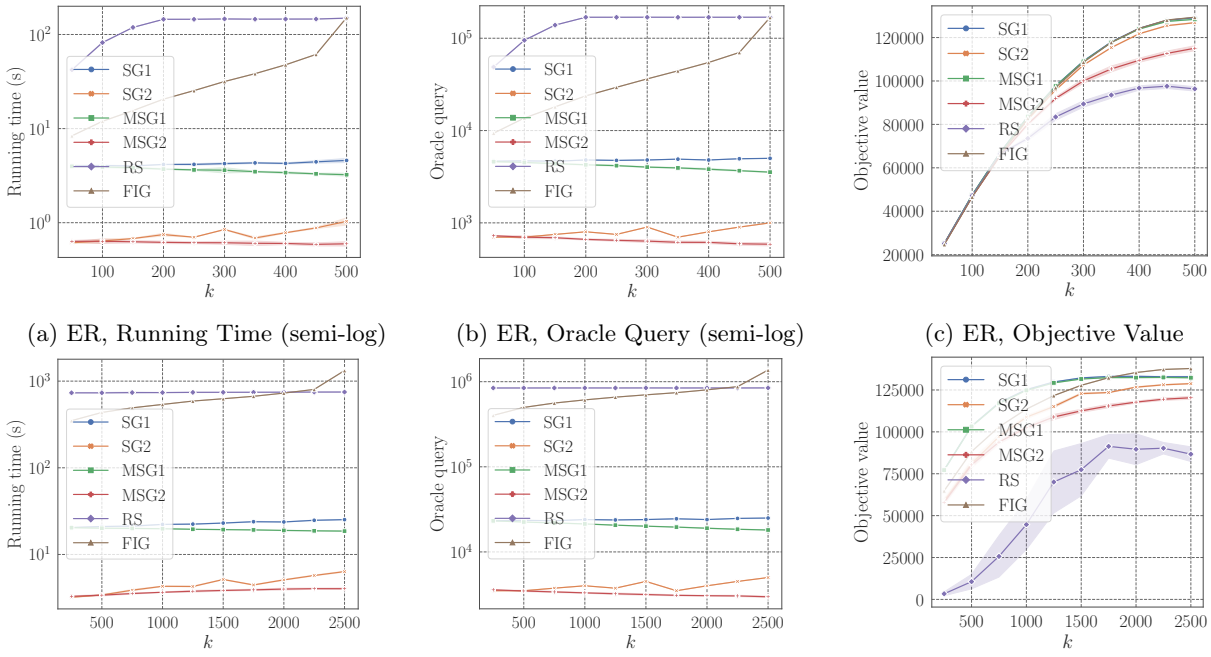
The expected and worst-case oracle complexities are bounded by $n \ln 2 + n\delta \frac{k}{k-1}$ and $\max\left\{n, k + \frac{2k}{\delta}\right\} \times \ln 2 + k$, respectively.

Note that Algorithm 2 can also achieve a $(1 - \frac{1}{\mathrm{e}} - \epsilon)$-approximation guarantee if f is monotone since it is equivalent to Algorithm 1 performed on $\overline{V}$ and the obtained solution is feasible as explained above.

4 EXPERIMENTS

We evaluate (modified) SG via experiments. All the algorithms are implemented in Python3, and all the experiments are conducted on a 64-bit macOS (Mojave) machine with 3.3 GHz Intel Core i7 CPUs and 16 GB RAM. In Section 4.1, we examine the empirical effect of the δ value on the behavior of modified SG. We then compare the following four kinds of algorithms with synthetic and real-world instances in Sections 4.2 and 4.3, respectively.

- SG (Algorithm 1): We consider two algorithms, SG1 and SG2, that employ $\epsilon = 0.01$ and $\epsilon = 1/2$, respectively. The approximation guarantee of SG1 is not proved since it violates $\epsilon \geq 1/\mathrm{e}$; we here use it as a heuristic method and study its empirical behavior. SG2 achieves a $\frac{1}{4}\left(1 - 4 \cdot \frac{k-1}{n-k}\right)$-approximation guarantee if Assumption 1 holds.
- Modified SG (MSG) (Algorithm 2): As with SG, we consider two algorithms: MSG1 ($\epsilon = 0.01$) and MSG2 ($\epsilon = 1/2$). In Sections 4.2 and 4.3, we let $\delta = 0.1$. The approximation guarantee of MSG1 is not proved, while MSG2 achieves a 0.2-approximation guarantee.



(a) ER, Running Time (semi-log) (b) ER, Oracle Query (semi-log) (c) ER, Objective Value

(d) BA, Running Time (semi-log) (e) BA, Oracle Query (semi-log) (f) BA, Objective Value

Figure 2: Comparison of Algorithms with Synthetic Cut-function Maximization Instances.

- Random sampling (RS) (Buchbinder et al., 2017): A randomized $(1/\mathrm{e} - \epsilon)$-approximation algorithm with $O\left(\frac{n}{\epsilon^2} \ln \frac{1}{\epsilon}\right)$ oracle queries. We set $\epsilon = 0.3$ as in the experiments of (Kuhnle, 2019), which yields about a 0.07-approximation guarantee.

- Fast interlace greedy (FIG) (Kuhnle, 2019): A deterministic $(1/4 - \epsilon)$-approximation algorithm with $O\left(\frac{n}{\epsilon} \ln \frac{n}{\epsilon}\right)$ oracle queries. As in the experiments of (Kuhnle, 2019), we set a parameter of the algorithm (denoted by δ in the original paper) at 0.1, which yields a 0.1-approximation guarantee.

When implementing those algorithms in practice, we may employ various acceleration methods including the lazy evaluation (Minoux, 1978; Leskovec et al., 2007) and memoization (Iyer and Bilmes, 2019). However, we here do not use them since our aim is to make simple and clear comparisons of the algorithms.

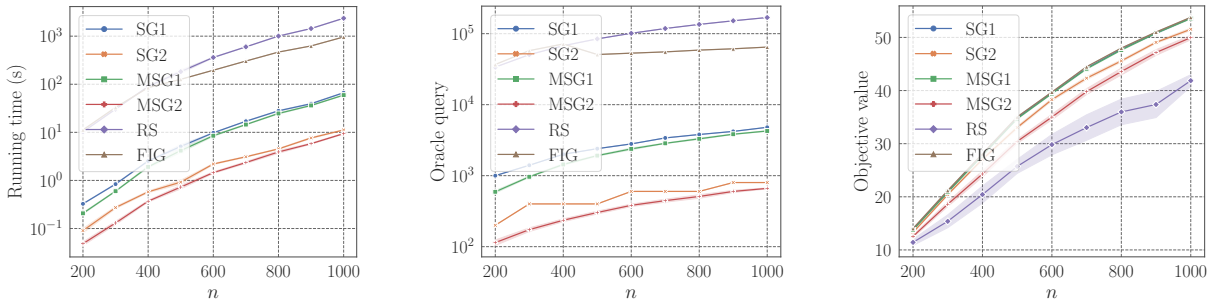
4.1 Empirical Effects of δ Values

We consider a synthetic instance of maximizing a cut function, which is non-negative and submodular. We construct an Erdős–Rényi (ER) random graph with $n = 100$ nodes, edge probability $p = 1/2$, and uniform edge weights. The objective function to be maximized is a cut function defined on the graph, where we can choose up to $k = 10$ nodes. We apply MSG1 and MSG2 with $\delta = 10^{-10}, 10^{-9} \ldots, 10^{-1}$ to the instance.

The numbers of oracle queries and objective values are shown in Figures 1a and 1b, respectively, where each curve and error band indicate the average and standard deviation calculated over 100 trials. While the ϵ value affects the performance (smaller ϵ leads to better objective values and more queries), the increase in the δ value has little effect. This suggests that, while δ is introduced to handle the $2 \cdot \frac{k-1}{n-k}$ term that appears in the approximation ratio derived in Section 3.1, it is actually not essential. We leave it an open problem whether we can prove an approximation guarantee without introducing parameters like δ. In what follows, we let $\delta = 0.1$.

4.2 Synthetic Instance

We compare the algorithms with two synthetic cut-function maximization instances. One is a larger version of the above instance: We construct an ER random graph with $n = 1000$, $p = 1/2$, and uniform edge weights. Another is defined with a Barabási–Albert (BA) random graph with $n = 5000$ nodes and uniform edge weights, which is constructed as follows: Starting from 50 nodes, we alternately add a new node and connect it to 50 existing nodes. For the ER and BA instances, we consider various cardinality constraints with $k = 50, 100, \ldots, 500$ and $k = 250, 500, \ldots, 2500$, respectively. We apply SG1, SG2, MSG1, MSG2, RS, and FIG to the instances and observe the running times, numbers of oracle queries, and objective values. The



(a) Running Time (semi-log) (b) Oracle Query (semi-log) (c) Objective Value

Figure 3: Comparison of Algorithms with Real-world Mutual Information Maximization Instances.

results of the randomized algorithms are shown by the mean and standard deviation calculated over 10 trails.

Figure 2 summarizes the results. With both ER and BA instances, SG and MSG run much faster and require far fewer oracle queries than RS and FIG. For each ϵ value, MSG tends to be more efficient than SG. Regarding the ER instances, SG1, MSG1, and FIG achieve almost the same objective values. The objective value of SG2 is slightly worse than them. MSG2 performs worse than those above, but it still outperforms RS by a considerable margin. As regards the BA instances, SG1 and MSG1 outperform FIG when k is small, and the opposite is true when k is large. Objective values of SG2 and MSG2 are worse than that of FIG, but they are far better than that of RS. To conclude, SG-style algorithms are far more efficient than the existing methods, while achieving comparable objective values.

4.3 Real-world Instance

We compare the algorithms with real-world instances. We employ the mutual information as an objective function. Given a positive semidefinite matrix $\mathbf{X} \in \mathbb{R}^{V \times V}$, we let $\mathbf{X}[S]$ denote the principal submatrix of $\mathbf{X}$ indexed by $S \subseteq V$. We define the entropy function as $H(S) \coloneqq \ln \det \mathbf{X}[S]$ ($H(\emptyset) \coloneqq 0$), which is submodular due to the Ky Fan's inequality. We assume that the smallest eigenvalue of $\mathbf{X}$ is larger than or equal to 1, which makes the entropy function monotone and non-negative. The mutual information is defined as $f(S) = H(S) + H(V \backslash S) - H(V)$, which is known to be submodular (Krause et al., 2008; Sharma et al., 2015). The function is non-negative since $f(S) = H(S) + H(V \backslash S) - H(V) \geq H(\emptyset) = 0$ for any $S \subseteq V$ due to the submodularity of $H(\cdot)$ and $H(\emptyset) = 0$.

We consider a feature selection instance based on mutual information maximization (Iyer and Bilmes, 2012; Sharma et al., 2015). Given a matrix $\mathbf{A}$, whose column indices correspond to features, we define the mutual information with $\mathbf{X} \coloneqq \mathbf{I} + \mathbf{A}^\top \mathbf{A}$. To obtain matrix $\mathbf{A}$, we use "Geographical Original of Music" dataset available at (Olson et al., 2017). The dataset has 117 features, and we create additional $\binom{117}{2}$ second-order polynomial feature vectors as in (Bertsimas et al., 2016). By adding some of them to the original 117 features and normalizing the columns of resulting $\mathbf{A}$, we obtain $n \times n$ matrices $\mathbf{X}$ for $n = 200, 300, \ldots, 1000$. We let $k = 200$. We apply the algorithms to the instances with various n values. The results are again shown by the mean and standard deviation over 10 trials.

Figure 3 summarizes the results. As with the results of synthetic instances, the SG-style algorithms are far more efficient than FIG and RS. Oracle queries of all the algorithms increase very slowly with n in the semi-log plot, which is consistent with the fact that their oracle complexities are (nearly) linear in n. The results of objective values are also similar to those of the synthetic instances: The objective values of SG-style algorithms are as good as or slightly worse than that of FIG, but they are far better than that of RS.

5 CONCLUSION AND DISCUSSION

We proved approximation guarantees of (modified) SG for non-monotone submodular maximization with a cardinality constraint. We first proved a $\frac{1}{4}\left(1 - 2 \cdot \frac{k-1}{n-k}\right)^2$-approximation guarantee of SG under some assumptions; this yields a positive approximation ratio if n is sufficiently larger than k. We then developed modified SG and proved its $\frac{1}{4}(1-\delta)^2$-approximation guarantee without using the assumptions. We also showed that modified SG requires at most $n \ln 2 + n\delta \frac{k}{k-1}$ and $\max\{n, k + \frac{2k}{\delta}\} \times \ln 2 + k$ oracle queries in expectation and in the worst-case, respectively. This result provides a constant-factor approximation algorithm with the fewest oracle queries. Experiments demonstrated that (modified) SG can run much faster and require far fewer oracle queries than existing methods while achieving comparable objective values.

Acknowledgements

The author is grateful to Kaito Fujii, Takanori Maehara, and anonymous reviewers for providing valuable comments.

References

Badanidiyuru, A. and Vondrák, J. (2014). Fast algorithms for maximizing submodular functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1497–1514. SIAM.

Balkanski, E., Breuer, A., and Singer, Y. (2018). Non-monotone submodular maximization in exponentially fewer iterations. In *Advances in Neural Information Processing Systems 31*, pages 2353–2364. Curran Associates, Inc.

Bertsimas, D., King, A., and Mazumder, R. (2016). Best subset selection via a modern optimization lens. *Ann. Statist.*, 44(2):813–852.

Buchbinder, N. and Feldman, M. (2018). Deterministic algorithms for submodular maximization problems. *ACM Trans. Algorithms*, 14(3):32:1–32:20.

Buchbinder, N., Feldman, M., Naor, J. S., and Schwartz, R. (2014). Submodular maximization with cardinality constraints. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1433–1452. SIAM.

Buchbinder, N., Feldman, M., and Schwartz, R. (2017). Comparing apples and oranges: Query trade-off in submodular maximization. *Math. Oper. Res.*, 42(2):308–329.

de Veciana, G., Hashemi, A., and Vikalo, H. (2019). Stochastic-greedy++: Closing the optimality gap in exact weak submodular maximization. *arXiv preprint arXiv:1907.09064*.

Ene, A., Nguyen, H., and Vladu, A. (2019). Submodular maximization with matroid and packing constraints in parallel. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 90–101. ACM.

Fahrbach, M., Mirrokni, V., and Zadimoghaddam, M. (2019). Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 1833–1842. PMLR.

Feldman, M., Harshaw, C., and Karbasi, A. (2017). Greed is good: Near-optimal submodular maximization via greedy optimization. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65, pages 758–784. PMLR.

Feldman, M., Naor, J., and Schwartz, R. (2011). A unified continuous greedy algorithm for submodular maximization. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 570–579.

Gharan, S. O. and Vondrák, J. (2011). Submodular maximization by simulated annealing. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1098–1116. SIAM.

Gupta, A., Roth, A., Schoenebeck, G., and Talwar, K. (2010). Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proceedings of the 6th International Conference on Internet and Network Economics*, pages 246–257. Springer-Verlag.

Harshaw, C., Feldman, M., Ward, J., and Karbasi, A. (2019). Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2634–2643. PMLR.

Hashemi, A., Ghasemi, M., Vikalo, H., and Topcu, U. (2018). A randomized greedy algorithm for near-optimal sensor scheduling in large-scale sensor networks. In *2018 Annual American Control Conference*, pages 1027–1032.

Hassidim, A. and Singer, Y. (2017). Robust guarantees of stochastic greedy algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1424–1432. PMLR.

Iyer, R. and Bilmes, J. (2012). Algorithms for approximate minimization of the difference between submodular functions, with applications. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 407–417. AUAI Press.

Iyer, R. and Bilmes, J. (2019). A memoization framework for scaling submodular optimization to large scale problems. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89, pages 2340–2349. PMLR.

Ji, S., Xu, D., Li, M., Wang, Y., and Zhang, D. (2020). Stochastic greedy algorithm is still good: Maximizing submodular + supermodular functions. In *Optimization of Complex Systems: Theory, Models, Algorithms and Applications*, pages 488–497. Springer International Publishing.

Khanna, R., Elenberg, E. R., Dimakis, A. G., Ghosh, J., and Negahban, S. (2017). On approximation guarantees for greedy low rank optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1837–1846. PMLR.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes:

Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9(Feb):235–284.

Kuhnle, A. (2019). Interlaced greedy algorithm for maximization of submodular functions in nearly linear time. In *Advances in Neural Information Processing Systems 32*, pages 2371–2381. Curran Associates, Inc.

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

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429. ACM.

Lin, H. and Bilmes, J. (2010). Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics.

Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. In *Proceedings of the 8th IFIP Conference on Optimization Techniques*, pages 234–243. Springer.

Mirzasoleiman, B., Badanidiyuru, A., and Karbasi, A. (2016). Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1358–1367. PMLR.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. (2015). Lazier than lazy greedy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 1812–1818. AAAI Press.

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

Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., and Moore, J. H. (2017). PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10(1):36.

Pan, X., Jegelka, S., Gonzalez, J. E., Bradley, J. K., and Jordan, M. I. (2014). Parallel double greedy submodular maximization. In *Advances in Neural Information Processing Systems 27*, pages 118–126. Curran Associates, Inc.

Qian, C., Yu, Y., and Tang, K. (2018). Approximation guarantees of stochastic greedy algorithms for subset selection. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1478–1484. International Joint Conferences on Artificial Intelligence Organization.

Sharma, D., Kapoor, A., and Deshpande, A. (2015). On greedy maximization of entropy. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1330–1338. PMLR.

Song, H. O., Jegelka, S., Rathod, V., and Murphy, K. (2017). Deep metric learning via facility location. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2206–2214.

Vondrák, J. (2013). Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304.

Wei, K., Iyer, R., and Bilmes, J. (2014). Fast multi-stage submodular maximization. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1494–1502. PMLR.