Nearly Linear-Time, Deterministic Algorithm for Maximizing (Non-Monotone) Submodular Functions Under Cardinality Constraint

Anonymous Author(s)
Affiliation
Address
email

Abstract

We develop two deterministic approximation algorithms for the maximization of non-monotone submodular functions under cardinality constraint: both are based upon the novel idea of interlacing two greedy procedures. Our algorithm FastInterlaceGreedy uses interlaced, thresholded greedy procedures to obtain ratio $1/4 - \varepsilon$ in $O\left(\frac{n}{\varepsilon}\log\left(\frac{n}{\varepsilon}\right)\right)$ queries of the objective, which improves upon both the ratio and the quadratic time complexity of the previously fastest deterministic algorithm for this problem. We validate our algorithms in the context of two applications of non-monotone submodular maximization, on which FastInterlaceGreedy outperforms the fastest deterministic and randomized algorithms in prior literature.

1 Introduction

Because of sundry applications, the maximization of a nonnegative submodular[1] function with respect to a cardinality constraint (MCC) has a long history of study (Nemhauser et al., 1978). Applications of MCC include viral marketing (Kempe et al., 2003), network monitoring (Leskovec et al., 2007), video summarization (Mirzasoleiman et al., 2018), and MAP Inference for Determinental Point Processes (Gillenwater et al., 2012), among many others. In recent times, the amount of data generated by many applications has been increasing exponentially; therefore, linear or sublinear-time algorithms are needed.

When f is monotone, greedy approaches for MCC have proven effective and nearly optimal, both in terms of query complexity and approximation factor: subject to a cardinality constraint k, a simple greedy algorithm gives a $(1 - 1/e)$ approximation ratio in $O(kn)$ queries (Nemhauser et al., 1978), where n is the size of the instance. Furthermore, this ratio is optimal under the value oracle model (Nemhauser and Wolsey, 1978). Badanidiyuru and Vondrák (2014) sped up the greedy algorithm to require $O(n \log n)$ queries while sacrificing only a small $\varepsilon > 0$ in the approximation ratio, while Mirzasoleiman et al. (2015) developed a randomized $(1 - 1/e - \varepsilon)$ approximation in $O(n/\varepsilon)$ queries.

When f is non-monotone, the situation is very different; no subquadratic deterministic algorithm has yet been developed. Although a linear-time, randomized $(1/e - \varepsilon)$-approximation has been developed by Buchbinder et al. (2015), which requires $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$ queries, the performance guarantee of this algorithm holds only in expectation. A derandomized version of the algorithm with ratio $1/e$ has been developed by Buchbinder and Feldman (2018a) but has time complexity $O(k^3 n)$. Therefore, in this work, an emphasis is placed upon the development of nearly linear-time, deterministic approximation algorithms.

[1]For technical definitions of terms used in the Introduction, the reader is referred to Section 1.

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

Table 1: Fastest algorithms for cardinality constraint

Algorithm	Ratio	Time complexity	Deterministic?
FastInterlaceGreedy (Alg. 2)	$1/4 - \varepsilon$	$O\left(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon}\right)$	Yes
Gupta et al. (2010)	$1/6 - \varepsilon$	$O\left(nk + \frac{n}{\varepsilon}\right)$	Yes
Buchbinder et al. (2015)	$1/e - \varepsilon$	$O\left(\frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon}\right)$	No

Contributions

We provide the deterministic approximation algorithm InterlaceGreedy (Alg. 1) for maximization of a submodular function subject to a cardinality constraint (MCC). InterlaceGreedy achieves ratio $1/4$ in $O(kn)$ queries to the objective function. We then speed the algorithm up in FastInterlaceGreedy (Alg. 2) to acheive ratio $(1/4 - \varepsilon)$ in $O\left(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon}\right)$ queries. In Table 1, we show the relationship to the fastest deterministic and randomized algorithms for MCC in prior literature.

Both algorithms operate by interlacing two greedy procedures together in a novel manner; that is, the two greedy procedures alternately select elements into disjoint sets and are disallowed from selection of the same element. We demonstrate this technique first with the interlacing of two standard greedy procedures in InterlaceGreedy, before interlacing thresholded greedy procedures developed by Badanidiyuru and Vondrák (2014) for monotone submodular functions to obtain the algorithm FastInterlaceGreedy.

Our algorithms are validated in the context of cardinality-constrained maximum cut and social network monitoring, which are both instances of MCC. In this evaluation, FastInterlaceGreedy is more than an order of magnitude faster than the fastest deterministic algorithm (Gupta et al., 2010) and is both faster and obtains better solution quality than the fastest randomized algorithm (Buchbinder et al., 2015). The anonymized source code to reproduce the evaluation is available at `https://gofile.io/?c=ChYSOQ`.

Organization The rest of this paper is organized as follows. Related work and preliminaries on submodular optimization are discussed in the rest of this section. In Section 2, InterlaceGreedy and FastInterlaceGreedy are presented and analyzed. Experimental validation is provided in Section 3.

Related Work

The literature on submodular optimization comprises many works. In this section, a short review of relevant techniques is given for MCC; that is, maximization of non-monotone, submodular functions over a ground set of size n with cardinality constraint k. For further information on other types of submodular optimization, interested readers are directed to the survey of Buchbinder and Feldman (2018b) and references therein.

A deterministic local search algorithm was developed by Lee et al. (2010), which achieves ratio $1/4 - \varepsilon$ in $O(n^4 \log n)$ queries. This algorithm runs two approximate local search procedures in succession. By contrast, our algorithm FastInterlaceGreedy employs interlacing of greedy procedures to obtain the same ratio in $O\left(\frac{n}{\varepsilon} \log \frac{n}{\varepsilon}\right)$ queries.

Gupta et al. (2010) developed a deterministic, iterated greedy approach, wherein two greedy procedures are run in succession and an algorithm for unconstrained submodular maximization are employed. This approach requires $O(nk)$ queries and has ratio $1/(4 + \alpha)$, where α is the inverse ratio of the employed subroutine for unconstrained, non-monotone submodular maximization; under the value query model, the smallest possible value for α is 2, as shown by Feige et al. (2011), so this ratio is at most $1/6$. The iterated greedy approach of Gupta et al. (2010) first runs one standard greedy algorithm to completion, then starts a second standard greedy procedure; this differs from our interlacing procedure which runs two greedy procedures concurrently and alternates between the selection of elements. The algorithm of Gupta et al. (2010) is experimentally compared to FastInterlaceGreedy in Section 3. The iterated greedy approach of Gupta et al. (2010) was extended and analyzed under more general constraints by a series of works: Mirzasoleiman et al. (2016); Feldman et al. (2017); Mirzasoleiman et al. (2018).

An elegant randomized greedy algorithm of Buchbinder et al. (2014) achieves expected ratio $1/e$ in $O(kn)$ queries for MCC; this algorithm was derandomized by Buchbinder and Feldman (2018a), but the derandomized version requires $O\left(k^3 n\right)$ queries. The randomized version was sped up in Buchbinder et al. (2015) to achieve expected ratio $1/e - \varepsilon$ and require $O\left(\frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon}\right)$ queries. Although this algorithm has better time complexity than FastInterlaceGreedy, the ratio of $1/e - \varepsilon$ holds only in expectation, which is much weaker than a deterministic approximation ratio. We compare experimentally with the algorithm of Buchbinder et al. (2015) in Section 3.

Recently, an improvement in the adaptive complexity of MCC was made by Balkanski et al. (2018). Their algorithm, BLITS, requires $O\left(\log^2 n\right)$ adaptive rounds of queries to the objective, where the queries within each round are independent of one another and thus can be parallelized easily. Previously the best adaptivity was the trivial $O(n)$. However, each round requires $\Omega(OPT^2)$ samples to approximate expectations, which for the applications we evaluated in Section 3 is $\Omega(n^4)$. For this reason, BLITS is evaluated as a heuristic in comparison with our algorithms in Section 3.

Currently, the best approximation ratio of any algorithm for MCC is 0.385 of Buchbinder and Feldman (2016). Their algorithm also works under a more general constraint than cardinality constraint; namely, a matroid constraint. This algorithm is the latest in a series of works (e.g. (Naor and Schwartz, 2011; Ene and Nguyen, 2016)) using the multilinear extension of a submodular function, which is expensive to evaluate.

Preliminaries

Given $n \in \mathbb{N}$, the notation $[n]$ is used for the set $\{0, 1, \ldots, n-1\}$. In this work, functions f with domain all subsets of a finite set are considered; hence, without loss of generality, the domain of the function f is taken to be $2^{[n]}$, which is all subsets of $[n]$. A nonnegative function $f : 2^{[n]} \to \mathbb{R}^+$ is *submodular* iff for all $A, B \subseteq [n]$, $x \in [n] \setminus B$, such that $A \subseteq B$, it holds that $f\left(B \cup x\right) - f(B) \leq f\left(A \cup x\right) - f(A)$.

In this work, the problem studied is to maximize a submodular function under a cardinality constraint (MCC), which is formally defined as follows. Let $f : 2^n \to \mathbb{R}^+$ be submodular; let $k \in [n]$. Determine $A \subseteq [n]$ such that $|A| \leq k$ and for all B such that $|B| \leq k$, $f(B) \leq f(A)$. An instance of MCC is the pair (f, k); however, rather than an explicit description of f, the function f is considered to be a value oracle; f may be queried on any set $A \subseteq [n]$ to yield $f(A)$. The efficiency or runtime of an algorithm is measured by the number of queries made to the oracle f.

Finally, without loss of generality, instances of MCC considered in the following satisfy $n \geq 4k$. If this condition does not hold, the function may be extended to $[m]$ by adding dummy elements to the domain which do not change the function value. That is, the function $g : 2^m \to \mathbb{R}^+$ is defined as $g(A) = f(A \cap [n])$; it may be easily checked that g remains submodular, and any possible solution to the MCC instance (g, k) maps[2] to a solution of (f, k) of the same value. Hence, the ratio of any solution to (g, k) to the optimal is the same as the ratio of the mapped solution to the optimal on (f, k).

2 Approximation Algorithms

In this section, we present the approximation algorithms based upon interlacing greedy procedures. In Section 2.1, the technique is demonstrated with standard greedy procedures in algorithm InterlaceGreedy. In Section 2.2, the nearly linear-time algorithm FastInterlaceGreedy is introduced.

2.1 The InterlaceGreedy Algorithm

In this section, the InterlaceGreedy algorithm (InterlaceGreedy, Alg. 1) is introduced. InterlaceGreedy takes as input an instance of MCC and outputs a set C, which approximates $\max_{|X| \leq k} f(X)$.

InterlaceGreedy operates by interlacing two standard ;j;greedy procedures. This interlacing is accomplished by maintaining two disjoint sets A and B, which are initially empty. For k iterations, the element $a \notin B$ with the highest marginal gain with respect to A is added to A, followed by an analogous greedy selection for B; that is, the element $b \notin A$ with the highest marginal gain with respect to B is added to B. After the first set of interlaced greedy procedures complete, a modified

[2]The mapping is to discard all elements greater than n.

Algorithm 1 InterlaceGreedy (f, k): The InterlaceGreedy Algorithm

```
Input: f : 2^[n] → ℝ^+, k ∈ [n]
Output: C ⊆ 2^[n], such that |C| ≤ k.
A_0 ← B_0 ← ∅
for i ← 0 to k − 1 do
    a_i ← arg max_{x ∈ 2^[n] \ (A_i ∪ B_i)} f_x(A_i)
    A_{i+1} ← A_i + a_i
    b_i ← arg max_{x ∈ 2^[n] \ (A_{i+1} ∪ B_i)} f_x(B_i)
    B_{i+1} ← B_i + b_i
D_1 ← E_1 ← {a_0}
for i ← 1 to k − 1 do
    d_i ← arg max_{x ∈ 2^[n] \ (D_i ∪ E_i)} f_x(D_i)
    D_{i+1} ← D_i + d_i
    e_i ← arg max_{x ∈ 2^[n] \ (D_{i+1} ∪ E_i)} f_x(E_i)
    E_{i+1} ← E_i + e_i
return C ← arg max{f(A_i), f(B_i), f(D_i), f(E_i) : i ∈ [k + 1]}
```

version is repeated sets D, E, which are initialized to the maximum-value singleton $\{a_0\}$. Finally, the algorithm returns the set with the maximum f-value of any query the algorithm has made to f.

If f is submodular, InterlaceGreedy has an approximation ratio of $1/4$ and query complexity $O(kn)$; the deterministic algorithm of Gupta et al. (2010) has the same time complexity to achieve ratio $1/6$. The full proof of Theorem 1 is provided in Appendix A.

Theorem 1. *Let $f : 2^{[n]} \to \mathbb{R}^+$ be submodular, let $k \in [n]$, let $O = \arg\max_{|S| \leq k} f(S)$, and let $C =$InterlaceGreedy (f, k). Then*

$$f(C) \geq f(O)/4,$$

and InterlaceGreedy makes $O(kn)$ queries to f.

Proof sketch. The main idea of the proof is to exploit the fact that if S and T are disjoint,

$$f(O \cup S) + f(O \cup T) \geq f(O) + f(O \cup S \cup T), \tag{1}$$

which is a consequence of the submodularity of f. Thus, if $f(S) \geq \alpha f(O \cup S)$ and $f(T) \geq \beta f(O \cup T)$, $\max_{X \in \{S,T\}} f(X) \geq (\alpha + \beta) f(O)/4$. Hence the proof proceeds by bounding $f(A) \geq f(O \cup A)/2$ and $f(B) \geq f((O \setminus \{a_0\}) \cup B)/2$. This is accomplished by an adaptation of the proof that the greedy algorithm is a $(1/2)$-approximation for monotone submodular maximization with respect to a matroid constraint (Fisher et al., 1978): the adaptation requires a re-ordering that is not possible subject to a general matroid constraint but is possible with the cardinality constraint considered here. Because of the way the re-ordering works, it is only possible to show that $f(B) \geq f((O \setminus \{a_0\}) \cup B)/2$, instead of the desired $f(B) \geq f(O \cup B)/2$. Hence, a second greedy interlacing is required, starting both sets from $\{a_0\}$, to produce D, E such that $f(D) \geq f(O \cup D)/2$ and $f(E) \geq f(O \cup E)/2$, with $f(O \cup D) + f(O \cup E) \geq f(O \cup \{a_0\})$ by submodularity. Finally, the argument concludes by noticing that either $a_0 \in O$ or $a_0 \notin O$. □

2.2 The FastInterlaceGreedy Algorithm

In this section, we provide a faster interlaced greedy algorithm (FastInterlaceGreedy (`FIG`), Alg. 2), which requires $O(n \log n)$ queries. As input, an instance (f, k) of MCC is taken, as well as a parameter $\delta > 0$.

The algorithm `FIG` works as follows. As in InterlaceGreedy, there is a repeated interlacing of two greedy procedures. However, to ensure a faster query complexity, these greedy procedures are thresholded: a separate threshold τ is maintained for each of the greedy procedures. The interlacing is accomplished by alternating calls to the `ADD` subroutine (Alg. 3), which adds a single element and is described below. When all of the thresholds fall below the value $\delta M/n$, the maximum of the greedy solutions is returned; here, $\delta > 0$ is the input parameter, M is the maximum value of a singleton, and n is the size of the ground set.

Algorithm 2 `FIG` (f, k, δ): The FastInterlaceGreedy Algorithm

Input: $f : 2^{[n]} \to \mathbb{R}^+$, $k \in [n]$
Output: $C \subseteq 2^{[n]}$, such that $|C| \leq k$.
$A_0 \leftarrow B_0 \leftarrow \emptyset$
$M \leftarrow \tau_A \leftarrow \tau_B \leftarrow \max_{x \in [n]} f(x)$
$i \leftarrow -1$, $a_{-1} \leftarrow 0$, $b_{-1} \leftarrow 0$
while $\tau_A \geq \varepsilon M/n$ or $\tau_B \geq \varepsilon M/n$ **do**
  $(a_{i+1}, \tau_A) \leftarrow \texttt{ADD}(A, B, a_i, \tau_A)$
  $(b_{i+1}, \tau_B) \leftarrow \texttt{ADD}(B, A, b_i, \tau_B)$
  $i \leftarrow i + 1$
$D_1 \leftarrow E_1 \leftarrow \{a_0\}$, $\tau_D \leftarrow \tau_E \leftarrow M$
$i \leftarrow 0$, $d_0 \leftarrow 0$, $e_0 \leftarrow 0$
while $\tau_D \geq \varepsilon M/n$ or $\tau_E \geq \varepsilon M/n$ **do**
  $(d_{i+1}, \tau_D) \leftarrow \texttt{ADD}(D, E, d_i, \tau_D)$
  $(e_{i+1}, \tau_E) \leftarrow \texttt{ADD}(E, D, e_i, \tau_E)$
  $i \leftarrow i + 1$
return $C \leftarrow \arg\max\{f(A), f(B), f(D), f(E)\}$

Algorithm 3 `ADD` (S, T, j, τ): The `ADD` subroutine

Input: Two sets $S, T \subseteq [n]$, element $j \in [n]$, $\tau \in \mathbb{R}^+$
Output: (i, τ), such that $i \in [n]$, $\tau \in \mathbb{R}^+$
if $|S| = k$ **then**
  return $(0, (1 - \delta)\tau)$
while $\tau \geq \varepsilon M/n$ **do**
  for $(x \leftarrow j; x < n; x \leftarrow x + 1)$ **do**
    if $x \notin T$ **then**
      if $f_x(S) \geq \tau$ **then**
        $S \leftarrow S \cup \{x\}$
        return (x, τ)
  $\tau \leftarrow (1 - \delta)\tau$
  $j \leftarrow 0$
return $(0, \tau)$

The `ADD` subroutine is responsible for adding a single element above the input threshold and decreasing the threshold. It takes as input four parameters: two sets S, T, element j, and threshold τ; furthermore, `ADD` is given access to the oracle f, the budget k, and the parameter δ of `FIG`. As an overview, `ADD` adds the first[3] on the element $x > j$, such that $x \notin T$ and such that the marginal gain $f_x(S)$ is at least τ. If no such element $x > j$ exists, the threshold is decreased by a factor of $(1 - \delta)$ and the process is repeated (with j set to 0). When such an element x is found, the element x is added to S, and the new threshold value and position x are returned. Finally, `ADD` ensures that the size of S does not exceed k.

Next, we prove the approximation ratio of `FIG`.

Theorem 2. *Let $f : 2^{[n]} \to \mathbb{R}^+$ be submodular, let $k \in [n]$, and let $\varepsilon > 0$. Let $O = \arg\max_{|S| \leq k} f(S)$. Choose δ such that $(1 - 6\delta)/4 > 1/4 - \varepsilon$, and let $C =$ `FIG` (f, k, δ). Then*

$$f(C) \geq (1 - 6\delta) f(O)/4 \geq (1/4 - \varepsilon)\, f(O).$$

Proof. Let A, B, C, D, E, M have their values at termination of $\texttt{FIG}(f, k, \delta)$. Let $A = \{a_0, \ldots, a_{|A|-1}\}$ be ordered by addition of elements by `FIG` into A. The proof requires the following four inequalities:

$$f(O \cup A) \leq (2 + 2\delta) f(A) + \delta M, \tag{2}$$
$$f((O \setminus \{a_0\}) \cup B) \leq (2 + 2\delta) f(B) + \delta M, \tag{3}$$
$$f(O \cup D) \leq (2 + 2\delta) f(D) + \delta M, \tag{4}$$
$$f(O \cup E) \leq (2 + 2\delta) f(E) + \delta M. \tag{5}$$

[3] The first element $x > j$ in the natural ordering on $[n] = \{0, \ldots, n - 1\}$.

Once these inequalities have been established, Inequalities 2, 3, submodularity of f, and $A \cap B = \emptyset$ imply

$$f(O \setminus \{a_0\}) \leq 2(1+\delta)(f(A)+f(B)) + 2\delta M. \tag{6}$$

Similarly, from Inequalities 4, 5, submodularity of f, and $D \cap E = \{a_0\}$, it holds that

$$f(O \cup \{a_0\}) \leq 2(1+\delta)(f(D)+f(E)) + 2\delta M. \tag{7}$$

Hence, from the fact that either $a_0 \in O$ or $a_0 \notin O$ and the definition of C, it holds that

$$f(O) \leq 4(1+\delta)f(C) + 2\delta M.$$

Since $f(C) \leq f(O)$ and $M \leq f(O)$, the theorem is proved.

The proofs of Inequalities 2–5 are similar. The proof of Inequality 3 is given here, while the proofs of the others are provided in Appendix B.

Proof of Inequality 3. Let $A = \{a_0, \ldots, a_{|A|-1}\}$ be ordered as specified by `FIG`. Likewise, let $B = \{b_0, \ldots, b_{|B|-1}\}$ be ordered as specified by `FIG`.

Lemma 1. *$O \setminus (B \cup \{a_0\}) = \{o_0, \ldots, o_{l-1}\}$ can be ordered such that*

$$f_{o_i}(B_i) \leq (1+2\delta) f_{b_i}(B_i), \tag{8}$$

for any $i \in [|B|]$.

Proof. For each $i \in [|B|]$, define τ_{B_i} to be the value of τ when b_i was added to B in the `ADD` subroutine. Order $o \in (O \setminus (B \cup \{a_0\})) \cap A = \{o_0, \ldots, o_{\ell-1}\}$ by the order in which these elements were added into A. Order the remaining elements of $O \setminus (B \cup \{a_0\})$ arbitrarily. Then, when b_i was chosen by `ADD`, it holds that $o_i \notin A_{i+1}$, since $A_1 = \{a_0\}$ and $a_0 \notin O \setminus (B \cup \{a_0\})$. Also, it is true $o_i \notin B_i$; hence o_i was not added into some (possibly non-proper) subset B_i' of B_i at the previous threshold value $\frac{\tau_{B_i}}{(1-\delta)}$. Hence $f_{o_i}(B_i) \leq f_{o_i}(B_i') < \frac{\tau_{B_i}}{(1-\delta)}$, since $o_i \notin A_{i+1}$. Since $f_{b_i}(B_i) \geq \tau_{B_i}$ and $\delta < 1/2$, inequality (8) follows.

Order $\hat{O} = O \setminus (B \cup \{a_0\}) = \{o_0, \ldots, o_{l-1}\}$ as indicated in the proof of Lemma 1, and let $\hat{O}_i = \{o_0, \ldots, o_{i-1}\}$, if $i \geq 1$, $\hat{O}_0 = \emptyset$. Then

$$\begin{aligned}
f(\hat{O} \cup B) - f(B) &= \sum_{i=0}^{l-1} f_{o_i}(\hat{O}_i \cup B) \\
&= \sum_{i=0}^{|B|-1} f_{o_i}(\hat{O}_i \cup B) + \sum_{i=|B|}^{l-1} f_{o_i}(\hat{O}_i \cup B) \\
&\leq \sum_{i=0}^{|B|-1} f_{o_i}(B_i) + \sum_{i=|B|}^{l-1} f_{o_i}(B) \\
&\leq \sum_{i=0}^{|B|-1} (1+2\delta) f_{b_i}(B_i) + \sum_{i=|B|}^{l-1} f_{o_i}(B) \\
&\leq (1+2\delta) f(B) + \delta M,
\end{aligned}$$

where any empty sum is defined to be 0; the first inequality follows by submodularity, the second follows from Lemma 1, and the third follows from the definition of B, and the facts that $\max_{x \in [n] \setminus A_{|B|+1}} f_x(B) < \varepsilon M/n$, $l - |B| \leq k$, and $o_i \notin A_{|B|+1}$, for $|B| \leq i < l$.

Theorem 3. *Let $f : 2^{[n]} \to \mathbb{R}^+$ be submodular, let $k \in [n]$, and let $\delta > 0$. Then the number of queries to f by `FIG`(f, k, δ) is at most $O\left(\frac{n}{\delta} \log \frac{n}{\delta}\right)$.*

Proof. Recall $[n] = \{0, 1, \ldots, n-1\}$. Let $S \in \{A, B, D, E\}$, and $S = \{s_0, \ldots, s_{|S|-1}\}$ in the order in which elements were added to S. When `ADD` is called by `FIG` to add an element $s_i \in [n]$ to S, if the value of τ is the same as the value when s_{i-1} was added to S, then $s_i > s_{i-1}$. Finally, once `ADD` queries the marginal gain of adding $(n-1)$, the threshold is revised downward by a factor of $(1-\delta)$.

Therefore, there are at most $O(n)$ queries of f at each distinct value of $\tau_A, \tau_B, \tau_D, \tau_E$. Since at most $O(\frac{1}{\delta} \log \frac{n}{\delta})$ values are assumed by each of these thresholds, the theorem follows. $\square$

3 Experimental Evaluation

In this section, performance of FastInterlaceGreedy (`FIG`) is compared with that of state-of-the-art algorithms on two applications of submodular maximization: cardinality-constrained maximum cut and network monitoring.

3.1 Setup

Algorithms We compare the following algorithms. Source code for the evaluated implementations of all algorithms is available at `https://gofile.io/?c=ChYSOQ`.

- **FastInterlaceGreedy (Alg. 2)**: `FIG` is implemented as specified in the pseudocode, with the following addition: a stealing procedure is employed at the end, which uses submodularity to quickly steal[4] elements from A, B, D, E into C in $O(k)$ queries. This does not impact the performance guarantee, as the value of C can only increase. The parameter δ is set to 0.1, yielding approximation ratio of 0.1.
- **Gupta et al. (2010)**: The algorithm of Gupta et al. (2010) for cardinality constraint; as the subroutine for the unconstrained maximization subproblems, the deterministic, linear-time $1/3$-approximation algorithm of Buchbinder et al. (2012) is employed. This yields an overall approximation ratio of $1/7$ for the implementation used herein. This algorithm is the fastest determistic approximation algorithm in prior literature.
- **FastRandomGreedy (FRG)**: The $O\left(\frac{n}{\varepsilon^2} \ln \frac{1}{\varepsilon}\right)$ randomized algorithm of Buchbinder et al. (2015) (Alg. 4 of that paper), with expected ratio $1/e - \varepsilon$; the parameter ε was set to 0.3, yielding expected ratio of ≈ 0.07 as evaluated herein. This algorithm is the fastest randomized approximation algorithm in prior literature.
- **BLITS**: The $O\left(\log^2 n\right)$-adaptive algorithm recently introduced in Balkanski et al. (2018); the algorithm is employed as a heuristic without performance ratio, with the same parameter choices as in Balkanski et al. (2018). In particular, $\varepsilon = 0.3$ and 30 samples are used to approximate the expections. Also, a bound on OPT is guessed in logarithmically many iterations as described in Balkanski et al. (2018) and references therein.

Results for randomized algorithms are the mean of 10 trials, and the standard deviation is represented in plots by a shaded region.

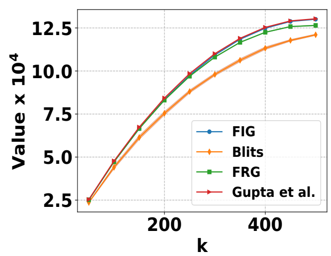
Applications Many applications with non-monotone, submodular objective functions exist. In this section, two applications are chosen to demonstrate the performance of the evaluated algorithms.

- Cardinality-Constrained Maximum Cut: The archetype of a submodular, non-monotone function is the maximum cut objective: given graph $G = (V, E)$, $S \subseteq V$, $f(S)$ is defined to be the number of edges crossing from S to $V \setminus S$. In this evaluation, we consider the cardinality constrained version of this problem.
- Social Network Monitoring: Given an online social network, suppose it is desired to choose k users to monitor, such that the maximum amount of content is propagated through these users. Suppose the amount of content propagated between two users u, v is encoded as weight $w(u, v)$. Then $f(S) = \sum_{u \in S, v \notin S} w(u, v)$.

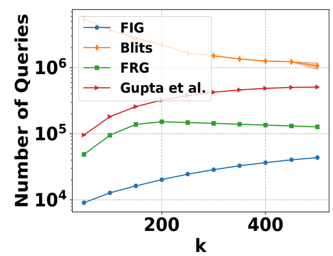
[4] Details of the stealing procedure are given in Appendix C.

3.2 Results

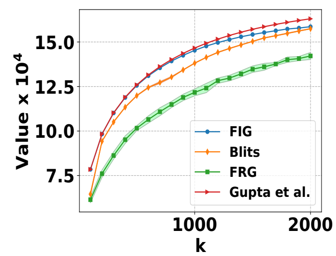
In this section, results are presented for the algorithms on the two applications. In overview: in terms of objective value, `FIG` and Gupta et al. (2010) were about the same and outperformed BLITS and FRG. Meanwhile, `FIG` was the fastest algorithm by the metric of queries to the objective and was faster than Gupta et al. (2010) by at least an order of magnitude.



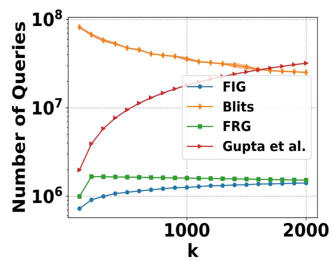
(a) ER, Cut Value



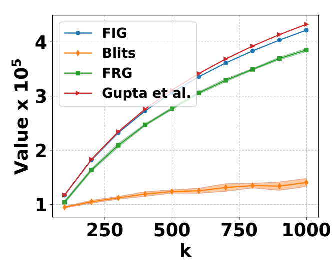
(b) ER, Function Queries



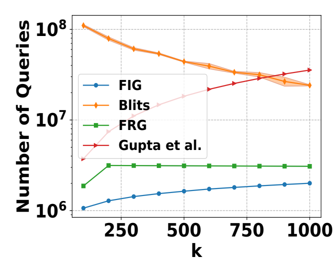
(c) BA, Cut Value



(d) BA, Function Queries



(e) Total content monitored versus budget k



(f) Number of Queries versus budget k

Figure 1: **(a)–(d)**: Objective value and runtime for cardinality-constrained maxcut on random graphs. **(e)–(f)**: Objective value and runtime for cardinality-constrained maxcut on ca-AstroPh with simulated amounts of content between users. In all plots, the x-axis shows the budget k.

Cardinality Constrained MaxCut For these experiments, two random graph models were employed: an Erdős-Rényi (ER) random graph with $1,000$ nodes and edge probability $p = 1/2$, and a Barabási–Albert (BA) graph with $n = 10,000$ and $m = m_0 = 100$.

On the ER graph, results are shown in Figs. 1(a) and 1(b); the results on the BA graph are shown in Figs. 1(c) and 1(d). In terms of cut value, the algorithm of Gupta et al. (2010) performed the best, although the value produced by `FIG` was nearly the same. On the ER graph, the next best was FRG followed by BLITS; whereas on the BA graph, BLITS outperformed FRG in cut value. In terms of efficiency of queries, `FIG` used the smallest number on every evaluated instance, although the number did increase logarithmically with budget. The number of queries used by FRG was higher, but after a certain budget remained constant. The next most efficient was Gupta et al. (2010) followed by BLITS.

Social Network Monitoring For the social network monitoring application, the citation network ca-AstroPh from the SNAP dataset collection was used, with $n = 18,772$ users and $198,110$ edges. Edge weights, which represent the amount of content shared between users, were generated uniformly randomly in $[1, 10]$. The results were similar qualitatively to those for the unweighted MaxCut problem presented previously. `FIG` is the most efficient in terms of number of queries, and `FIG` is only outperformed in solution quality by Gupta et al. (2010), which required more than an order of magnitude more queries.

References

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

Eric Balkanski, Adam Breuer, and Yaron Singer. Non-monotone submodular maximization in exponentially fewer iterations. In *NeurIPS*, 2018.

Niv Buchbinder and Moran Feldman. Constrained Submodular Maximization via a Non-symmetric Technique. 2016.

Niv Buchbinder and Moran Feldman. Deterministic Algorithms for Submodular Maximization. *ACM Transactions on Algorithms*, 14(3), 2018a.

Niv Buchbinder and Moran Feldman. Submodular Functions Maximization Problems – A Survey. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*. Second edition, 2018b.

Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. A Tight Linear Time (1 / 2) -Approximation for Unconstrained Submodular Maximization. In *FOCS*, pages 649–658, 2012.

Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *Symposium on Discrete Algorithms (SODA)*. ACM, 2014.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing Apples and Oranges: Query Tradeoff in Submodular Maximization. In *Symposium on Discrete Algorithms (SODA)*, pages 1149–1168. ACM-SIAM, 2015.

Alina Ene and Huy L. Nguyen. Constrained Submodular Maximization : Beyond 1 / e. In *FOCS*, pages 248–257, 2016.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing Non-Monotone Submodular Functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is Good: Near-Optimal Submodular Maximization via Greedy Optimization. In *COLT*, pages 1–26, 2017.

M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximations for maximizing submodular set functions-II. *Mathematical Programming*, 8:73–87, 1978.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-Optimal MAP Inference for Determinantal Point Processes. In *NIPS*, 2012.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, volume 6484 LNCS, pages 246–257, 2010.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.

Jon Lee, Vahab Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing Nonmonotone Submodular Functions under Matroid or Knapsack Constraints. *Siam Journal of Discrete Math*, 23 (4):2053–2078, 2010.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective Outbreak Detection in Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 420–429, 2007.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier Than Lazy Greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1812–1818, 2015.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast Constrained Submodular Maximization : Personalized Data Summarization. In *ICML*, 2016.

Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming Non-Monotone Submodular Maximization: Personalized Video Summarization on the Fly. In *AAAI*, pages 1379–1386, 2018.

Joseph Seffi Naor and Roy Schwartz. A Unified Continuous Greedy Algorithm for Submodular Maximization. pages 570–579, 2011.

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

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

A Proof of Theorem 1

Proof of Theorem 1.

Lemma 2.

$$4f(C) \geq f\left(O \setminus \{a_0\}\right).$$

Proof. Let $A = \arg\max_{i \in [k+1]} f(A_i)$. Let $\hat{O} = O \setminus A_k = \{o_0, \ldots, o_{l-1}\}$ be ordered such that for each $i \in [l]$, $o_i \notin B_i$; this ordering is possible since $B_0 = \emptyset$ and $l \leq k$. Also, for each $i \in [l]$, let $\hat{O}_i = \{o_0, \ldots, o_i\}$, and let $\hat{O}_0 = \emptyset$. Then

$$\begin{aligned} f(O \cup A_k) - f(A_k) &= \sum_{i=0}^{l-1} f_{o_i}(\hat{O}_i \cup A_k) \\ &\leq \sum_{i=0}^{l-1} f_{o_i}(A_i) \\ &\leq \sum_{i=0}^{l-1} f_{a_i}(A_i) = f(A_l), \end{aligned}$$

where the first inequality follows from submodularity, the second inequality follows from the greedy choice $a_i = \arg\max_{x \in 2^{[n]} \setminus (A_i \cup B_i)} f_x(A_i)$ and the fact that $o_i \notin B_i$. Hence

$$f(O \cup A_k) \leq f(A_l) + f(A_k) \leq 2f(A). \tag{9}$$

Let $B = \arg\max_{i \in [k+1]} f(B_i)$. Let $\hat{O} = O \setminus (\{a_0\} \cup B_k) = \{o_0, \ldots, o_{l-1}\}$ be ordered such that for each $i \in [l]$, $o_i \notin A_{i+1}$; this ordering is possible since $A_1 = \{a_0\}$, $a_0 \notin \hat{O}$, and $l \leq k$. Also, for each $i \in [l]$, let $\hat{O}_i = \{o_0, \ldots, o_i\}$, and let $\hat{O}_0 = \emptyset$. Then

$$\begin{aligned} f((O \setminus \{a_0\}) \cup B_k) - f(B_k) &= \sum_{i=0}^{l-1} f_{o_i}(\hat{O}_i \cup B_k) \\ &\leq \sum_{i=0}^{l-1} f_{o_i}(B_i) \\ &\leq \sum_{i=0}^{l-1} f_{b_i}(B_i) = f(B_l), \end{aligned}$$

where the first inequality follows from submodularity, the second inequality follows from the greedy choice $b_i = \arg\max_{x \in 2^{[n]} \setminus (A_{i+1} \cup B_i)} f_x(B_i)$ and the fact that $o_i \notin A_{i+1}$. Hence

$$f((O \setminus \{a_0\}) \cup B_k) \leq f(B_l) + f(B_k) \leq 2f(B). \tag{10}$$

By inequalities (9), (10), the fact that $A_k \cap B_k = \emptyset$, and submodularity, we have

$$f(O \setminus \{a_0\}) \leq f(O \cup A_k) + f((O \setminus \{a_0\} \cup B_k) \leq 2(f(A) + f(B)) \leq 4f(C).$$

□

Lemma 3.

$$4f(C) \geq f\left(O \cup \{a_0\}\right).$$

Proof. Let $D = \arg\max_{i \in [k+1]} f(A_i)$. Let $\hat{O} = O \setminus D_k = \{o_0, \ldots, o_{l-1}\}$ be ordered such that for each $i \in [l]$, $o_i \notin E_i$; this ordering is possible since $E_0 = \emptyset$ and $l \leq k$. Also, for each $i \in [l]$, let

$\hat{O}_i = \{o_0, \dots, o_i\}$, and let $\hat{O}_0 = \emptyset$. Then

$$\begin{aligned} f(O \cup D_k) - f(D_k) &= \sum_{i=0}^{l-1} f_{o_i}(\hat{O}_i \cup D_k) \\ &\leq \sum_{i=0}^{l-1} f_{o_i}(D_i) \\ &\leq \sum_{i=0}^{l-1} f_{d_i}(D_i) = f(D_l), \end{aligned}$$

where the first inequality follows from submodularity, the second inequality follows from the greedy choice $d_i = \arg\max_{x \in 2^{[n]} \setminus (D_i \cup E_i)} f_x(D_i)$ and the fact that $o_i \notin E_i$. Hence

$$f(O \cup D_k) \leq f(D_l) + f(D_k) \leq 2f(D). \tag{11}$$

Let $E = \arg\max_{i \in [k+1]} f(E_i)$. Let $\hat{O} = O \setminus E_k = \{o_0, \dots, o_{l-1}\}$ be ordered such that for each $i \in [l]$, $o_i \notin D_{i+1}$; this ordering is possible since $D_1 = \{a_0\}$, $a_0 \notin \hat{O}$ (since $a_0 \in E_k$), and $l \leq k$. Also, for each $i \in [l]$, let $\hat{O}_i = \{o_0, \dots, o_i\}$, and let $\hat{O}_0 = \emptyset$. Then

$$\begin{aligned} f(O \cup E_k) - f(E_k) &= \sum_{i=0}^{l-1} f_{o_i}(\hat{O}_i \cup E_k) \\ &\leq \sum_{i=0}^{l-1} f_{o_i}(E_i) \\ &\leq \sum_{i=0}^{l-1} f_{e_i}(E_i) = f(E_l), \end{aligned}$$

where the first inequality follows from submodularity, the second inequality follows from the greedy choices $e_0 = \arg\max_{x \in [n]} f(x)$, and if $i > 0$, $e_i = \arg\max_{x \in 2^{[n]} \setminus (D_{i+1} \cup E_i)} f_x(E_i)$ and the fact that $o_i \notin D_{i+1}$. Hence

$$f((O \cup E_k) \leq f(E_l) + f(E_k) \leq 2f(E). \tag{12}$$

By inequalities (11), (12), the fact that $D_k \cap E_k = \{a_0\}$, and submodularity, we have

$$f(O \cup \{a_0\}) \leq f(O \cup D_k) + f((O \cup E_k) \leq 2(f(D) + f(E)) \leq 4f(C).$$

□

The proof of the theorem follows from Lemmas 2, 3, and the fact that one of the statements $a_0 \in O$ or $a_0 \notin O$ must hold; hence, either $O \cup \{a_0\} = O$ or $O \setminus \{a_0\} = O$. □

B Proofs for Theorem 2

Proof of Inequality 2. Let $A = \{a_0, \dots, a_{|A|-1}\}$ be ordered as specified by `FIG`. Likewise, let $B = \{b_0, \dots, b_{|B|-1}\}$ be ordered as specified by `FIG`.

Lemma 4. *$O \setminus A = \{o_0, \dots, o_{l-1}\}$ can be ordered such that*

$$f_{o_i}(A_i) \leq (1 + 2\delta) f_{a_i}(A_i), \tag{13}$$

if $i \in [|A|]$.

Proof. Order $o \in (O \setminus A) \cap B = \{o_0, \dots, o_{\ell-1}\}$ by the order in which these elements were added into B. Order the remaining elements of $O \setminus A$ arbitrarily. Then, when a_i was chosen by `ADD`, it holds that $o_i \notin B_i$. Also, it is true $o_i \notin A_i$; hence o_i was not added into some (possibly non-proper) subset A'_i of A_i at the previous threshold value $\frac{\tau_{A_i}}{(1-\delta)}$. Hence $f_{o_i}(A_i) \leq f_{o_i}(A'_i) < \frac{\tau_{A_i}}{(1-\delta)}$, since $o_i \notin B_i$. Since $f_{a_i}(A_i) \geq \tau_{A_i}$ and $\delta < 1/2$, inequality (13) follows. □

Order $\hat{O} = O \setminus A = \{o_0, \ldots, o_{l-1}\}$ as indicated in the proof of Lemma 4, and let $\hat{O}_i = \{o_0, \ldots, o_{i-1}\}$, if $i \geq 1$, $\hat{O}_0 = \emptyset$. Then

$$\begin{aligned} f(O \cup A) - f(A) &= \sum_{i=0}^{l-1} f_{o_i}(\hat{O}_i \cup A) \\ &= \sum_{i=0}^{|A|-1} f_{o_i}(\hat{O}_i \cup A) + \sum_{i=|A|}^{l-1} f_{o_i}(\hat{O}_i \cup A) \\ &\leq \sum_{i=0}^{|A|-1} f_{o_i}(A_i) + \sum_{i=|A|}^{l-1} f_{o_i}(A) \\ &\leq \sum_{i=0}^{|A|-1} (1+2\delta) f_{a_i}(A_i) + \sum_{i=|A|}^{l-1} f_{o_i}(A) \\ &\leq (1+2\delta) f(A) + \delta M, \end{aligned}$$

where any empty sum is defined to be 0; the first inequality follows by submodularity, the second follows from Lemma 4, and the third follows from the definition of A, and the facts that $\max_{x \in [n] \setminus B_{|A|}} f_x(A) < \varepsilon M / n$ and $l - |A| \leq k$. □

Proof of Inequality 4. As in the proof of Inequality 2, it suffices to establish the following lemma. □

Lemma 5. *$O \setminus D = \{o_0, \ldots, o_{l-1}\}$ can be ordered such that*

$$f_{o_i}(D_i) \leq (1+2\delta) f_{d_i}(D_i), \tag{14}$$

for $i \in [|D|]$.

Proof. Order $o \in (O \setminus D) \cap E = \{o_0, \ldots, o_{\ell-1}\}$ by the order in which these elements were added into E. Order the remaining elements of $O \setminus D$ arbitrarily. Then, when d_i was chosen by `ADD`, it holds that $o_i \notin E_i$. Also, it is true $o_i \notin D_i$; hence o_i was not added into some (possibly non-proper) subset D'_i of D_i at the previous threshold value $\frac{\tau_{D_i}}{(1-\delta)}$. Hence $f_{o_i}(D_i) \leq f_{o_i}(D'_i) < \frac{\tau_{D_i}}{(1-\delta)}$, since $o_i \notin E_i$. Since $f_{d_i}(D_i) \geq \tau_{D_i}$ and $\delta < 1/2$, inequality (14) follows. □

Proof of Inequality 5. As in the proof of Inequality 2, it suffices to establish the following lemma.

Lemma 6. *$O \setminus E = \{o_0, \ldots, o_{l-1}\}$ can be ordered such that*

$$f_{o_i}(E_i) \leq (1+2\delta) f_{e_i}(E_i), \tag{15}$$

for $i \in [|E|]$.

Proof. Order $o \in (O \setminus E) \cap D = \{o_0, \ldots, o_{\ell-1}\}$ by the order in which these elements were added into D. Order the remaining elements of $O \setminus E$ arbitrarily. Then, when e_i was chosen by `ADD`, it holds that $o_i \notin D_{i+1}$, since $D_1 = \{a_0\}$ and $a_0 = d_0 \notin O \setminus E$. Also, it is true $o_i \notin E_i$; hence o_i was not added into some (possibly non-proper) subset E'_i of E_i at the previous threshold value $\frac{\tau_{E_i}}{(1-\delta)}$. Hence $f_{o_i}(E_i) \leq f_{o_i}(E'_i) < \frac{\tau_{E_i}}{(1-\delta)}$, since $o_i \notin D_{i+1}$. Since $f_{e_i}(E_i) \geq \tau_{E_i}$ and $\delta < 1/2$, inequality (15) follows. □

□

C Stealing Procedure for FastInterlaceGreedy

In this section, we describe an $O(k)$ procedure that may improve the quality of the solution found by FastInterlaceGreedy (a similar procedure could also be employed for InterlaceGreedy).

Let A, B, C, D, E have their values at the termination of FastInterlaceGreedy. Then calculate the sets $G = \{B_c = f(C) - f(C \setminus \{c\}) : c \in C\}$ and $H = \{A_x = f(C \cup \{x\}) - f(C) : x \in A \cup B \cup D \cup E\}$.

Then sort $G = (B_{c_1}, \ldots, B_{c_k})$ in non-decreasing order and sort $H = (A_{x_1}, \ldots, A_{x_l})$ in non-increasing order. Computing and sorting these sets requires $O(k \log k)$ time (and only $O(k)$ queries to f).

Finally, iterate through the elements of G in the sorted order, and if $B_{c_i} < A_{x_i}$ then C is assigned $C \setminus \{c_i\} \cup \{x_i\}$ if this assignment increases the value $f(C)$.