Streaming Based Bicriteria Approximation Algorithms for Submodular Optimization

Victoria G. Crawford[1]

[1]*vcrawford01@ufl.edu, Department of Computer and Information Science and Engineering, University of Florida*

Abstract

This paper proposes the optimization problem Submodular Cover (SC), which is to minimize the cost required to ensure that a non-monotone submodular benefit function exceeds a given threshold. Two algorithms are presented for SC that both give a $((1+\epsilon)(4/\epsilon^2+1), 1/2(1-\epsilon))$ bicriteria approximation guarantee to the problem. Both algorithms process the ground set in a stream, one in multiple passes. Further, a $(1/2(1-\epsilon), (1+\epsilon)(4/\epsilon^2+1))$ bicriteria approximation guarantee is given for the related optimization problem Submodular Knapsack (SK).

1 Introduction

A function $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ defined on subsets of a ground set U of size n is submodular if it possesses the following property: For all $A \subseteq B \subseteq U$ and $x \notin B$, $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$. Submodular set functions are found in many applications in data mining and machine learning including data summarization [Barezi *et al.*, 2019; Xu *et al.*, 2015; Tschiatschek *et al.*, 2014], influence maximization in a social network [Kempe *et al.*, 2003], dictionary selection [Das and Kempe, 2011], monitor placement [Leskovec *et al.*, 2007], as well as in classic optimization problems such as maximum weighted cut, set cover, and facility location. Further, many applications involve non-monotone[1] submodular functions [Barezi *et al.*, 2019; Xu *et al.*, 2015; Tschiatschek *et al.*, 2014].

The simplest optimization problem involving submodular functions is the NP-hard unconstrained submodular maximization problem, where we wish to find $\text{argmax}\{f(X) : X \subseteq U\}$. On the other hand, the constrained submodular maximization problem has received a lot of attention, where we maximize f subject to some constraint. For example, the cardinality constraint (find $\text{argmax}\{f(X) : X \subseteq U, |X| \leq \kappa\}$) [Buchbinder *et al.*, 2014], or a knapsack constraint ($\text{argmax}\{f(X) : X \subseteq U, \sum_{x \in X} w(x) \leq \kappa\}$) [Gupta *et al.*, 2010], both NP-hard.

However, in some applications it is desirable that rather than maximizing the submodular function, we only wish that the submodular function be *sufficiently good* and then minimize some other value. For example, in the data summarization application one may wish to find a sufficiently good summary (i.e. get a submodular function above a threshold) while minimizing the total memory of the summary. Motivated by this, we introduce here the optimization problem Submodular Cover (SC).

Definition 1 (Submodular Cover (SC))**.** *Let $f : 2^U \rightarrow \mathbb{R}_{\geq 0}$ be a submodular function defined over subsets of the universe U of size n such that $f(\emptyset) = 0$, and let $w : U \rightarrow \mathbb{R}_{\geq 0}$ be a cost function defined over the elements of the universe. The Submodular Cover problem (SC) is, given $\tau \leq \max\{f(X) : X \subseteq U\}$, find*

$$\textit{argmin}_{X \subseteq U} \left\{ \sum_{x \in X} w(x) : f(X) \geq \tau \right\}.$$

[1] f is monotone if for all $A \subseteq B$, $f(A) \leq f(B)$.

An instance of SC is written as SC(U, f, w, τ, OPT) where OPT is the cost of the optimum solution.

SC has been considered in the special setting where f is also assumed to be monotone [Wolsey, 1982a], but to the best of our knowledge has never been considered in the non-monotone setting. The topic of this paper is therefore to consider whether approximation algorithms can be developed for this problem, and further whether those algorithms can be made practical in the face of large data sets. In particular, the streaming model is considered: U is assumed to arrive in an arbitrary order, and the goal is to solve SC so that very few passes are made through the entire data set and in addition memory used at any point in time is limited.

1.1 Contributions

In particular, the contributions are:

(i) It is proven in Theorem 1 that we cannot approximate the constraint of SC better than $1/2$ in polynomially many queries of f. On the other hand, two bicriteria approximation algorithms that can yield constraint approximations of arbitrarily close to $1/2$ are presented (MULTI-PASS-COVER and SINGLE-PASS-COVER).

(ii) The algorithm MULTI-PASS-COVER is proposed for SC, which is a multi-pass streaming approximation algorithm with a bicriteria approximation guarantee of $((1+\epsilon)(4/\epsilon^2+1), \gamma(1-\epsilon))$, where γ is the approximation ratio of an unconstrained submodular maximization algorithm used as a subroutine (which can be 1/2 by using the algorithm of Buchbinder and Feldman [2018]). MULTI-PASS-COVER takes at most $\ln(OPT/w_{min})/\ln(1+\epsilon)$ passes through U in an arbitrary order, while storing elements of total weight at most $(1+\epsilon)(4/\epsilon^2+1)OPT$.

(iii) The algorithm SINGLE-PASS-COVER is proposed for SC, which takes a single pass through U in an arbitrary order and has a bicriteria approximation guarantee of $((1+\epsilon)(4/\epsilon^2+1), \gamma(1-\epsilon))$. The total weight of all elements stored at one time is at most $(1+\epsilon)(4/\epsilon^2+1)\ln(2B/(\epsilon\tau\xi))/\ln(1+\epsilon)B$, where $B \geq OPT$ is an input. SINGLE-PASS-COVER does not have a bound on the total weight of stored elements relative to OPT, which is in fact shown to be impossible. Instead, SINGLE-PASS-COVER has a bound on the total weight of stored elements relative to OPT_i, the optimum solution over the first i elements read in, provided it exists: Once the ith element of the universe has been read in by SINGLE-PASS-COVER, the total weight of all elements stored at one time at most $(1+\epsilon)(4/\epsilon^2+1)\ln(2OPT_i/(\epsilon\tau\xi))/\ln(1+\epsilon)OPT_i$ from that point on, where ξ is instance dependent.

(iv) A similar approach to STREAM can be taken for the related problem SK (Submodular Knapsack). In particular, the algorithm SINGLE-PASS-KNAPSACK is proposed for SK. SINGLE-PASS-KNAPSACK takes a single pass through U in an arbitrary order and has a bicriteria approximation guarantee of $(\gamma(1-\epsilon), (1+\epsilon)(4/\epsilon^2+1))$. The total weight of all stored elements is at most $(4/\epsilon^2+2/\epsilon)\ln(2\kappa/(w_{min}\epsilon))/\ln(1+\epsilon)\kappa$.

1.2 Notation and Definitions

The following notation and definitions will be used throughout the paper: (i) Define $w(X)$ for $X \subseteq U$ to be $\sum_{x\in X} w(x)$; (ii) Define $\Delta f(X,x) = f(X\cup\{x\}) - f(X)$; (iii) Define $w_{\min} = \min_{x\in U} w(x)$.

1.3 Related Work

Unconstrained Submodular Maximization

SC is related to the unconstrained submodular maximization problem. In particular, if $\tau = \max\{f(X) : X \subseteq U\}$, then SC produces a solution to the unconstrained submodular maximization problem. It was proven by Feige *et al.* [2011] that given any instance of unconstrained submodular maximization with optimum solution S^*, there is no algorithm using fewer than $e^{\epsilon^2 n/8}$ queries that always finds a solution of expected value at least $(1/2+\epsilon)f(S^*)$ for any $\epsilon > 0$ (even if the objective is assumed to have a symmetric function). This implies Theorem 1 stated in this paper.

A number of approximation algorithms have been proposed for unconstrained submodular maximization [Feige *et al.*, 2011; Buchbinder *et al.*, 2015; Dobzinski and Mor, 2015; Buchbinder and Feldman, 2018; Ene *et al.*, 2018; Chen *et al.*, 2019]. One approach is via local search algorithms [Feige *et al.*, 2011; Gharan and Vondrák, 2011; Dobzinski and Mor, 2015], of which the best deterministic approximation ratio is 0.4 [Dobzinski and Mor, 2015] and the best randomized is 0.41 [Gharan and Vondrák, 2011]. On the other hand, Buchbinder *et al.* [2015] proposed a deterministic 1/3, and a randomized 1/2 algorithm that are more of a greedy approach, both running in linear time. Interestingly, a random set is a 1/4 approximation in expectation [Feige *et al.*, 2011]. Recently, Buchbinder and Feldman [2018] introduced a method of de-randomizing algorithms, which could be applied to get a 1/2 guarantee in n^2 time, or alternatively a $1/2 - \epsilon$ guarantee in $O(n/\epsilon)$ time. Chen *et al.* [2019] introduced a constant adaptivity algorithm with a $1/2 - \epsilon$ randomized guarantee, based on the multilinear extension. Independently, the same result was found by Ene *et al.* [2018]. All of the unconstrained submodular maximization algorithms require the entire ground set to be stored in memory.

To the best of our knowledge, none of the above algorithms for unconstrained submodular cover have been shown to give an approximation guarantee for SC. For some of them, it is easy to see that they do not give a non-trivial approximation guarantee for SC: Both the local search algorithm of Feige *et al.* [2011] and the double greedy algorithm of Buchbinder *et al.* [2015] can return solutions that have n times the cost of that of the optimal.

Submodular Cover

Variants of Submodular Cover have been studied where f is assumed to be monotone [Wolsey, 1982b; Wan *et al.*, 2010; Crawford *et al.*, 2019]. In particular, the greedy algorithm produces a $1 + \ln(\alpha/\beta)$-approximate solution, where α and β are instance dependent parameters [Wolsey, 1982b]. In addition, a slightly modified greedy algorithm produces a $(\ln(1/\epsilon), 1 - \epsilon)$-bicriteria approximation ratio. To the best of our knowledge, a non-monotone version of submodular cover has never been proposed.

Monotone submodular cover with cardinality cost has been studied previously in the streaming setting [Norouzi-Fard *et al.*, 2016]. In particular, if an upper bound ϵM of the optimal solution S^* is given, the streaming algorithm makes a single pass through the data and returns a $(2/\epsilon, 1 - \epsilon)$ approximate solution, storing a maximum of M elements, and making at most $\mathcal{O}(M)$ function evaluations per received element. Alternatively, if $\ln(1/\epsilon)|S^*|$ passes are allowed through the data, the approximation ratio can be improved to a $(\ln(1/\epsilon), 1 - \epsilon)$-bicriteria approximation.

Iyer and Bilmes [2013] proposed algorithms where weighted cost could be extended to general monotone submodular cost function. It appears that our result could fit into their algorithm's framework, therefore could be used for more general cost functions.

Constrained Submodular Maximization

The constrained submodular maximization problem has been extensively studied with many constraints, both with monotone objectives [Nemhauser and Wolsey, 1978; Badanidiyuru *et al.*, 2014; Mirzasoleiman *et al.*, 2015], and non-monotone [Lee *et al.*, 2009; Gupta *et al.*, 2010; Feige *et al.*, 2011; Buchbinder *et al.*, 2014, 2017]. While the seminal greedy algorithm produces an optimal $1 - 1/e$-approximate solution for monotone submodular maximization subject to a cardinality constraint [Nemhauser and Wolsey, 1978], it does not produce any non-trivial guarantee for non-monotone objectives.

Submodular maximization and submodular cover are related: It was proven by Iyer and Bilmes that algorithms for submodular maximization can be used as a subroutine for algorithms for submodular cover (and vice versa). In particular, an (a, b)-bicriteria approximation algorithm for submodular maximization with a knapsack constraint that runs in time T can be used to get a $((1 + \epsilon)b, a)$-bicriteria approximation algorithm for submodular cover in time $T \ln(w(S^*)/w_{min})/\ln(1 + \epsilon)$. This is done by guessing $w(S^*)$ in order $w_{min}, (1 + \epsilon)w_{min}, ..., w(S^*)$ and running the submodular maximization algorithm with budget equal to each guess. For the case of uniform weights, the best currently known approximation guarantee for

submodular maximization is 0.385, using the multilinear extension, to the best of our knowledge [Buchbinder and Feldman, 2019]. In addition, it has been proven that in the value oracle model it is impossible to get a better approximation guarantee than 0.491 for uniform weights [Gharan and Vondrák, 2011]. Therefore this approach to solving SC is limited. Notice that SINGLE-PASS-KNAPSACK is a bicriteria approximation algorithm, and therefore does not necessarily produce a feasible solution and so can get approximation guarantee above 0.491.

Of particular interest is the algorithm of Gupta *et al.* [2010] for submodular maximization with a knapsack constraint, which is a greedy-like approach that yields a solutions that is $\alpha + 2e/(e-1)$-approximate, and a $(4+\alpha)$ approximate algorithm for uniform cost. The downside to using this algorithm to solve SC as described above is that the resulting solution would not be very close to feasible, and the algorithm is relatively slow for knapsack constraints. An alternative algorithm for submodular maximization with a cardinality constraint are randomized greedy approaches [Buchbinder *et al.*, 2014, 2017]. These give approximation guarantees in expectation about $1/e$, and therefore can be used to get significantly closer to a feasible solution for SC compared to Gupta *et al.* [2010], but only for uniform cost. Further, they are faster, and Buchbinder *et al.* [2017] runs in linear time.

One question is whether any of the greedy approaches can be extended to *keep going* and produce a solution that is closer to feasible for SC. This approach works when the objective is monotone. However, the non-monotonicity of the objective prevents this approach from working, and it is not clear that any greedy-like approach that works for submodular maximization can be extended to SC.

A number of algorithms have been proposed for constrained submodular maximization in the streaming setting [Chakrabarti and Kale, 2015; Alaluf *et al.*, 2020]. The algorithm of Alaluf *et al.* is especially related to those proposed in this paper because it uses a procedure where elements from the stream are stored in $O(1/\epsilon)$ disjoint sets, and then runs an offline algorithm on the result. Alaluf *et al.* provide a streaming algorithm for cardinality constrained submodular maximization that takes a single pass through the universe using $O(\kappa/\epsilon^2)$ (κ is the cardinality constraint) memory. The resulting solution is a $\alpha/(1+\alpha) - \epsilon$-approximate solution where α is the approximation guarantee of the cardinality constrained submodular maximization algorithm used as a subroutine. This yields a 0.2779 approximation guarantee if using the state-of-the-art algorithm.

2 Algorithms and Theoretical Guarantees

In this Section, several bicriteria approximation algorithms are presented for SC, and their approximation guarantees proven. As mentioned in the introduction, the impossibility results of Feige *et al.* [2011] have implications for the approximation guarantees of SC. This is because an algorithm for SC can also be used as an algorithm for unconstrained submodular maximization by repeatedly guessing τ. These implications are stated in the following Theorem.

Theorem 1. *For any $\epsilon > 0$, there are instances of nonnegative symmetric submodular cover such that there is no (adaptive, possibly randomized) algorithm using fewer than $\Omega(\ln(1+\epsilon)e^{\epsilon^2 n}/\ln(n))$ queries that always finds a solution of expected f value at least $(1/2+\epsilon)\tau$.*

Proof. Suppose such an algorithm existed, and let it be called $\mathcal{A}$. Then a new algorithm for unconstrained submodular maximization is defined as follows: $\mathcal{A}$ is run on instance $SC(U, f, (1+\epsilon)^i)$ for every $i \in \mathbb{Z}$ such that $\max_{u\in U} f(\{u\}) \leq (1+\epsilon)^i \leq n \max_{u\in U} f(\{u\})$, and the solution with the highest value of f is returned. Notice this results in running $\mathcal{A}\ln(n)/\ln(1+\epsilon)$ times. Because OPT is in the above range, there exists some i such that $(1+\epsilon)^{i-1} \leq OPT \leq (1+\epsilon)^i$. Once $\mathcal{A}$ is run on $SC(U, f, (1+\epsilon)^i)$, by assumption it will return X such that $\mathbb{E}[f(X)] \geq (1/2+\epsilon)\tau$. This contradicts the result of Feige *et al.*. □

As a result of Theorem 1, is is not possible to develop an (α, β)-bicriteria approximation algorithm for SC such that $\beta > 1/2$. Therefore the algorithms presented in this section approximate the feasibility constraint as well as possible.

2.1 STREAM

The algorithms for SC presented in this section all depend on a subroutine called STREAM. In this section, STREAM is described and some theoretical properties of STREAM are proven.

Algorithm Description

STREAM takes as input a parameter $\epsilon \in (0, 1)$ and a guess of the optimal solution value, κ. STREAM then makes a single pass through the universe U in an arbitrary order, and stores a portion of the elements of total cost at most $(4/\epsilon^2 + 1)\kappa$. The stored elements are broken into $2/\epsilon$ disjoint sets, $S_1, ..., S_{2/\epsilon}$. The elements are chosen to be stored in at most one of the sets or not as they arrive, based upon their marginal gain to f with respect to the set, as well as their cost. Once the entire stream has been read, an unconstrained maximization algorithm, UNCONSTRAINEDMAX$_\gamma$, with a γ approximation ratio, is run on the union of the stored elements. Pseudocode for STREAM is presented in Algorithm 1.

Algorithm 1 STREAM

1: **procedure** STREAM(ϵ, κ)
2: $S_1 \leftarrow \emptyset, ..., S_{2/\epsilon} \leftarrow \emptyset$
3: **for** x received from stream **do**
4: **if** $w(x) \leq \kappa$ and $\exists j \in \{1, ..., 2/\epsilon\}$ s.t. $\Delta f(S_j, x)/w(x) \geq \epsilon\tau/(2\kappa)$ **then**
5: $S_j \leftarrow S_j \cup \{x\}$
6: **if** $w(S_j) > 2\kappa/\epsilon$ **then**
7: **break**
8: $S_0 \leftarrow$UNCONSTRAINEDMAX$_\gamma$ $\left(\cup_{j=1}^{2/\epsilon} S_j\right)$
9: **return** argmax$\{f(S_0), ..., f(S_{2/\epsilon})\}$

Theoretical Guarantees of STREAM

Two important theoretical results about STREAM are stated and proven in this section. Lemma 1 gives guarantees as far as the total weight of all stored elements of STREAM. Since the solution returned by STREAM is a subset of all stored elements, this also implies theoretical guarantees about the weight of the returned solutions. Lemma 2 gives needed theoretical guarantees about the f value of the solution returned by STREAM.

Lemma 1. *Let S be the set returned by* STREAM. *Then $w(S) \leq (4/\epsilon^2 + 1)\kappa$, and further throughout the duration of* STREAM *the total weight of all elements stored at once is at most $(4/\epsilon^2 + 1)\kappa$.*

Proof. Consider the state of STREAM at the beginning of an iteration of the for loop on Line 3, when an element x has been read in but not yet added to any of the sets $S_1, ..., S_{2/\epsilon}$. Then the if statement on Line 6 ensures that for all $i \in \{1, ..., 2/\epsilon\}$, $w(S_i) \leq 2\kappa/\epsilon$. At the end of the iteration, x has been added to at most a single set S_j, and the condition that $w(x) \leq \kappa$ on Line 4 to add x ensures that $w(S_j) \leq (2/\epsilon + 1)\kappa$. Further S_0 is a subset of $\cup_{i=1}^{2/\epsilon} S_i$. Therefore, at any point in STREAM before Line 8,

$$\begin{aligned} w\left(\cup S_{i=0}^{2/\epsilon}\right) &\leq \sum_{i=0}^{2/\epsilon} w(S_i) \\ &\leq (4/\epsilon^2 + 1)\kappa. \end{aligned}$$

Therefore the bound on the total weight at any point in STREAM holds. Because the solution returned by STREAM is a subset of $\cup S_{i=1}^{2/\epsilon}$, the bound on its weight is the same as the bound on its total memory. □

Lemma 2. *Suppose that* STREAM *is run with input $\epsilon \in (0, 1)$, and $\kappa \geq OPT$. Let S be the set returned by* STREAM. *Then $f(S) \geq \gamma(1 - \epsilon)\tau$.*

Proof. The loop on Line 3 of STREAM completes in one of two ways: (i) The **if** statement on Line 6 is satisfied; or (ii) All of the elements of U have been read from the stream, and Line 6 was never satisfied. The proof of Lemma 2 is broken up into each of these two events.

First suppose event (i) above occurs. Then at the completion of the loop there exists some $r \in \{1, ..., 2/\epsilon\}$ such that $w(S_r) \geq 2\kappa/\epsilon$. Let $S_r(\ell)$ be S_r after the ℓth element was added to it and $S_r(0) = \emptyset$. Then at the completion of STREAM

$$\begin{aligned} f(S_r) &\overset{(a)}{\geq} f(S_r) - f(\emptyset) \\ &= \sum_{\ell=1}^{|S_r|} (f(S_r(\ell)) - f(S_r(\ell-1))) \\ &\overset{(b)}{\geq} \sum_{x \in S_r} w(x)\epsilon\tau/(2\kappa) \\ &= w(S_r)\epsilon\tau/(2\kappa) \\ &\overset{(c)}{\geq} \tau \end{aligned}$$

where (a) is because $f(\emptyset) \geq 0$; (b) is by the condition on Line 7; and (c) is by the assumption that $w(S_r) \geq 2\kappa/\epsilon$. Therefore at the completion of STREAM $\max\{f(S_0), ..., f(S_{2/\epsilon})\} \geq f(S_r) \geq \tau$.

Now suppose that event (ii) above occurs. Then at the end of STREAM, $w(S_j) < 2\kappa/\epsilon$ for all $j \in \{1, ..., 2/\epsilon\}$. For this case, we need the following claim.

Claim 1. *Let $A_1, ..., A_m \subseteq U$ be disjoint, and $B \subseteq U$. Then there exists $i \in \{1, ..., m\}$ such that $f(A_i \cup B) \geq (1 - 1/m)f(B)$.*

Proof. Define $g(X) = f(B \cup X)$. Then g is a non-negative submodular function. Consider choosing A uniformly randomly from the disjoint sets $A_1, ..., A_m$. Then any element of U has probability at most $1/m$ of being in A. Then

$$\begin{aligned} \frac{1}{m}\sum_{i=1}^{m} f(B \cup A_i) &= \frac{1}{m}\sum_{i=1}^{m} g(A_i) \\ &= \mathbb{E}[g(A)] \\ &\overset{(a)}{\geq} \left(1 - \frac{1}{m}\right) g(\emptyset) \\ &= \left(1 - \frac{1}{m}\right) f(B) \end{aligned}$$

where (a) is from Lemma 4. Therefore there must exist some $i \in \{1, ..., m\}$ such that $f(A_i \cup B) \geq (1 - 1/m)f(B)$. □

Let S^* be an optimal solution to the instance of SC. By Claim 1, there exists $t \in \{1, ..., 2/\epsilon\}$ such that $(1 - \epsilon/2)\tau \leq f(S^* \cup S_t)$. Define $X_1 = S^* \cap (\cup_{i=1}^{2/\epsilon} S_i)$ and $X_2 = S^* \setminus X_1$. Then,

$$\begin{aligned} (1 - \epsilon/2)\tau &\leq f(S^* \cup S_t) \\ &= f(X_1 \cup S_t) + f(S^* \cup S_t) - f(X_1 \cup S_t) \\ &\overset{(a)}{\leq} f(X_1 \cup S_t) + \sum_{x \in X_2} \Delta f(X_1 \cup S_t, x) \\ &\overset{(b)}{\leq} f(X_1 \cup S_t) + \sum_{x \in X_2} \Delta f(S_t, x) \end{aligned} \tag{1}$$

where (a) and (b) are both due to submodularity. In addition,

$$\begin{aligned}\sum_{x\in X_2}\Delta f(S_t,x) &\overset{(a)}{\leq} \sum_{x\in X_2} w(x)\epsilon\tau/(2\kappa)\\ &= w(X_2)\epsilon\tau/(2\kappa)\\ &\overset{(b)}{\leq} w(X_2)\epsilon\tau/(2OPT)\\ &\overset{(c)}{\leq} \epsilon\tau/2 \end{aligned} \tag{2}$$

where (a) is by submodularity and the condition on Line 4; (b) is because $\kappa \geq OPT$; (c) is because $X_2 \subseteq S^*$ implies that $w(X_2) \leq OPT$. Then by combining Inequalities 1 and 2, we have that

$$\begin{aligned}(1-\epsilon)\tau &\leq f(X_1 \cup S_t)\\ &\overset{(a)}{\leq} \max_{Y\subseteq \cup_{i=1}^{2/\epsilon} S_i} f(Y)\\ &\overset{(b)}{\leq} \frac{1}{\gamma} f(S_0)\end{aligned}$$

where (a) is because $X_1 \cup S_t \subseteq \cup_{i=1}^{2/\epsilon} S_i$; (b) is because S_0 is an γ-approximate maximum of f over $\cup_{i=1}^{2/\epsilon} S_i$. Therefore $\max\{f(S_0), ..., f(S_{2/\epsilon})\} \geq f(S_0) \geq \gamma(1-\epsilon)\tau$. □

2.2 MULTI-PASS-COVER

In this section, the multiple pass streaming algorithm for SC, MULTI-PASS-COVER, is presented. MULTI-PASS-COVER takes $\mathcal{O}(\ln(OPT))$ passes through the universe U, stores elements of total weight at most $\mathcal{O}(OPT)$, and produces a constant bicriteria approximate solution with constraint approximation near to the optimal $1/2$.

Algorithm Description

MULTI-PASS-COVER takes as input a parameter $\epsilon \in (0,1)$. MULTI-PASS-COVER works be sequentially running STREAM for increasingly large guesses of OPT. First, the smallest possible guess of w_{min} is made for OPT. Each iteration, the guess is increased by a multiplicative factor of $1+\epsilon$. During each iteration, the solution S returned by STREAM is tested as to whether $f(S) \geq \gamma(1-\epsilon)\tau$, where γ is the approximation ratio of UNCONSTRAINEDMAX$_\gamma$. Once the guess is at least OPT, it is guaranteed that $f(S) \geq \gamma(1-\epsilon)\tau$ (as proven in Theorem 2), and then MULTI-PASS-COVER returns S and terminates. Pseudocode for MULTI-PASS-COVER is given in Algorithm 2.

Algorithm 2 MULTI-PASS-COVER

```
1: procedure MULTI-PASS-COVER(ε)
2:     κ ← w_min
3:     while true do
4:         S ← STREAM(ε, κ)
5:         if f(S) ≥ γ(1 − ε)τ then
6:             return S
7:         κ ← (1 + ε)κ
```

Theoretical Guarantees of MULTI-PASS-COVER

The theoretical guarantees of MULTI-PASS-COVER are now presented in Theorem 2.

Theorem 2. *Suppose that* MULTI-PASS-COVER *is run for an instance of SC. Then:*

(i) The returned set S satisfies $f(S) \geq \gamma(1-\epsilon)\tau$ and $w(S) \leq (1+\epsilon)(4/\epsilon^2+1)OPT$;
(ii) At most $\ln(OPT/w_{min})/\ln(1+\epsilon)$ passes through U are made;
(iii) The total cost of all elements needing to be stored at once is at most $(1+\epsilon)(4/\epsilon^2+1)OPT$;

Proof. Define $q \in \mathbb{Z}_{>0}$ to be the unique value where

$$(1+\epsilon)^{q-1}w_{min} < OPT \leq (1+\epsilon)^q w_{min}. \tag{3}$$

By Lemma 2, if the loop on Line 3 reaches $OPT = (1+\epsilon)^q w_{min}$, STREAM will return a set S that satisfies $f(S) \geq \gamma(1-\epsilon)\tau$. Then the if statement on Line 5 will be satisfied, and MULTI-PASS-COVER will terminate with solution S. Further, by Lemma 1, $w(S) \leq (4/\epsilon^2+1)(1+\epsilon)^q w_{min} \leq (4/\epsilon^2+1)(1+\epsilon)OPT$. Therefore item (i) is proven.

Each iteration of the loop on Line 3 corresponds to one pass through U. Since the loop on Line 3 stops before or once κ reaches $(1+\epsilon)^q w_{min}$ (as explained above), there are at most $\ln(OPT/w_{min})/\ln(1+\epsilon)$ passes through U. Therefore item (ii) is proven.

Over the course of MULTI-PASS-COVER, κ increases from w_{min} to $(1+\epsilon)^q w_{min}$ (as explained above). Further, each iteration of the for loop on Line 3 stores elements only needed in the corresponding call of STREAM. By Lemma 1, therefore the total weight of all elements stored at once is at most $(4/\epsilon^2+1)(1+\epsilon)^q w_{min} \leq (4/\epsilon^2+1)(1+\epsilon)OPT$. Therefore item (iii) is proven. □

2.3 SINGLE-PASS-COVER

While MULTI-PASS-COVER took $\mathcal{O}(ln(OPT))$ passes through the ground set U, an algorithm that makes a single pass through U while storing a low total cost is desirable for applications such as where the data is not stored at all.

Unfortunately, it is not possible to develop a single pass streaming algorithm for SC that returns an approximately feasible solution, while also maintaining low total stored cost relative to OPT. For example, suppose we have some single pass streaming algorithm for SC that produces a solution with constraint value at least $\alpha\tau$. Consider two instances of SC with uniform weight defined as follows: (i) SC $(\{u_1, ..., u_n\}, f_1, \tau)$ where f_1 is modular and $f(u_i) = \tau/n$ for all i; (ii) SC $(\{u_1, ..., u_n\}, f_2, \tau)$ where f_2 is modular and $f(u_i) = \tau/n$ for all $i \neq n$ and $f(u_n) = \tau$. Suppose the algorithm receives the universe in order $u_1, ..., u_n$. Then because the returned solution has constraint value at least αn, in instance (i) the algorithm must store at least $\alpha n - 1$ elements before reading element u_n. On the other hand, instances (i) and (ii) are indistinguishable up to element u_n, therefore for instance (ii) the algorithm also stores at least $\alpha n - 1$ elements. However, $OPT = 1$ in the latter case, and therefore this stored memory is very large compared to OPT.

In this section, a single pass algorithm is proposed, SINGLE-PASS-COVER, that instead maintains low memory relative to the optimal solution to an instance of SC where the universe is only those elements *read so far*. Once the entire universe is read in, SC produces a solution with the same bicriteria approximation ratio as MULTI-PASS-COVER.

Algorithm Description

SINGLE-PASS-COVER takes as input a parameter $\epsilon \in (0,1)$, and a parameter $B \in \mathbb{R}_{\geq 0}$. Like MULTI-PASS-COVER, SINGLE-PASS-COVER essentially works by running STREAM for guesses of OPT. However, SINGLE-PASS-COVER runs STREAM *in parallel*. Instead of guessing OPT sequentially, SINGLE-PASS-COVER maintains a set of guesses of OPT and updates a lower bound for the guesses lazily. On the other hand, an upper bound for OPT is initially given as input (B), and then updated by running UNCONSTRAINEDMAX$_\gamma$ for each guess after reading in each element. Pseudocode for SINGLE-PASS-COVER is given in Algorithm 3.

Algorithm 3 SINGLE-PASS-COVER

1: **procedure** SINGLE-PASS-COVER(ϵ, B)
2: $S_{(1+\epsilon)^i,j} \leftarrow \emptyset \; \forall i \in \mathbb{Z}, j \in \{0, ..., 2/\epsilon\}$
3: **for** x received from stream **do**
4: **if** $f(\{x\})/w(x) > m$ **then**
5: $m \leftarrow f(\{x\})/w(x)$
6: **for** κ in $\{(1+\epsilon)^i : i \in \mathbb{Z}, \epsilon\tau/(2m) \leq (1+\epsilon)^i \leq B\}$ **do**
7: **if** $\exists i \in \{1, ..., 2/\epsilon\}$ s.t. $|S_{\kappa,i}| < 2\kappa/\epsilon$ and $\Delta f(S_{\kappa,i}, x) \geq w(x)\epsilon\tau/(2\kappa)$ **then**
8: $S_{\kappa,i} \leftarrow S_{\kappa,i} \cup \{x\}$
9: $S_{\kappa,0} \leftarrow$UNCONSTRAINEDMAX$_\gamma$ $(\cup_{i=1}^{2/\epsilon} S_{\kappa,i})$
10: **if** $\max\{f(S_{\kappa,i}) : i \in \{0, ..., 2/\epsilon\}\} \geq \gamma(1-\epsilon)\tau$ **then**
11: $B \leftarrow \kappa$
12: $S_{(1+\epsilon)^i,j} \leftarrow \emptyset \; \forall i \in \mathbb{Z}, (1+\epsilon)^i > B, j \in \{0, ..., 2/\epsilon\}$
13: **return** argmax$\{f(S_{\kappa,i}) : \kappa \leq B, i \in \{0, ..., 2/\epsilon\}\}$

Theoretical Guarantees of SINGLE-PASS-COVER

The theoretical guarantees of SINGLE-PASS-COVER are presented in Theorem 3.

Theorem 3. *Define $\xi = \min_{u \in U} w(u)/f(\{u\})$. Then if $B \geq OPT$:*

(i) The set S returned by SINGLE-PASS-COVER *satisfies $f(S) \geq \gamma(1-\epsilon)\tau$ and $w(S) \leq (1+\epsilon)(4/\epsilon^2 + 1)OPT$;*
(ii) SINGLE-PASS-COVER *makes a single pass through U;*
(iii) The total weight of all elements stored at one time is at most $(1+\epsilon)(4/\epsilon^2+1)\ln(2B/(\epsilon\tau\xi))/\ln(1+\epsilon)B$;

Let $u_1, ..., u_n$ be the order that the elements of U arrive in, $U_i = \{u_1, ..., u_i\}$, $SC(U_i, f, \tau, OPT_i)$ to be the instance of SC corresponding to ground set U_i, and $r = \min\{i \in \{1, ..., n\} : SC(U_i, f, \tau, OPT_i)$ has a feasible solution$\}$. Then if $B \geq OPT_r$:

(v) Once the iteration of the loop corresponding to element u_i, $i \geq r$, is complete, the total weight of all elements stored at one time at most $(1+\epsilon)(4/\epsilon^2+1)\ln(2OPT_i/(\epsilon\tau\xi))/\ln(1+\epsilon)OPT_i$ from that point on.

Proof. Consider an alternate version of STREAM where instead of running UNCONSTRAINEDMAX$_\gamma$ on $\cup S_i$ after receiving all elements in the stream (Line 8), UNCONSTRAINEDMAX$_\gamma$ is run at the end of each iteration of the loop on Line 3 of STREAM. I.e., UNCONSTRAINEDMAX$_\gamma$ is run after reading in each element of U. Notice that this does not change any of the properties of STREAM detailed in Lemmas 1 and 2. Then, one can imagine SINGLE-PASS-COVER as running many different instances of STREAM in parallel as U is read in. In particular, the set $\{(1+\epsilon)^i : i \in \mathbb{Z}, \epsilon\tau/(2m) \leq (1+\epsilon)^i \leq B\}$ are the guesses of OPT, and there is an instance of STREAM corresponding to each guess. For each guess κ, $S_{\kappa,0}, ..., S_{\kappa,2/\epsilon}$ in SINGLE-PASS-COVER correspond to the sets $S_0, ..., S_{2/\epsilon}$ in STREAM.

Consider the value of B at the end of some iteration of the for loop on Line 3. It is now shown that without loss of generality, one can assume that up to this point SINGLE-PASS-COVER is equivalent to running STREAM in parallel with guesses of OPT $\{(1+\epsilon)^i : i \in \mathbb{Z}, \epsilon\tau/(2m) \leq (1+\epsilon)^i \leq B\}$ up to this point in the algorithm. B is only decreasing throughout STREAM, therefore we need to show that small guesses of OPT are wlog running in parallel. Consider any $(1+\epsilon)^i \leq B$. Consider any previous iteration of the loop on Line 3 such that for the first time an x has arrived such that $\Delta f(\emptyset, x) \geq w(x)\epsilon\tau/(2(1+\epsilon)^i)$ (i.e. the first time an element should be added to $S_{(1+\epsilon)^i,j}$ for some $j \in \{1, ..., 2/\epsilon\}$), and we are at the beginning of the loop on Line 3. If

$\epsilon\tau/(2m) > (1+\epsilon)^i$, then

$$\begin{aligned} f(\{x\})/w(x) &\geq \Delta f(\emptyset, x)/w(x) \\ &\geq \epsilon\tau/(2(1+\epsilon)^i) \\ &> m. \end{aligned}$$

Therefore the if statement on Line 4 will be true, m will be reset to $f(\{x\})/w(x)$, and $(1+\epsilon)^i$ added to the guesses of OPT since

$$\begin{aligned} (1+\epsilon)^i &\geq w(x)\epsilon\tau/(2\Delta f(\emptyset, x)) \\ &\geq \epsilon\tau/(2m). \end{aligned}$$

Item (i) is now proven. By Lemma 2, if there exists a run of STREAM with a guess of OPT that is at least as big, then the set returned by STREAM has f value at least $\gamma(1-\epsilon)\tau$. Therefore by the end of SINGLE-PASS-COVER, any run of STREAM corresponding to a guess of OPT that is at least as big as OPT must have triggered the if statement on Line 10. Initially $B \geq OPT$, and only decreases if the if statement on Line 10 is true, it must be that the solution S of SINGLE-PASS-COVER has $f(S) \geq \gamma(1-\epsilon)\tau$. In addition, the above discussion implies that B is no greater than $(1+\epsilon)OPT$ at the end of SINGLE-PASS-COVER, then Lemma 1 implies the remaining part of item (i).

Item (iii) is now proven. By Lemma 1, the total weight of all elements stored by each run of STREAM with input (ϵ, κ) is $(4/\epsilon^2+1)\kappa$, which is bounded above by $(4/\epsilon^2+1)B$. In addition, $\epsilon\tau/(2m) \geq \xi$, and therefore there are at most $\ln(B/\xi)/\ln(1+\epsilon)$ parallel instances of STREAM running in SINGLE-PASS-COVER. This proves item (iii).

Finally, item (v) is proven. Suppose the iteration of the for loop on Line 3 corresponding to element u_i is complete. By a nearly identical argument to that used for item (i), one can see that the largest guess of OPT is no bigger than $(1+\epsilon)OPT_i$ at this point. Therefore the largest memory for any run of STREAM is $(1+\epsilon)(4/\epsilon^2+1)OPT_i$ by Lemma 1. As shown when proving item (ii), there are at most $\ln(B/\xi)/\ln(1+\epsilon)$ parallel instances of STREAM running in SINGLE-PASS-COVER. Altogether this implies item (v).

□

2.4 Submodular Maximization with a Knapsack Constraint

A related optimization problem to SC is Submodular Maximization with a Knapsack Constraint, defined as follows.

Definition 2 (Submodular Knapsack (SK))**.** *Let $f : 2^U \to \mathbb{R}_{\geq 0}$ be a submodular function defined over subsets of the universe U of size n such that $f(\emptyset) = 0$, and let $w : U \to \mathbb{R}_{\geq 0}$ be a cost function. The Submodular Knapsack problem (SK) is, given $\kappa \in \mathbb{R}_{\geq 0}$, find*

$$\textit{argmax}_{X \subseteq U}\{f(X) : \sum_{x \in X} w(x) \leq \kappa\}.$$

Bicriteria approximation algorithms similar in spirit to those presented previously for SC can also be used for SK. These algorithms as well as their theoretical guarantees are presented in the current section.

Algorithm Description

SINGLE-PASS-KNAPSACK is most related to SINGLE-PASS-COVER for SC, in that it runs STREAM in parallel for many guesses of OPT. However, it is simpler to guess OPT for this problem. SINGLE-PASS-KNAPSACK keeps track of a set of guesses of OPT, T, and updates them lazily (Line 6 of Algorithm 4). There is no need to run UNCONSTRAINEDMAX$_\gamma$ repeatedly after every element is read in, UNCONSTRAINEDMAX$_\gamma$ is run for each guess of OPT once all elements have read in. Pseudocode for SINGLE-PASS-KNAPSACK can be found in Algorithm 4.

Algorithm 4 SINGLE-PASS-KNAPSACK

procedure SINGLE-PASS-KNAPSACK(ϵ)
  $S_{(1+\epsilon)^i,j} \leftarrow \emptyset \; \forall i \in \mathbb{Z}, j \in \{0, ..., 2/\epsilon\}$
  for x received from stream **do**
    if $f(\{x\})/w(x) > m$ **then**
      $m \leftarrow f(\{x\})/w(x)$
      $S_{(1+\epsilon)^i,j} \leftarrow \emptyset \; \forall i \in \mathbb{Z}, (1+\epsilon)^i < f(\{x\}), j \in \{0, ..., 2/\epsilon\}$
    for τ in $\{(1+\epsilon)^i : i \in \mathbb{Z}, f(\{x\}) \leq (1+\epsilon)^i \leq 2m\kappa/\epsilon\}$ **do**
      if $\exists i \in \{1, ..., 2/\epsilon\}$ s.t. $|S_{\kappa,i}| < 2\kappa/\epsilon$ and $\Delta f(S_{\kappa,i}, x) \geq w(x)\epsilon\tau/(2\kappa)$ **then**
        $S_{\kappa,i} \leftarrow S_{\kappa,i} \cup \{x\}$
  for τ in T **do**
    $S_{\tau,0} \leftarrow$UNCONSTRAINEDMAX$_\gamma$ $(\cup_{i=1}^{2/\epsilon} S_{\tau,i})$
  return argmax$\{f(S_{\tau,i}) : \tau \in T, i \in \{0, ..., 2/\epsilon\}\}$

Theoretical Guarantess of SINGLE-PASS-KNAPSACK

Theorem 4. *Suppose that* SINGLE-PASS-KNAPSACK *is run for SK*(U, f, w, κ, OPT) *with input* $\epsilon \in (0, 1)$. *Then:*

(i) The set S returned by SINGLE-PASS-KNAPSACK *satisfies $f(S) \geq \gamma(1-\epsilon)OPT$ and $w(S) \leq (1+\epsilon)(4/\epsilon^2+1)\kappa$;*
(ii) At most $(4/\epsilon^2 + 2/\epsilon)\ln(2\kappa/(w_{min}\epsilon))/\ln(1+\epsilon)\kappa$ elements of U are stored all at once;

Proof. In order to prove Theorem 4, a new version of Lemma 2 is needed. The following Lemma is proved in as essentially identical way to Lemma 2:

Lemma 3. *Suppose that* STREAM *is run with input $\epsilon \in (0, 1)$, and $\tau \geq OPT$. Let S be the set returned by* STREAM. *Then $f(S) \geq \gamma(1-\epsilon)OPT$.*

Similar to SINGLE-PASS-COVER, SINGLE-PASS-KNAPSACK is essentially running a bunch of instances of STREAM in parallel as U is read in. In particular, the set $\{(1+\epsilon)^i : i \in \mathbb{Z}, f(\{x\} \leq (1+\epsilon)^i \leq 2m\kappa/\epsilon\}$ are the guesses of OPT, and there is an instance of STREAM corresponding to each guess. For each guess τ, $S_{\tau,0}, ..., S_{\tau,2/\epsilon}$ in SINGLE-PASS-COVER correspond to the sets $S_0, ..., S_{2/\epsilon}$ in STREAM.

Define $q \in \mathbb{Z}$ to be the unique value such that

$$(1+\epsilon)^q \leq OPT < (1+\epsilon)^{q+1}.$$

Then we may assume without loss of generality that there is an instance of STREAM corresponding to $(1+\epsilon)^q$ as a guess of OPT for the duration of SINGLE-PASS-KNAPSACK, as explained as follows. First of all, clearly $(1+\epsilon)^q \geq \max\{f(\{x\}) : x \in U\}$ and therefore is at least the smallest guess throughout the duration of SINGLE-PASS-KNAPSACK. On the other hand, suppose that for the first time we have received from the stream an element x such that $\Delta f(\emptyset, x) \geq \epsilon w(x)(1+\epsilon)^q/(2\kappa)$ (i.e. the first time an element x should be added to $S_{(1+\epsilon)^q,i}$ for some $i \in \{1, ..., 2/\epsilon\}$). If $(1+\epsilon)^q > 2m\kappa/\epsilon$ at the beginning of the for loop then

$$\begin{aligned} f(\{x\})/w(x) &\overset{(a)}{\geq} \Delta f(\emptyset, x)/w(x) \\ &\geq \epsilon(1+\epsilon)^q/(2\kappa) \\ &> m \end{aligned}$$

where (a) is because $f(\emptyset) \geq 0$. Therefore the if statement on Line **??** will be true, m will be re-assigned as

$f(\{x\})/w(x)$, and $(1+\epsilon)^q$ added to the guess of OPT since

$$\begin{aligned}(1+\epsilon)^q &\leq 2\Delta f(\emptyset, x)\kappa/(w(x)\epsilon)\\ &\leq 2f(\{x\})\kappa/(w(x)\epsilon)\\ &= 2m\kappa/\epsilon\end{aligned}$$

and will remain in the guesses until the end.

In light of the above, items (i) and (ii) follow by Lemmas 1 and 3. □

3 Appendix

Lemma 4. *(Lemma 2.2 from Feige* et al. *[2011]) Let $g : 2^U \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Denote by $A(p)$ a random subset of A where each element appears with probability at most p (not necessarily independently). Then $\mathbb{E}[g(A(p))] \geq (1-p)g(\emptyset)$.*

References

Naor Alaluf, Alina Ene, Moran Feldman, Huy L Nguyen, and Andrew Suh. Optimal streaming algorithms for submodular maximization with cardinality constraints. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and Data Mining (KDD)*, pages 671–680, 2014.

Elham J Barezi, Ian D Wood, Pascale Fung, and Hamid R Rabiee. A submodular feature-aware framework for label subset selection in extreme classification problems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1009–1018, 2019.

Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Transactions on Algorithms (TALG)*, 14(3):1–20, 2018.

Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Mathematics of Operations Research*, 44(3):988–1005, 2019.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. SIAM, 2014.

Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query trade-off in submodular maximization. *Mathematics of Operations Research*, 42(2):308–329, 2017.

Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1):225–247, 2015.

Lin Chen, Moran Feldman, and Amin Karbasi. Unconstrained submodular maximization with constant adaptive complexity. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 102–113, 2019.

Victoria Crawford, Alan Kuhnle, and My Thai. Submodular cost submodular cover with an approximate oracle. In *International Conference on Machine Learning*, pages 1426–1435. PMLR, 2019.

Abhimanyu Das and David Kempe. Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection. *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011.

Shahar Dobzinski and Ami Mor. A deterministic algorithm for maximizing submodular functions. *arXiv*, pages arXiv–1507, 2015.

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

Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1098–1116. SIAM, 2011.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics*, pages 246–257. Springer, 2010.

Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 323–332. ACM, 2009.

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*, pages 420–429, 2007.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

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.

Ashkan Norouzi-Fard, Abbas Bazzi, Marwa El Halabi, Ilija Bogunovic, Ya-Ping Hsieh, and Volkan Cevher. An efficient streaming algorithm for the submodular cover problem. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4500–4508, 2016.

Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421, 2014.

Peng Jun Wan, Ding Zhu Du, Panos Pardalos, and Weili Wu. Greedy approximations for minimum submodular cover with submodular cost. *Computational Optimization and Applications*, 45(2):463–474, 2010.

Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

Jia Xu, Lopamudra Mukherjee, Yin Li, Jamieson Warner, James M Rehg, and Vikas Singh. Gaze-enabled egocentric video summarization via constrained submodular maximization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2235–2244, 2015.