Streaming Submodular Maximization under a k-Set System Constraint

Ran Haba [1] **Ehsan Kazemi** [2 3] **Moran Feldman** [4] **Amin Karbasi** [2]

Abstract

In this paper, we propose a novel framework that converts streaming algorithms for monotone submodular maximization into streaming algorithms for non-monotone submodular maximization. This reduction readily leads to the currently tightest deterministic approximation ratio for submodular maximization subject to a k-matchoid constraint. Moreover, we propose the first streaming algorithm for monotone submodular maximization subject to k-extendible and k-set system constraints. Together with our proposed reduction, we obtain $O(k \log k)$ and $O(k^2 \log k)$ approximation ratio for submodular maximization subject to the above constraints, respectively. We extensively evaluate the empirical performance of our algorithm against the existing work in a series of experiments including finding the maximum independent set in randomly generated graphs, maximizing linear functions over social networks, movie recommendation, Yelp location summarization, and Twitter data summarization.

1. Introduction

Submodularity captures an intuitive diminishing returns property where the benefit of an item decreases as the context in which it is considered grows. This property naturally occurs in many applications where items may represent data points, features, actions, etc. Moreover, submodularity is a sufficient condition that leads to an efficient optimization procedure for many discrete optimization problems. The above reasons have led to a surge of applications in machine learning where the gain of discrete choices shows diminishing returns and the optimization can be handled efficiently. Novel examples include non-parametric learning (Mirzasoleiman et al., 2016b), dictionary learning (Das & Kempe, 2011), crowd teaching (Singla et al., 2014), regression under human assistance (De et al., 2019), sequence selection (Tschiatschek et al., 2017; Mitrovic et al., 2019) interpreting neural networks (Elenberg et al., 2017), adversarial attacks (Lei et al., 2019), fairness (Kazemi et al., 2018), social graph analysis (Norouzi-Fard et al., 2018), data summarization (Dasgupta et al., 2013; Tschiatschek et al., 2014; Elhamifar & Clara De Paolis Kaluza, 2017; Kirchhoff & Bilmes, 2014; Mitrovic et al., 2018; Kazemi et al., 2020), fMRI parcellation (Salehi et al., 2017), and DNA sequencing (Libbrecht et al., 2018).

More formally, a set function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ is called **submodular** if for all sets $A \subseteq B \subseteq \mathcal{N}$ and element $u \notin B$ we have $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$. Moreover, a set function is called monotone if $f(A) \leq f(B)$ whenever $A \subseteq B$. The focus of this paper is on maximizing a general submodular function (not necessarily monotone). More concretely, we consider a very general form of constrained submodular maximization, i.e.,

$$\texttt{OPT} = \arg\max_{A \in \mathcal{I}} f(A) \ , \tag{1}$$

where $\mathcal{I}$ represents the set of feasible solutions. For instance, when f is monotone and $\mathcal{I}$ represents a cardinality/size constraint,[1] the celebrated result of (Nemhauser et al., 1978) states that the greedy algorithm achieves a $e/(e-1)$-approximation for this problem, which is known to be optimal (Nemhauser & Wolsey, 1978). In the recent years, there has been a large body of literature aiming at solving Problem (1) in the offline/centralized setting under various types of feasibility constraints such as matroid, k-matchoid, k-extendible system, and k-set system (formal definitions of some of these terms appear in Section 3).

In the offline/centralized setting, the problem of maximizing a (non-monotone) submodular function subject to the above types of constraints is fairly well understood and easy-to-implement algorithms have been proposed. For instance, for maximization under a k-set system constraint, one obtains an approximation ratio of $k + O(\sqrt{k})$ using roughly $\sqrt{k}$ invocations of the natural greedy algorithm and an algorithm for unconstrained submodular maximization. Or, when

[1]Depart. of Mathematics and Computer Science, The Open University of Israel [2]Yale Institute for Network Science, Yale University [3]Now at Google, Zürich, Switzerland [4]Department of Computer Science, University of Haifa, Israel. Correspondence to: Ehsan Kazemi <ehsankazemi@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]Formally, $\mathcal{I}$ contains in this case all subsets of $\mathcal{N}$ of size at most ρ for some value ρ.

the constraint is a k-extendible system, running the greedy algorithm only once over a carefully subsampled ground set achieves a $k + 3$ approximation ratio (Feldman et al., 2017). It should also be noted that as is the greedy algorithm fails to provide any constant factor approximation guarantee when the submodular function is non-monotone, and thus, the above modifications of it are necessary.

In the streaming setting, when the elements arrive one at a time and the memory footprint is not allowed to grow significantly with the size of the data, the landscape of constrained submodular maximization is much less understood. In particular, even for the simple problem of monotone submodular maximization subject to a cardinality constraint, the best known approximation guarantee is 2 (Badanidiyuru et al., 2014) (as opposed to $e/(e-1)$ in the offline setting). Moreover, no algorithm is currently known to achieve a non-trivial guarantee for more complicated constraints such as k-extendible or k-set system in the streaming setting even when the submodular objective function is monotone.

In this paper, we propose the first streaming algorithm for maximizing a general submodular function (not necessarily monotone) subject to a general k-set system constraint. Our algorithm achieves an $O(k^2 \log k)$ approximation ratio for this problem. Moreover, when the constraint reduces to a k-extendible system the approximation guarantee of our streaming method improves to a better $O(k \log k)$ approximation ratio, nearly matching a lower bound of roughly k due to (Feldman et al., 2017) that applies even for the offline version of the problem. Interestingly, the last approximation ratio is a significant improvement even compared to the best approximation ratio previously known for the very special case of this problem in which the objective function is linear. The current state-of-the-art algorithm for this special case, due to (Crouch & Stubbs, 2014), guarantees only an $O(k^2)$-approximation.

With the exception some algorithms designed for the simple cardinality constraint (Alaluf & Feldman, 2019; Badanidiyuru et al., 2014; Ene et al., 2019; Kazemi et al., 2019), all the streaming algorithms previously suggested for submodular maximization (see (Buchbinder et al., 2014; Chekuri et al., 2015; Chakrabarti & Kale, 2015; Feldman et al., 2018)) have been based on the same basic technique. These algorithms maintain a feasible solution, and update it in the following way. When an element u arrives, the algorithm (1) determines a set of elements that have to be removed from the current feasible solution to allow u to be added without violating feasibility, and then (2) decides using some algorithm specific rule whether it is beneficial to make this trade (i.e., add u and remove the necessary elements to recover feasibility). Our algorithm uses a very different technique of maintaining multiple feasible solutions to which elements can be added (but can never be removed), which is inspired by the technique of (Crouch & Stubbs, 2014) for maximization of linear functions subject to k-set systems. Intuitively, each one of the solutions maintained by our algorithm is associated with a particular importance of elements, and the role of this solution is to collect enough elements of this importance. Since we collect elements from each level of importance, once the stream ends, the union of the solutions we maintain is a good enough summary of the stream, and our algorithm is able to pick a feasible subset of this union which is competitive with respect to the optimal solution.

One component of our algorithm is a general framework that is able to convert many streaming algorithms for monotone submodular maximization to similar algorithms for non-monotone submodular maximization. As an immediate consequence of this framework, we get a deterministic streaming algorithm for maximizing a general (not necessarily monotone) submodular function subject to a k-matchoid constraint, which is an improvement over the state-of-the-art deterministic approximation ratio for this problem due to (Chekuri et al., 2015). We also compare the empirical performance of our algorithm with the existing work and natural baselines in a set of experiments including independent set over randomly generated graphs, maximizing a linear function over edges of a graph, movie recommendation, and Yelp location data summarization. In all these applications, the various constraints are modeled as an instance of a k-extendible or a k-set system.

Before concluding this section, we need to highlight a technical issue. The standard definition of streaming algorithms requires them to use poly-logarithmic amount of space, which is less than the space necessary for keeping a solution to our problem. Thus, no algorithm for this problem aiming to produce a solution (rather than just estimate the value of the optimal solution) can be a true streaming algorithm. This is true also for all the above mentioned streaming algorithms, which are in fact semi-streaming algorithms—a semi-streaming algorithm is an algorithm that processes the data as a sequence of elements using an amount of space which is nearly linear in the maximum size of a feasible solution and typically makes only a single pass over the entire data stream. Since true streaming algorithms are almost irrelevant to our setting, we ignore the distinction between streaming and semi-streaming algorithms in this paper and often use the term "streaming algorithm" to refer to a semi-streaming algorithm.

Paper Structure. In Section 2, we review the related work. In Section 3, we formally define different types of constraints and the notation we use, and then formally state the technical results that we need. In Section 4, we describe our above mentioned framework for converting streaming algorithms for monotone submodular maximization into streaming algorithms for non-monotone submodular max-

imization. Then, in Section 5, we describe and formally analyze our algorithm, and in Section 6 we describe the experiments we conducted to study the empirical performance of this algorithm. **Most of the proofs for the theoretical results are deferred to the Supplementary Material.** An earlier version of this paper, in which the result applies only to non-negative linear functions subject to k-extendible constraints (Feldman & Haba, 2019), appeared on arXiv at the past under a different title.

2. Related Work

The study of submodular maximization in the streaming setting was initialized by the works of Badanidiyuru et al. (2014) and Chakrabarti & Kale (2015). As discussed above, the work of (Chakrabarti & Kale, 2015) was based on a technique allowing the removal of elements from the solution (also known as preemption). Originally, (Chakrabarti & Kale, 2015) suggested this technique only for constraints formed by the intersection of k-matroids and a monotone submodular objective function, but later works extended the use of the technique to the more general class of k-matchoid constraints as well as non-monotone submodular functions (Buchbinder et al., 2014; Chekuri et al., 2015; Feldman et al., 2018).

The above mentioned algorithm of Badanidiyuru et al. (2014) works only for the simple cardinality constraint and monotone submodular objective functions, but provides an improved approximation ratio of 2 for this setting (recently, Feldman et al. (2020) proved the optimality of this approximation factor). The technique at the heart of this algorithm is based on growing a set to which elements can only be added, which becomes the output solution of the algorithm by the end of the stream (unlike the case in the technique of (Crouch & Stubbs, 2014) on which we base our results, in which the final solution is obtained by combining multiple sets grown by the algorithm). More recent works improved the algorithm of (Badanidiyuru et al., 2014) by improving its space complexity (Kazemi et al., 2019) and extending its technique to non-monotone submodular functions (Alaluf & Feldman, 2019; Ene et al., 2019).

The study of submodular maximization in the offline (centralized) setting is very vast, and thus, we concentrate here only on results for general k-extendible or k-set system constraints. Already in 1978, Fisher et al. (1978) proved that the natural greedy algorithm obtains $k+1$ approximation for the problem of maximizing a monotone submodular function subject to a k-set system constraint (some of their proof was given implicitly, and the details were filled in by (Călinescu et al., 2011)). This was recently proved to be almost optimal. Specifically, Badanidiyuru & Vondrák (2014) proved that no polynomial time algorithm can obtain $k-\varepsilon$ approximation for this problem for any constant $\varepsilon > 0$, and the same inapproximability result was later shown to apply also to k-extendible constraints by (Feldman et al., 2017). As mentioned in Section 1, Feldman et al. (2017) presented the state-of-the-art algorithms for maximizing a (not necessarily monotone) submodular function subject to k-set system and k-extendible constraints. Both algorithms obtain $k + o(k)$ approximation, which improves over two previous results due to (Gupta et al., 2010) and (Mirzasoleiman et al., 2016a) that obtained roughly $3k$ and $2k$ approximation, respectively, for the more general case of a k-set system constraint.

This hierarchy of independence systems (including matroids, k-matchoids, k-extendible systems, and k-set system) is quite rich and expressive. Several recent works have used these general constraints for modeling real-world applications. Badanidiyuru et al. (2020) cast text, location, and video data summarization tasks to the problem of maximizing a submodular function subject to the interaction of k-matroids and extend their results to k-matchoids. Mirzasoleiman et al. (2018) and Feldman et al. (2018) studied the video and location data summarization applications subject to k-matchoid constraints in the streaming setting. Feldman et al. (2017) used a k-extendible constraint to model a movie recommendation application. Mirzasoleiman et al. (2016a) used the intersection of k matroids and ℓ knapsacks to model several machine learning applications, including recommendation systems, image summarization, and revenue maximization tasks.

3. Preliminaries and Notation

We begin this section by presenting some notation that we use in this paper. Then, we formally define some types of constraints mentioned in Section 1, and discuss the guarantee of a simple greedy algorithm for these constraints.

Given an element u and a set A, we use $A+u$ as a shorthand for the union $A \cup \{u\}$. We also denote the marginal gain of adding u to A with respect to a set function $f\colon 2^{\mathcal{N}} \to \mathbb{R}$ using $f(u \mid A) \triangleq f(A+u) - f(A)$. Similarly, the marginal gain of adding a set $B \subseteq \mathcal{N}$ to another set $A \subseteq \mathcal{N}$ is denoted by $f(B \mid A) \triangleq f(B \cup A) - f(A)$. Note that this notation allows us, for example, to rewrite the definition of submodularity as the requirement that $f(u \mid A) \geq f(u \mid B)$ for every two sets $A \subseteq B \subseteq \mathcal{N}$ and element $u \notin B$.

A constraint is defined, for our purposes, as a pair $(\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is a ground set and $\mathcal{I}$ is the collection of all feasible subsets of $\mathcal{N}$. All the types of constraints discussed in Section 1 are independence systems according to the following definition.

Definition 1. *Given a ground set $\mathcal{N}$ and a collection of sets $\mathcal{I} \subseteq 2^{\mathcal{N}}$, the pair $(\mathcal{N}, \mathcal{I})$ is an independence system if (i) $\varnothing \in \mathcal{I}$ and (ii) for $B \in \mathcal{I}$ and any $A \subseteq B$ we have $A \in \mathcal{I}$.*

It is customary to call a set $A \subseteq \mathcal{N}$ *independent* if it belongs to $\mathcal{I}$ and *dependent* if it does not (i.e., it is infeasible). An independent set $B \in \mathcal{I}$ which is maximal with respect to inclusion is called a *base*; that is, $B \in \mathcal{I}$ is a base if $A \in \mathcal{I}$ and $B \subseteq A$ imply that $B = A$. Furthermore, an independent set $B \in \mathcal{I}$ which is a subset of some set $E \subseteq \mathcal{N}$ is called a base of E if it is a base of the independence system $(E, 2^E \cap \mathcal{I})$. Note that this means that a set B is a base of $(\mathcal{N}, \mathcal{I})$ if and only if it is a base of $\mathcal{N}$.

With this terminology, we can now define k-set systems.

Definition 2. *An independence system $(\mathcal{N}, \mathcal{I})$ is a k-set system for an integer $k \geq 1$ if for every set $E \subseteq \mathcal{N}$, all the bases of E have the same size up to a factor of k (in other words, the ratio between the sizes of the largest and smallest bases of E is at most k).*

An immediate consequence of the definition of k-set systems is that any base of such a system is a maximum size independent set up to an approximation ratio of k. Thus, one can get a k-approximation for the problem of finding a maximum size set subject to a k-set system constraint by outputting an arbitrary base of the k-set system, which can be done using the following simple strategy. Start with the empty solution, and consider the elements of the ground set $\mathcal{N}$ in an arbitrary order. When considering an element, add it to the current solution, unless this will make the solution dependent. We refer to this procedure as the unweighted greedy algorithm.

Let us now define k-extendible systems. We remind the reader that k-extendible systems are well-known to be a restricted class of k-set systems.

Definition 3. *An independence system $(\mathcal{N}, \mathcal{I})$ is a k-extendible system for an integer $k \geq 1$ if for any two independent sets $S \subseteq T \subseteq \mathcal{N}$, and an element $u \notin T$ such that $S + u \in \mathcal{I}$, there is a subset $Y \subseteq T \setminus S$ of size at most k such that $T \setminus Y + u \in \mathcal{I}$.*

Since k-extendible systems are, in particular, k-set systems, the above discussion already implies that the unweighted greedy algorithm obtains k-approximation for the problem of finding a maximum size independent set in such a system. The following lemma strengthens this observation, and is the key technical reason that our algorithm has a better approximation guarantee for k-extendible system constraints than for k-set system constraints. The proof of the lemma can be found in Appendix A.1.

Lemma 4. *Given a k-extendible set system $(\mathcal{N}, \mathcal{I})$, the unweighted greedy algorithm is guaranteed to produce an independent set B such that $k \cdot |B \setminus A| \geq |A \setminus B|$ for any independent set $A \in \mathcal{I}$.*

4. A Framework: From Monotone to Non-Monotone Streaming Maximization

Mirzasoleiman et al. (2018) proposed a framework for the following task. Given a streaming[2] algorithm for maximizing monotone submodular functions, the framework produces a similar algorithm that works also for non-monotone submodular objectives. Unfortunately, however, this framework applies only to algorithms satisfying a property which, to the best of our knowledge, is not satisfied by any streaming algorithm from the literature (except algorithms that work for non-monotone functions by design). In particular, this is the case for the algorithm of Chekuri et al. (2015) explicitly mentioned by (Mirzasoleiman et al., 2018) as a natural fit for their framework. In the rest of this section we discuss this issue in more detail, and then introduce a different framework which achieves the same goal (converting algorithms for monotone submodular maximization into algorithms for non-monotone submodular maximization), but requires a different property from the input algorithms which is satisfied by both existing algorithms from the literature and the new algorithm we suggest in this paper.

The algorithm of (Chekuri et al., 2015) discussed above is a streaming algorithm for maximizing monotone submodular functions under a k-matchoid constraint, and Mirzasoleiman et al. (2018) applied their framework to it in order to get such an algorithm for non-monotone function. Formally, this framework requires the input streaming algorithm to satisfy the inequality

$$f(S) \geq \alpha \cdot f(S \cup T) \ , \tag{2}$$

where S as the output of the algorithm, T is an arbitrary feasible solution and α is a positive value. Unfortunately, the algorithm of (Chekuri et al., 2015) fails to satisfy Eq. (2) for any constant α, so does the algorithms of Buchbinder et al. (2019) and Chakrabarti & Kale (2015). In Appendix B we provide examples showing that this is the case for all these algorithms even under a simple cardinality constraint.

Interestingly, Chekuri et al. (2015) presented, prior to the work of (Mirzasoleiman et al., 2018), an alternative method to convert their algorithm into a deterministic algorithm for non-monotone functions based on a technique due to Gupta et al. (2010). The framework we suggest in this paper, can be viewed as a formalization and generalization of this technique. As an alternative to the property (2), which does not have counterparts in most of the existing streaming algorithms, our framework uses the property described by Definition 5.

[2]Recall that in this paper we use the term "streaming algorithm" to refer to algorithms that are technically "semi-streaming algorithms", i.e., their space complexity is allowed to be nearly-linear in the size of the output set.

Definition 5. *Consider a data stream algorithm*[3] *for maximizing a non-negative submodular function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ subject to a constraint $(\mathcal{N}, \mathcal{I})$. We say that such an algorithm is an (α, γ)-approximation algorithm, for some $\alpha \geq 1$ and $\gamma \geq 0$, if it returns two sets $S \subseteq A \subseteq \mathcal{N}$ such that $S \in \mathcal{I}$, and for all $T \in \mathcal{I}$ we have*

$$\mathbb{E}[f(T \cup A)] \leq \alpha \cdot \mathbb{E}[f(S)] + \gamma \ .$$

Intuitively, the set S in Definition 5 is the output of the streaming algorithms, and the set A is the set of "bad" elements in the sense that the approximation guarantee of the algorithm is with respect to $f(OPT \cup A)$ rather than $f(OPT)$ (for non-monotone functions $f(OPT \cup A)$ might be smaller than $f(OPT)$). Many previous algorithms satisfy Definition 5 even with $\gamma = 0$, when the set A is the set of all elements that are "seriously" considered by the algorithm at some point. Unfortunately, however, this set A can be quite large, and therefore, cannot be stored by the algorithm if we want it to remain a streaming algorithm. Chekuri et al. (2015) described a technique to bypass this issue by creating a tradeoff between the size of A and the value of γ. They do that by guessing the value of the optimal solution, and discarding elements whose marginal is very small compared to the guess. This shrinks the size of the elements that are "seriously" considered, and thus, the size of the set A, but requires a positive value for γ representing the value of elements of OPT that are discarded. By setting the parameters right, it is possible to keep both γ and $|A|$ reasonably small. The same technique can be used to get a similar result for the algorithms of (Buchbinder et al., 2019; Chakrabarti & Kale, 2015) as well. More generally, as far as we know, similar modifications can be used to make every currently existing streaming algorithm for submodular maximization fit Definition 5, and thus, our framework is applicable to all these algorithms.

We are now ready to describe the algorithm at the heart of our framework. This algorithm is given as Algorithm 1, and it assumes access to two procedures: (1) a data stream algorithm StreamingAlg for the problem of maximizing a non-negative submodular function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ subject to a constraint $(\mathcal{N}, \mathcal{I})$, and (2) an offline algorithm ConstrainedAlg for the same problem. For the data stream algorithm StreamingAlg we use the following, not very standard, semantics. Algorithm 1 has two ways to call StreamingAlg. In the first way, every time that Algorithm 1 would like to pass additional elements to StreamingAlg, it calls it with the set of these new elements, and StreamingAlg updates its internal data structures accordingly and returns a set including all the elements that it decided to remove from its memory. In the second way, once the stream ends, Algorithm 1 calls StreamingAlg (denoted by the subscript end) and passes to it any final elements it would like StreamingAlg to get. StreamingAlg then process these elements and returns three sets: the sets S and A produced by StreamingAlg (as described in Definition 5) and a set D consisting of all the elements that are still in the memory of StreamingAlg and did not end up in A. Figure 1 is a graphic representation of the flow of elements within Algorithm 1.

[3] We remind the reader that a data stream algorithm is any algorithm that receives its input in the form of a stream. A (semi-)streaming algorithm is a data stream algorithm whose space complexity is nearly linear.

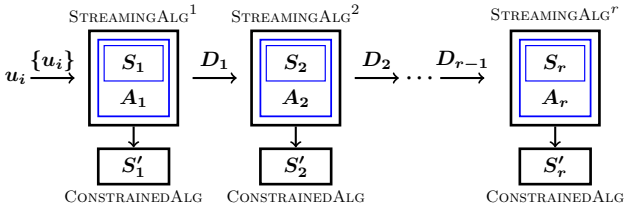


Figure 1. Schematic representation of Algorithm 1.

Algorithm 1: Non-monotone Data Stream Algorithm

Input: a positive integer r
Output: a set $S \in \mathcal{I}$
Initialize r independent copies of StreamingAlg: StreamingAlg$^{(1)}, \ldots,$ StreamingAlg$^{(r)}$.
while *there are more elements in the stream* **do**
  Let D_0 be a singleton set containing the next element of the stream.
  for $i = 1$ **to** r **do**
    $D_i \leftarrow$ StreamingAlg$^{(i)}(D_{i-1})$.
Let $D_0 \leftarrow \varnothing$.
for $i = 1$ **to** r **do**
  $[S_i, A_i, D_i] \leftarrow$ StreamingAlg$^{(i)}_{\mathsf{end}}(D_{i-1})$.
Let $S' \leftarrow$ ConstrainedAlg(A_i).
return *the set maximizing f among S' and $\{S_i\}_{i=1}^r$.*

It is clear that Algorithm 1 outputs a feasible solution. The following theorem gives additional properties of this algorithm. Its proof can be found in Appendix A.2.

Theorem 6. *Given an (α, γ)-approximation data stream algorithm* StreamingAlg *for maximizing a non-negative submodular function subject to some constraint and an offline β-approximation algorithm* ConstrainedAlg *for the same problem. There exists a data stream algorithm returning a feasible set S that obeys*

$$\mathbb{E}[f(S)] \geq \frac{(1 - 1/r) \cdot \mathtt{OPT} - \gamma}{\alpha + \beta} \ .$$

Furthermore,

- *this algorithm is deterministic if* StreamingAlg *and*

CONSTRAINEDALG *are both deterministic.*

- *the space complexity of this algorithm is upper bounded by* $O(r \cdot M_{\text{STREAMINGALG}} + M_{\text{CONSTRAINEDALG}})$*, where* $M_{\text{STREAMINGALG}}$ *and* $M_{\text{CONSTRAINEDALG}}$ *represent the space complexities of their matching algorithms under the assumption that the input for* STREAMINGALG *is a subset of the full input and the input for* CONSTRAINEDALG *is the set* A *produced by* STREAMINGALG *on some such subset.*

We note that the algorithm guaranteed by Theorem 6 is a streaming algorithm when STREAMINGALG is a streaming algorithm, the algorithm CONSTRAINEDALG is a nearly-linear space algorithm and r is upper bounded by a poly-log function (note that the sets S_i and A_i are produced by STREAMINGALG, and thus, their space complexity is already accounted for by the space complexity of STREAMINGALG).

In Appendix C we show that by plugging one of the versions of the algorithm of Chekuri et al. (2015) into our framework it is straightforward to get a deterministic streaming algorithm for the problem of maximizing a non-negative (not necessarily monotone) submodular function subject to a k-matchoid constraint whose approximation ratio is $(5 + 15\varepsilon)k + O(\sqrt{k})$ for every constant $\varepsilon > 0$, which is an improvement over the guarantee of the previous state-of-the-art deterministic streaming algorithm for this problem (also due to (Chekuri et al., 2015)) which has an approximation guarantee of $8k + \gamma$, where γ is the best offline approximation ratio for the same problem.

5. Streaming Submodular Maximization under a k-System Constraint

In this section, we formally prove our results for the problem of maximizing a non-negative submodular function $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ subject to a k-set system and k-extendible system constraint $(\mathcal{N}, \mathcal{I})$. A simple version of the algorithm we use to prove these results is given as Algorithm 2. This version assumes pre-access to a value ρ equal to the size of the largest independent set in $\mathcal{I}$ and a threshold τ estimating the value $M = \max_{u \in \mathcal{N}, \{u\} \in \mathcal{I}} f(\{u\})$. In Appendix D, we present a more involved version of our algorithm that does not need this pre-access, but has a space complexity larger than that of Algorithm 2 by a factor of $O(\log \rho + \log k)$—the approximation guarantee remains unchanged.

Intuitively, Algorithm 2 maintains ℓ independent sets E_i, where each one of these sets corresponds to a different range of marginal contributions: the larger i, the smaller the marginal contributions E_i is associated with. When an element u arrives, the algorithm calculates the marginal contribution $m(u)$ of u with respect to the union of the E_i sets,[4] and then adds u to the E_i corresponding to this marginal contribution, unless this violates the independence of this E_i. Once the entire input has been processed, the algorithm combines the E_i sets into h possible output sets $T_0, T_1, \ldots, T_{h-1}$. Each output sets T_j is constructed by greedily taking elements from the E_i sets obeying $i \equiv j \pmod h$ (the algorithm scans the sets obeying this condition in an increasing i order, which is a decreasing order with respect to the marginal contributions associated with these sets). The final output of the algorithm is simply the best set among the sets $T_0, T_1, \ldots, T_{h-1}$.

Algorithm 2: Streaming Algorithm for k-set Systems

Input: a threshold $\tau \in [M, 2M]$, the size ρ of the largest independent set, and the parameter k of the constraint. // $M = \max_{u \in \mathcal{N}, \{u\} \in \mathcal{I}} f(\{u\})$.
Output: a solution $T \in \mathcal{I}$
Let $\ell \leftarrow \lfloor \log_2(4\rho) \rfloor$ and $h \leftarrow \lceil \log_2(2k+1) \rceil$.
for $i = 0$ ***to*** ℓ **do** Initialize $E_i \leftarrow \varnothing$.
for *every element* u *arriving* **do**
  /* Adds u to a set $E_{i(u)}$ based on its marginal gain. */
  Let $m(u) \leftarrow f\left(u \mid \cup_{i=0}^{\ell} E_i\right)$.
  if $m(u) > 0$ **then** Let $i(u) \leftarrow \lfloor \log_2(\tau / m(u)) \rfloor$ **else** Let $i(u) \leftarrow \infty$.
  if $0 \leq i(u) \leq \ell$ *and* $E_{i(u)} + u \in \mathcal{I}$ **then** Update $E_{i(u)} \leftarrow E_{i(u)} + u$.
for $j = 0$ ***to*** $h - 1$ **do**
  Let $i \leftarrow j$ and $T_j \leftarrow \varnothing$.
  while $i \leq \ell$ **do**
    while *there is an element* $u \in E_i$ *such that* $T_j + u \in \mathcal{I}$ **do** Update $T_j \leftarrow T_j + u$.
    $i \leftarrow i + h$. // Greedily generates solution T_j from sets E_i for $i \equiv j \pmod h$.
return *the set* T *maximizing function* f *among sets* $T_0, T_1, \cdots, T_{h-1}$.

Let's denote $E = \cup_{i=0}^{\ell} E_i$. It is not difficult to argue that the value of $f(E)$ is large. In Lemma 7, we show that the value of the output set of the algorithm is proportional to $f(E)$ subject to a k-set system constraint.[5]

Lemma 7. *If* $(\mathcal{N}, \mathcal{I})$ *is a* k*-set system, then Algorithm 2 returns a set* T *such that*

$$f(T) \geq \frac{f(E \cup U) - \tau/4}{4kh(2k+1)} = \frac{f(E \cup U) - \tau/4}{O(k^2 \log k)}$$

[4] The notation $m(u)$ might suggest that the value $m(u)$ depends only on the identity of the element u. However, this is not the case. In fact, $m(u)$ might depend also on the set of elements that arrived before u and the order of their arrival.

[5] In the supplementary material, we provide a stronger version of this result for k-extendible constraints.

for every set $U \in \mathcal{I}$.

Following (see Lemma 8) is the key result that we use to prove Lemma 7. This lemma relates the value of $f(T_j)$ to the sum of the $m(u)$ values of the elements u that belong to the sets E_i that are combined to create T_j (recall that these are exactly the sets E_i for which $i \equiv j \pmod h$). Intuitively, the lemma holds because when Algorithm 2 adds elements of a set E_i to a set T_j, this increases the size of the set T_j to at least E_i/k (since the constraint is k-set system). Thus, either about $1/k$ of the elements of E_i are added to T_j, or the size of T_j before the addition of the elements of E_i is already significant compared to the size of E_i. Moreover, in the latter case, the elements of T_j can pay for the elements of E_i that they have blocked because they all have a relatively high value (as they originate in a set $E_{i'}$ for some $i' \leq i - h$).

Lemma 8. *If $(\mathcal{N}, \mathcal{I})$ is a k-set system, then for every integer $0 \leq j < h$ we have*

$$f(T_j \mid \varnothing) \geq \frac{1}{4k} \cdot \sum_{\substack{0 \leq i \leq \ell \\ i \equiv j \pmod h}} \sum_{u \in E_i} m(u) \ .$$

From the result of Lemma 7, we can directly guarantee the performance of our algorithm for monotone submodular functions. Furthermore, Lemma 7 implies that Algorithm 2 is a $(4kh(2k+1), \tau/4)$-approximation data stream algorithm subject to a k-set system constraint.[6] Therefore, we can use the framework of Algorithm 1 to make this algorithm suitable for non-monotone submodular functions. The following two theorems describe the guarantees we provide for Algorithm 2, and their complete proofs can be found in Appendix A.3.

Theorem 9. *There is a streaming $O(k^2 \log k) = \tilde{O}(k^2)$-approximation algorithm for maximizing a non-negative submodular function subject to a k-set system constraint.*

Theorem 10. *There is a streaming $O(k \log k) = \tilde{O}(k)$-approximation algorithm for maximizing a non-negative submodular function subject to a k-extendible system constraint.*

6. Experiments

In this section, we compare our proposed algorithms (both monotone and non-monotone versions) with two other groups of algorithms: other streaming algorithms and state-of-the-art *offline* algorithms.

We consider three baseline streaming algorithms: i) The **Streaming-Greedy** algorithm: this algorithm keeps a solution S which is initially set to the empty set. For every incoming element u, it is added to the set S if this does not violate feasibility (i.e., $S \cup \{u\} \in \mathcal{I}$). ii) The **Preemption** algorithm: Inspired by the streaming algorithms of (Chekuri et al., 2015; Buchbinder et al., 2019), we consider a heuristic preemptive algorithm. For every incoming element u, given that S is the current solution, this algorithm generates a set $U \subseteq S$ such that $(S \cup \{u\}) \setminus U$ is feasible under the non-knapsack constraints. The element u is then added to the solution in exchange for the elements of U if this does not violate the knapsack constraints and the exchange is beneficial in the sense that $f(u \mid S) \geq \sum_{u' \in U} f(u' : S)$, where $f(u' : S) = f(u' \mid S')$ for $S' = \{s \in S : \text{element } s \text{ arrived before } u'\}$. For more detail refer to (Chekuri et al., 2015; Feldman et al., 2018). iii) The **Sieve-Streaming** algorithm: this heuristic algorithm is implemented based on the ideas of (Badanidiyuru et al., 2014). In the first step, it finds an accurate estimation of $\texttt{OPT}$. Then, each incoming element u is added to the solution S if $S \cup \{u\} \in \mathcal{I}$ and $f(u \mid S) \geq \texttt{OPT}/(2\rho)$, where ρ is the maximum cardinality of a feasible solution.

For the offline algorithms we consider 1) the vanilla greedy algorithm (referred to as **Greedy**), 2) **Fast** (Badanidiyuru & Vondrák, 2014) and **FANTOM** (Mirzasoleiman et al., 2016a). Both Fast and FANTOM are designed to maximize submodular functions under a k-set system constraint combined with ℓ knapsack constraints.

Sections 6.1 and 6.2 compare the above algorithms on tasks of maximizing linear and cut objective functions over instances produced using synthetic and real-world data, respectively. Then, in Section 6.3 and Appendices E.4 and E.5, we evaluate the performance of the same algorithms on three different real-world applications. In a movie recommendation system application, we are given movie ratings from users, and our goal is to recommend diverse movies from different genres. In a Yelp location summarization application, we are given thousands of business locations with several related attributes. Our objective is to find a good summary of the locations from six different cities. In a third application, our goal is to generate real-time summaries for Twitter feeds of several news agencies.

6.1. Independent set

In the experiments of this section we define submodular functions over the nodes of a given graph $G = (V, E)$, and consider the maximization of such functions subject to an independent set constraint, i.e., we are not allowed to select a set of vertices if there is any edge of the graph connecting any two of these vertices. It is easy to show that this constraint is a $d_{\max}$-extendible system, where $d_{\max}$ is the maximum degree in graph G. In our experiments in this section, we use two types of synthetically generated random graphs: Erdős Rény graphs (Erdős & Rény,

[6] In the supplementary material, we prove that Algorithm 2 is a $(4h(2k+1), \tau/4)$-approximation data stream algorithm subject to a k-extendible constraint.

1960) and Watts–Strogatz graphs (Watts & Strogatz, 1998). For the Erdős Rény graphs we vary in our experiments the probability p that each possible edge is included in the graph (independently from every other edge), and for the Watts-Strogatz graphs we vary the rewiring probability β. The number of nodes is set to $n = 2000$ in all the graphs, and in the Watts–Strogatz model each node is connected to $k = 100$ nearest neighbors in the ring structure.

In our first experiment of the section, we study the maximization of the following monotone linear function $f(S) = \sum_{u \in S} w_u$, where $S \subseteq V$ and w_u is the weight of node $u \in S$. Figs. 2a and 2b compare different algorithms for optimizing this function over a random graph chosen from the above discussed random graph models. We observe that our proposed algorithm consistently outperforms the other baseline streaming algorithms. We also observe that the performance our algorithm is comparable with (or even better at times) the greedy algorithm, which is provably optimal for the maximization of linear functions subject to k-extendible constraints (Feldman et al., 2017).

In the second experiment, we study the non-monotone submodular graph-cut function: $f(S) = \sum_{u \in S} \sum_{v \in V \setminus S} w_{u,v}$, where $w_{u,v}$ is the weight of the edge $e = (u, v)$. Again, in Figs. 2c and 2d we observe that the solutions provided by our algorithms are clearly better than those produced by the other streaming algorithms. Furthermore, the non-monotone version of our algorithm always outperforms the monotone algorithm. We also note that the for this non-monotone submodular function, the vanilla greedy algorithm performs very poorly (which is consistent with the lack of a theoretical guarantee for this algorithm for such functions).

6.2. Graph Planarity with Knapsack

In the experiment of this section, our objective is to maximize a linear function over the edges of a graph $G = (V, E)$. For the constraint, we require that an independent set of edges corresponds to a planar sub-graph of G, and in addition, it satisfies a given knapsack constraint. We remind the reader that a graph is planar if it can be embedded in the plane. Furthermore, a knapsack constraint is defined by a cost function $c \colon \mathcal{N} \to \mathbb{R}_{\geq 0}$, and we say that a set $S \subseteq \mathcal{N}$ satisfies the knapsack constraint if $c(S) = \sum_{e \in S} c(e) \leq b$ for a given knapsack budget b. In Appendix E.1 we explain why the above mentioned constraint is k-set system for a (relatively) modest value of k.[7].

For the objective function f, we use the monotone linear function: $f(S) = \sum_{e \in S} w_e \quad \forall S \subseteq E$, where w_e is the weight of edge $e \in S$, and for simplicity, we set all these weights to 1. The knapsack cost of each edge $e = (u, v) \in E$ is chosen to be proportional to $\max(1, d_u - q)$, where d_u is the degree of node u in graph G and $q = 6$, and the costs are normalized so that $\sum_{e \in E} c_e = |V|$, where c_e represents the knapsack cost of edge e.

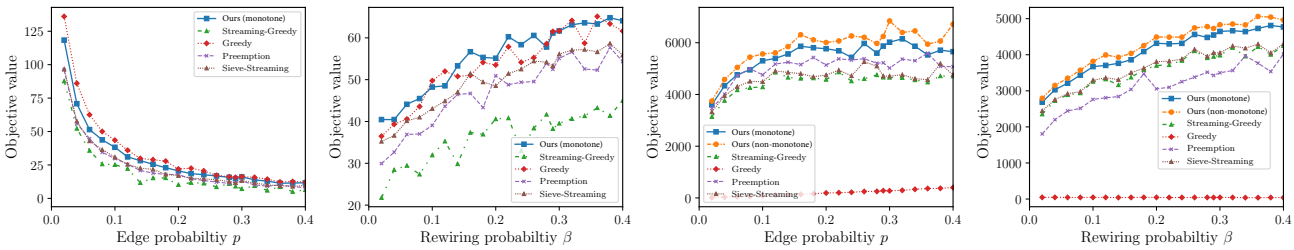
In the experiment, we use two real-world networks from (Leskovec & Krevl, 2014) as the graph, and vary the knapsack budget between 0 and 1 (note that the normalization gives this range of budgets an intuitive meaning). In Figs. 3a and 3b we compare the performance of our streaming algorithm with the performance of Streaming Greedy and Sieve Streaming. One can observe that our algorithm outperforms the two other baselines. Due to the prohibitive computational complexity of the offline algorithms, we do not report their results for this experiment. Furthermore, as it is not clear how to execute a preemptive streaming algorithm under a planarity constraint, we did not include a version of the Preemption algorithm in this experiment. Further experiments in this setting can be found in Appendix E.2.

6.3. Movie Recommendation

In the movie recommendation application, our goal is to select a diverse set of movies subject to constraints that can be adjusted by the user. The dataset for this experiment contains 1793 movies from the genres: Adventure, Animation, and Fantasy (note that a single movie may be identified with multiple genres). The user may specify an upper limit m on the number of movies in the set we recommend for them, as well as an upper limit m_i on the number of movies from each genre. For simplicity, we use a single value for all m_i and refer to this value as the genre limit. It is easy to show that this set of constraints forms a 3-extendible system. In addition, we enforce two knapsack constraints. For the first knapsack constraint c_1, the cost of each movie is proportional to the absolute difference between the release year of the movie and the year 1985 (the implicit goal of this constraint is to pick movies with a release year which is as close as possible to the year 1985). For the second knapsack constraint c_2, the cost of each movie is proportional to the difference between the maximum possible rating (which is 10) and the rating of the particular movie—here the goal is to pick movies with higher ratings. More formally, for a movie $v \in \mathcal{N}$, we have: $c_1(v) \propto |1985 - \text{year}_v|$ and $c_2(v) \propto (10 - \text{rating}_v)$. Here, year_v and rating_v, respectively, denote the release year and IMDb rating of movie v. We normalize the costs in both knapsacks constraints so that the average cost of each movie is $1/10$, i.e., $\frac{\sum_{v \in \mathcal{N}} c_i(V)}{|\mathcal{N}|} = 1/10$, and we set the knapsack budgets to 1; which intuitively means that a feasible set can contain no more than about 10% of the movies.

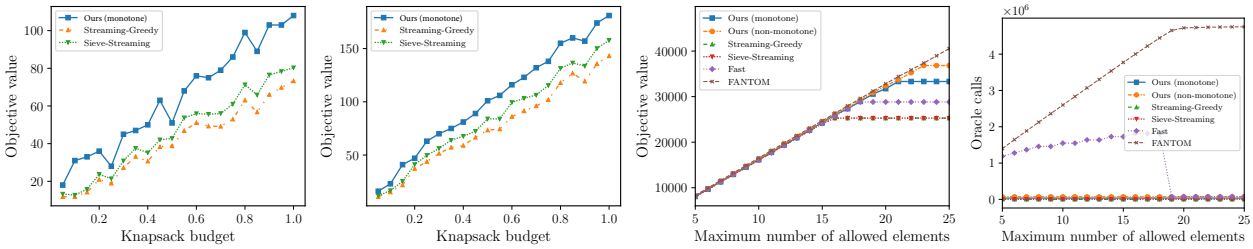
In our experiments, we try to maximize two kinds of objective functions (each trying to capture diversity in a different way) subject to these constraints, and we vary the upper limit m on the number of movies in the recommended

[7] We would like to thank Chandra Chekuri for pointing out some parts of this explanation to us.



(a) Erdős Rény (linear) (b) Watts–Strogatz (linear) (c) Erdős Rény (cut) (d) Watts–Strogatz (cut)

Figure 2. For Erdős Rény graphs p is the probability of having an edge between any two nodes. For Watts–Strogatz graphs β is the probability of rewiring of each edge.



(a) EU Email (b) Facebook ego network (c) Non-monotone function (d) Non-monotone function

Figure 3. (a) and (b): Planarity with knapsack (linear objective function). (c) and (d): Movie recommendation with two knapsacks.

set of movies. Both objective functions are based a set of attributes calculated for each movie using the method described in (Lindgren et al., 2015), and both objective functions are non-negative and submodular. However, one of them is monotone, and the other is not (guaranteed to be) monotone. The experimental result for the monotone function is given in Appendix E.3.

An intuitive utility function for choosing a diverse set of movies S is the following not necessarily monotone submodular function

$$f(S) = \sum_{i \in S} \sum_{j \in \mathcal{N}} M_{i,j} - \sum_{i \in S} \sum_{j \in S} M_{i,j} \quad , \tag{3}$$

where $\mathcal{N}$ is the set of all movies and $M_{i,j}$ is the non-negative similarity score between movies $i, j \in \mathcal{N}$ as defined in the previous section. It is beneficial to note that the first term is a sum-coverage function that captures the representativeness of the selected set, and the second term is a dispersion function penalizing similarity within S (Feldman et al., 2017).

In our experiment with this function as the object, we set the genre limit to 20. In Figs. 3c and 3d, we observe that i) our streaming algorithm returns solutions with higher utilities compared to the baseline streaming algorithms, ii) the non-monotone version of our algorithm clearly outperforms the monotone one for this non-monotone function, and iii) the quality of the solutions returned by our algorithms is comparable with the quality obtained by offline algorithms.

7. Conclusion

In this paper, we have proposed a novel framework for converting streaming algorithms for monotone submodular maximization into streaming algorithms for non-monotone submodular maximization, which immediately led us to the currently tightest deterministic approximation ratio for submodular maximization subject to a k-matchoid constraint. We also proposed the first streaming algorithm for monotone submodular maximization subject to k-extendible and k-set system constraints, which (together with our proposed framework), yields approximation ratios of $O(k \log k)$ and $O(k^2 \log k)$ for maximization of general non-negative submodular functions subject to the above constraints, respectively. Finally, we extensively evaluated the empirical performance of our algorithm against the existing work in a series of experiments including finding the maximum independent set in randomly generated graphs, maximizing linear functions over social networks, movie recommendation, Yelp location summarization, and Twitter data summarization.

Acknowledgements

The research of Moran Feldman and Ran Haba was partially supported by ISF grant 1357/16. Amin Karbasi is partially supported by NSF (IIS- 1845032), ONR (N00014-19-1-2406), and AFOSR (FA9550-18-1-0160).

References

Alaluf, N. and Feldman, M. Making a sieve random: Improved semi-streaming algorithm for submodular maximization under a cardinality constraint. *CoRR*, abs/1906.11237, 2019.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *ACM-SIAM symposium on Discrete algorithms (SODA)*, pp. 1497–1514, 2014.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming Submodular Maximization:Massive Data Summarization on the Fly. In *International Conference on Knowledge Discovery and Data Mining, KDD*, pp. 671–680, 2014.

Badanidiyuru, A., Karbasi, A., Kazemi, E., and Vondrák, J. Submodular maximization through barrier functions. *arXiv preprint arXiv:2002.03523*, 2020.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. Submodular Maximization with Cardinality Constraints. In *SODA*, pp. 1433–1452, 2014.

Buchbinder, N., Feldman, M., and Schwartz, R. Online Submodular Maximization with Preemption. *ACM Trans. Algorithms*, 15(3):30:1–30:31, 2019.

Călinescu, G., Chekuri, C., Pál, M., and Vondrák, J. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

Chakrabarti, A. and Kale, S. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1–2):225–247, 2015.

Chekuri, C., Gupta, S., and Quanrud, K. Streaming algorithms for submodular function maximization. In *ICALP*, pp. 318–330, 2015.

Crouch, M. and Stubbs, D. M. Improved streaming algorithms for weighted matching, via unweighted matching. In *APPROX*, pp. 96–104, 2014.

Das, A. and Kempe, D. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *International Conference on International Conference on Machine Learning*, pp. 1057–1064, 2011.

Dasgupta, A., Kumar, R., and Ravi, S. Summarization through submodularity and dispersion. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1014–1022, 2013.

De, A., Koley, P., Ganguly, N., and Gomez-Rodriguez, M. Regression Under Human Assistance. *CoRR*, abs/1909.02963, 2019.

Elenberg, E. R., Dimakis, A. G., Feldman, M., and Karbasi, A. Streaming Weak Submodularity: Interpreting Neural Networks on the Fly. In *Advances in Neural Information Processing Systems*, pp. 4047–4057, 2017.

Elhamifar, E. and Clara De Paolis Kaluza, M. Online summarization via submodular and convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1783–1791, 2017.

Ene, A., Nguyen, H. L., and Suh, A. An optimal streaming algorithm for non-monotone submodular maximization. *CoRR*, abs/1911.12959, 2019.

Erdős, P. and Rény, A. On the Evolution of Random Graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

Feldman, M. and Haba, R. Almost Optimal Semi-streaming Maximization for k-Extendible Systems. *arXiv preprint arXiv:1906.04449*, 2019.

Feldman, M., Harshaw, C., and Karbasi, A. Greed is good: Near-optimal submodular maximization via greedy optimization. In *COLT*, pp. 758–784, 2017.

Feldman, M., Karbasi, A., and Kazemi, E. Do less, get more: Streaming submodular maximization with subsampling. In *NeurIPS*, pp. 730–740, 2018.

Feldman, M., Norouzi-Fard, A., Svensson, O., and Zenklusen, R. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Symposium on Theory of Computing (STOC)*, pp. 1363–1374, 2020.

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

Frieze, A. M. A cost function property for plant location problems. *Mathematical Programming*, 7(1):245–248, 1974.

Gupta, A., Roth, A., Schoenebeck, G., and Talwar, K. Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms. In *WINE*, pp. 246–257, 2010.

Herbrich, R., Lawrence, N. D., and Seeger, M. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*, pp. 625–632, 2003.

Kazemi, E., Zadimoghaddam, M., and Karbasi, A. Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints. In *International Conference on Machine Learning (ICML)*, pp. 2549–2558, 2018.

Kazemi, E., Mitrovic, M., Zadimoghaddam, M., Lattanzi, S., and Karbasi, A. Submodular Streaming in All Its Glory: Tight Approximation, Minimum Memory and Low Adaptive Complexity. In *International Conference on Machine Learning (ICML)*, pp. 3311–3320, 2019.

Kazemi, E., Minaee, S., Feldman, M., and Karbasi, A. Regularized submodular maximization at scale. *arXiv preprint arXiv:2002.03503*, 2020.

Kirchhoff, K. and Bilmes, J. Submodularity for data selection in statistical machine translation. In *Proceedings of EMNLP*, 2014.

Krause, A. and Golovin, D. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2012.

Lei, Q., Wu, L., Chen, P.-Y., Dimakis, A., Dhillon, I., and Witbrock, M. Discrete Adversarial Attacks and Submodular Optimization with Applications to Text Classification. *Systems and Machine Learning (SysML)*, 2019.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data, June 2014.

Libbrecht, M. W., Bilmes, J. A., and Noble, W. S. Choosing non-redundant representative subsets of protein sequence data sets using submodular optimization. *Proteins: Structure, Function, and Bioinformatics*, 2018. ISSN 1097-0134.

Lindgren, E. M., Wu, S., and Dimakis, A. G. Sparse and greedy: Sparsifying submodular facility location problems. In *NIPS Workshop on Optimization for Machine Learning*, 2015.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed Submodular Maximization: Identifying Representative Elements in Massive Data. In *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

Mirzasoleiman, B., Badanidiyuru, A., and Karbasi, A. Fast Constrained Submodular Maximization: Personalized Data Summarization. In *ICML*, pp. 1358–1367, 2016a.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed Submodular Maximization. *Journal of Machine Learning Research*, 17:238:1–238:44, 2016b.

Mirzasoleiman, B., Jegelka, S., and Krause, A. Streaming Non-Monotone Submodular Maximization: Personalized Video Summarization on the Fly. In *AAAI Conference on Artificial Intelligence,*, pp. 1379–1386, 2018.

Mitrovic, M., Kazemi, E., Zadimoghaddam, M., and Karbasi, A. Data Summarization at Scale: A Two-Stage Submodular Approach. In *International Conference on Machine Learning (ICML)*, pp. 3593–3602, 2018.

Mitrovic, M., Kazemi, E., Feldman, M., Krause, A., and Karbasi, A. Adaptive sequence submodularity. In *Advances in Neural Information Processing Systems*, pp. 5352–5363, 2019.

Nemhauser, G. L. and Wolsey, L. A. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

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

Norouzi-Fard, A., Tarnawski, J., Mitrovic, S., Zandieh, A., Mousavifar, A., and Svensson, O. Beyond 1/2-approximation for submodular maximization on massive data streams. In *International Conference on Machine Learning (ICML)*, pp. 3826–3835, 2018.

Salehi, M., Karbasi, A., Scheinost, D., and Constable, R. T. A Submodular Approach to Create Individualized Parcellations of the Human Brain. In *MICCAI*, pp. 478–485, 2017.

Singla, A., Bogunovic, I., Bartók, G., Karbasi, A., and Krause, A. Near-Optimally Teaching the Crowd to Classify. In *International Conference on Machine Learning (ICML)*, 2014.

Tschiatschek, S., Iyer, R. K., Wei, H., and Bilmes, J. A. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pp. 1413–1421, 2014.

Tschiatschek, S., Singla, A., and Krause, A. Selecting sequences of items via submodular maximization. In *AAAI Conference on Artificial Intelligence*, 2017.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

Yelp. Yelp Academic Dataset. https://www.kaggle.com/yelp-dataset/yelp-dataset, 2019a.

Yelp. Yelp Dataset. https://www.yelp.com/dataset, 2019b.