The Power of Subsampling in Submodular Maximization*

Christopher Harshaw[1], Ehsan Kazemi[2], Moran Feldman[3], and Amin Karbasi[4]

[1]Yale University, Department of Computer Science
[2]Google, Zürich
[3]University of Haifa, Department of Computer Science
[4]Yale University, Department of Electrical Engineering

Abstract

We propose subsampling as a unified algorithmic technique for submodular maximization in centralized and online settings. The idea is simple: independently sample elements from the ground set, and use simple combinatorial techniques (such as greedy or local search) on these sampled elements. We show that this approach leads to optimal/state-of-the-art results despite being much simpler than existing methods. In the usual offline setting, we present SAMPLEGREEDY, which obtains a $(p + 2 + o(1))$-approximation for maximizing a submodular function subject to a p-extendible system using $O(n + nk/p)$ evaluation and feasibility queries, where k is the size of the largest feasible set. The approximation ratio improves to $p + 1$ and p for monotone submodular and linear objectives, respectively. In the streaming setting, we present SAMPLE-STREAMING, which obtains a $(4p + 2 - o(1))$-approximation for maximizing a submodular function subject to a p-matchoid using $O(k)$ memory and $O(km/p)$ evaluation and feasibility queries per element, where m is the number of matroids defining the p-matchoid. The approximation ratio improves to $4p$ for monotone submodular objectives. We empirically demonstrate the effectiveness of our algorithms on video summarization, location summarization, and movie recommendation tasks.

1   Introduction

Submodular functions have played a celebrated role in both the theory of discrete optimization and practical modeling scenarios. Submodular functions are defined by a diminishing returns property, which makes this function class natural for modeling many applications in a wide variety of domains, from economics to machine learning. Constrained submodular maximization has found numerous applications, including viral marketing [Kempe et al., 2003], network monitoring [Leskovec et al., 2007, Gomez-Rodriguez et al., 2010], sensor placement and information gathering [Guestrin et al., 2005], news article recommendation [El-Arini et al., 2009], movie recommendation [Mitrovic et al., 2019, Haba et al., 2020], active set selection in non-parametric learning [Mirzasoleiman et al., 2016b], image summarization [Tschiatschek et al., 2014, Kazemi et al., 2020], location summarization [Badanidiyuru et al., 2020], corpus summarization [Lin and

---

*The results presented in this paper previously appeared in COLT 2017 [Feldman et al., 2017] and NeurIPS 2018 [Feldman et al., 2018] in the form of extended abstracts. We note, however, that some results of Feldman et al. [2017] were not included in the current paper since they are unrelated to its central theme. Nevertheless, for completeness, we reference these results while discussing the related work.

Bilmes, 2011, Kirchhoff and Bilmes, 2014, Sipos et al., 2012], fMRI parcellation [Salehi et al., 2017], ensuring privacy and fairness [Kazemi et al., 2018, Mitrovic et al., 2017], two-stage sublinear data summarization [Balkanski et al., 2016, Stan et al., 2017, Mitrovic et al., 2018] and removing redundant elements from DNA sequencing [Libbrecht et al., 2018]. For a more detailed description of theses applications in machine learning and signal processing, we refer the interested reader to Tohidi et al. [2020].

Although producing an exactly optimal solution for constrained submodular maximization is computationally hard [Nemhauser and Wolsey, 1978], the seminal work of Nemhauser et al. [1978] showed that the natural greedy algorithm produces a $(1-e^{-1})^{-1}$-approximate solution when the objective function is monotone submodular and the constraint set is a cardinality constraint; however, the greedy algorithm may perform much worse in more complex scenarios, e.g., when the objective is non-monotone or the constraint is more involved. Most existing algorithms developed for these more complicated settings can be grouped into a few categories: repeated greedy procedures, local search techniques, and relax-and-round methods which go through a continuous relaxation of the problem. Unfortunately, these techniques tend to be quite slow, and moreover, some of them are quite complex, making their implementation challenging Buchbinder and Feldman [2018].

In this paper, we propose *subsampling* as a simpler alternative to existing algorithmic techniques. In particular, we present two algorithms which use subsampling to achieve better approximation guarantees than existing techniques at a fraction of the computational costs. At the heart of our algorithms is a carefully designed—but simple to implement—subsampling of the ground set. Interestingly, our algorithms naturally produce state-of-the-art results for both monotone and non-monotone objective functions; which is rare for a submodular maximization algorithm.

Our first algorithm, SAMPLEGREEDY, is designed for maximizing a submodular function subject to a p-extendible system. The algorithm achieves a $(p+1)^2/p = p+2+o(1)$ approximation guarantee (which is nearly tight for this problem by a result of Feldman et al. [2017]) and uses only $O(n + \frac{nk}{p})$ function evaluations and feasibility queries, where n is the size of the ground set and k is the largest feasible set. The technique is simple: independently sample elements from the ground set and run the greedy algorithm. Moreover, the approximation guarantee improves to $p + 1$ or p when the function is monotone or linear, respectively. Our second algorithm, SAMPLE-STREAMING, is designed for maximizing a submodular function subject to a p-matchoid constraint in the streaming setting where elements arrive one at a time and only a small working memory is kept. SAMPLE-STREAMING achieves a $4p+2-o(1)$ approximation ratio in this setting and uses $O(k)$ memory. To process the arrival of every element the algorithm uses, in expectation, $O(km/p)$ function evaluations and matroid feasibility queries, where m is the number of matroids used to define the p-matchoid. We also note that the approximation ratio improves to $4p$ for monotone functions.

We empirically demonstrate the effectiveness of our subsampling based algorithms for video summarization and movie recommendation tasks with real datasets. We show that our algorithms are competitive with respect to existing algorithms, but require a fraction of the computational cost.

Organization. Section 2 contains a brief summary of related works on constrained submodular maximization as well as a comparison of our new results with these works. We review preliminary definitions in Section 3. Section 4 presents and analyzes our subsampling algorithms, focusing on the offline algorithm in Sections 4.1 and the streaming algorithm in Section 4.2. In Section 5, we empirically evaluate the performance of our algorithms against existing methods on real datasets. Finally, we provide concluding remarks in Section 6.

Algorithm	Function	Approx. Ratio	Query Complexity	Reference
Deterministic	Monotone	$p+1$	$O(nk)$	Fisher et al. [1978]
Deterministic	Non-monotone	$\approx 2p$	$O(nkp)$	Mirzasoleiman et al. [2016a]
Deterministic	Non-monotone	$\approx 3p$	$O(nkp)$	Gupta et al. [2010]
Deterministic	Non-monotone	$p+O(\sqrt{p})$	$O(nk\sqrt{p})$	Feldman et al. [2017]
Randomized	Non-monotone	$\frac{(p+1)^2}{p} = p+2+o(1)$	$O(n+nk/p)$	SampleGreedy (this paper)
Randomized	Monotone	$p+1$	$O(n+nk/p)$	SampleGreedy (this paper)

Table 1: Greedy algorithms for submodular maximization subject to a p-extendible constraint.

2 Related Work

In this section, we briefly survey the most relevant related work on constrained submodular function maximization. In what follows, n is the size of the ground set, p is in reference to p-extendible systems or p-matchoids, and k is the size of the largest independent set.[1]

The most relevant works for comparing our SampleGreedy algorithm are the repeated greedy algorithms, which have been historically developed for a slightly broader class of constraints known as p-systems. Fisher et al. [1978] showed that the natural greedy algorithm achieves a $(p+1)$ approximation for maximizing a monotone submodular function subject to a p-system. Algorithms for the non-monotone variant of this problem were developed only much more recently and rely on repeated applications of the greedy algorithm. Gupta et al. [2010] showed that iteratively running the greedy algorithm on the constrained problem followed by an unconstrained optimization on the greedy solution results in an approximation guarantee of roughly $3p$ for general non-monotone submodular functions, while requiring $O(nkp)$ function evaluations and independence oracles queries. Using a different analysis, Mirzasoleiman et al. [2016a] improved the approximation guarantee of this algorithm to roughly $2p$. Feldman et al. [2017] showed that an improved approximation guarantee of $p+O(\sqrt{p})$ is possible with fewer iterations of the repeated greedy procedure. The main drawback of all these algorithms is the large number of function evaluation and independence oracle queries they require, which grows unfavorably with p. Our proposed subsampling based algorithm, SampleGreedy, significantly improves upon these algorithms in the case of p-extendible systems in two ways: the oracle complexity is greatly reduced and the approximation guarantee is improved. A summary and comparison of these algorithms are presented in Table 1.

Local search algorithms have also been proposed for maximization over various subclasses of p-extendible systems. Lee et al. [2010b] developed a local search method which attains a $p+\varepsilon$ approximation for maximizing a monotone submodular function subject to the intersection of p matroids using a number of evaluations and independence oracle queries which is polynomial in n and exponential in $\frac{1}{\varepsilon}$. They also showed how to use this algorithm to obtain a $p+1+\frac{1}{p+1}+\varepsilon$ approximation for non-monotone objectives, improving over a $(p+2+\frac{1}{p}+\varepsilon)$ approximation due to Lee et al. [2010a]. Feldman et al. [2011b] showed that the same results can also be obtained for maximization over a p-exchange system, which is a different subclass of p-extendible systems. For $p \geq 4$, the last approximation guarantee was improved to $(p+3)/2+\varepsilon$ by Ward [2012]. Despite running in polynomial time, these local search algorithms have very large oracle queries complexity, and so they are mostly of theoretical interest.

There has also been a long sequence of works which aim to obtain a tighter approximation

---

[1]These terms, along with a precise notion of oracle complexity, are defined in Section 3.

guarantees for the special case of matroid constraints. Such methods rely on approximately optimizing continuous extensions of the discrete submodular objective, followed by rounding to obtain a discrete solution. The seminal work of Călinescu et al. [2011] showed that this technique achieves the tight $(1 - e^{-1})^{-1}$ approximation ratio for maximizing a monotone submodular function subject to a matroid constraint. In the non-monotone setting, a long series of work [Vondrák, 2013, Oveis Gharan and Vondrák, 2011, Feldman et al., 2011a, Ene and Nguyen, 2016, Buchbinder and Feldman, 2019] has further developed these techniques to obtain a 2.59 approximation ratio, but the best inapproximability result is still slightly further away at 2.09 [Oveis Gharan and Vondrák, 2011]. Although these algorithms achieve tighter approximation guarantees for the special case of matroids, they suffer from a high evaluation oracle complexity due to the sampling techniques they use to obtain gradient estimates for the continuous extension. A more recent line of work suggests some techniques to (partially) remedy this problem [Badanidiyuru and Vondrák, 2014, Buchbinder et al., 2016, Mokhtari et al., 2018, Korula et al., 2018, Buchbinder et al., 2019, Ene and Nguyen, 2018]. Finally, we remark that [Mirzasoleiman et al., 2015] devised a different randomized subsampling technique which achieves a $(1 - e^{-1} - \varepsilon)^{-1}$ approximation ratio using $O(n \log \frac{1}{\varepsilon})$ evaluation queries for monotone submodular objectives under the cardinality constraint, improving upon the query complexity of the greedy algorithm.

Additional recent work in constrained submodular maximization has focused on the streaming environment, where data points appear one at a time and centralized storage capacity is limited. A streaming algorithm for monotone submodular maximization under a cardinality constraint was presented by Badanidiyuru et al. [2014], which achieves a $1/2 - \varepsilon$ approximation using $O(\varepsilon^{-1} k \log k)$ memory. Recently, Kazemi et al. [2019] presented modification of this algorithm which reduces the memory complexity to $O(k/\varepsilon)$. A different series of work [Chakrabarti and Kale, 2015, Chekuri et al., 2015] used a different technique to provide a $4p$ approximation for monotone submodular maximization subject to p-matchoid constraints. The first streaming algorithm for non-monotone submodular maximization was given by Buchbinder et al. [2015], whose randomized algorithm achieves 11.197 approximation for non-monotone maximization under a cardinality constraint. This was shortly after improved by Chekuri et al. [2015], who presented a randomized streaming algorithm for non-monotone maximization under p-matchoid constraints which achieves an approximation ratio of $(5p + 2 + 1/p)/(1 - \varepsilon)$ and a deterministic algorithm which achieves a slightly worse approximation ratio of $(9p + O(\sqrt{p}))/(1 - \varepsilon)$ but is more memory and update efficient. Recently, Mirzasoleiman et al. [2018] proposed a deterministic algorithm which they claim achieves an approximation ratio of $4p + 4\sqrt{p} + 1$ and uses $O(k\sqrt{p})$ memory; however, Haba et al. [2020] pointed out several errors in their analysis and so the guarantees of Mirzasoleiman et al. [2018] may not hold. While the monotone algorithms mentioned above are quite efficient in terms of memory and update cost, the non-monotone algorithms are much less efficient in these aspects, having unfavorable dependence on p or ε terms. In contrast, our randomized streaming algorithm, SAMPLE-STREAMING, achieves an improved approximation ratio of $4p + 2 - o(1)$ for non-monotone maximization over a p-matchoid constraint using only $O(k)$ memory and $O(km/p)$ expected evaluation and independence queries per iteration. With a minor modification to the algorithm, this approximation ratio improves to $4p$ for monotone functions. A summary and comparison of these algorithms is given in Table 2

---

[1]The memory and query complexities of the algorithm of Mirzasoleiman et al. [2018] have been calculated based on the corresponding complexities of the algorithm of [Chekuri et al., 2015] for monotone objectives and the properties of the reduction used by [Mirzasoleiman et al., 2018]. We note that these complexities do not match the memory and query complexities stated by [Mirzasoleiman et al., 2018] for their algorithm.

Algorithm	Function	Approx. Ratio	Memory	Queries per Element	Reference
Deterministic	Monotone	$4p$	$O(k)$	$O(km)$	Chekuri et al. [2015]
Randomized	Non-monotone	$\frac{5p+2+1/p}{1-\varepsilon}$	$O(\frac{k}{\varepsilon^2}\log\frac{k}{\varepsilon})$	$O(\frac{k^2 m}{\varepsilon^2}\log\frac{k}{\varepsilon})$	Chekuri et al. [2015]
Deterministic	Non-monotone	$\frac{9p+O(\sqrt{p})}{1-\varepsilon}$	$O(\frac{k}{\varepsilon}\log\frac{k}{\varepsilon})$	$O(\frac{km}{\varepsilon}\log\frac{k}{\varepsilon})$	Chekuri et al. [2015]
Randomized	Monotone	$4p$	$O(k)$	$O(km/p)$	SAMPLE-STREAMING(this paper)
Randomized	Non-monotone	$4p + 2 - o(1)$	$O(k)$	$O(km/p)$	SAMPLE-STREAMING(this paper)

Table 2: Streaming algorithms for submodular maximization subject to a p-matchoid constraint.

3 Preliminaries

In this section, we describe the mathematical formulation of the constrained submodular maximization problem and preliminary definitions.

Let $\mathcal{N}$ be a finite set of size n, which we refer to as the *ground set*. The objective functions are real-valued set functions of the form $f : 2^{\mathcal{N}} \to \mathbb{R}$, which assign a real number to each set $S \subseteq \mathcal{N}$. Such a function f is *submodular* if

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \tag{1}$$

for all sets $A \subseteq B \subseteq \mathcal{N}$ and element $e \notin B$. Inequality (1) is also referred to as the *diminishing returns property*. Indeed, when f is interpreted as a utility, (1) states that the marginal gain in utility of an element $e \in \mathcal{N}$ decreases as the current set grows. For shorthand, we write the marginal gain of an element as $f(e \mid S) \triangleq f(S \cup \{e\}) - f(S)$ and the marginal gain of adding an entire set as $f(A \mid S) \triangleq f(S \cup A) - f(S)$. A function f is *monotone* if $f(A) \leq f(B)$ for all sets $A \subseteq B$. A function f is *linear* if (1) holds with equality for all $A \subseteq B$ and $e \notin B$.

We now describe the structure of the constraints we consider in this paper. Given a ground set $\mathcal{N}$ and a collection of sets $\mathcal{I} \subseteq 2^{\mathcal{N}}$, we say that the pair $(\mathcal{N}, \mathcal{I})$ is an *independence system* if $\varnothing \in \mathcal{I}$ and $A \subseteq B$, $B \in \mathcal{I}$ implies that $A \in \mathcal{I}$. A set $A \in \mathcal{I}$ is called *independent*, and a set $B \notin \mathcal{I}$ is called *dependent*. An independent set $A \in \mathcal{I}$ 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 $B = A$. Given an independent set $A \in \mathcal{I}$, an *extension* is an independent set $B \in \mathcal{I}$ that contains A, i.e., $A \subseteq B$. There is a hierarchy of classes of independence systems which are considered in the literature as constraint families. Our results only require two such classes: p-extendible systems and p-matchoids. However, for the sake of context, we present here a few additional central classes from the hierarchy.

An independence system $(\mathcal{N}, \mathcal{I})$ is a *p-system* if for every set $S \subseteq \mathcal{N}$, the ratio $|B_1|/|B_2|$ is upper bounded by p for every two bases B_1 and B_2 of $(S, 2^S \cap \mathcal{I})$. The class of p-systems is the most general class usually included in the hierarchy of independence systems. An important class included in it is the class of p-extendible systems. An independence system is a *p-extendible system* if for all $A \in \mathcal{I}$, extension $B \in \mathcal{I}$ of A and element $e \notin A$ such that $A \cup \{e\} \in \mathcal{I}$, there exists a set $Y \subseteq B \setminus A$ with $|Y| \leq p$ such that $B \setminus Y \cup \{e\} \in \mathcal{I}$. Intuitively, an independence system is p-extendible if adding an element e to an independent set B requires the removal of at most p other elements in order to keep the resulting set independent. Another important class of independence systems is the class of matroids. While the usual definition of matroids is based on linear algebra intuition, [Mestre, 2006] showed that this definition is equivalent to the definition of a 1-extendible system. An independence system $(\mathcal{N}, \mathcal{I})$ is a *p-matchoid* if there exist m matroids $(\mathcal{N}_1, \mathcal{I}_1), \ldots (\mathcal{N}_m, \mathcal{I}_m)$ such that $\mathcal{N} = \cup_{i=1}^m \mathcal{N}_i$, each element $e \in \mathcal{N}$ appears in
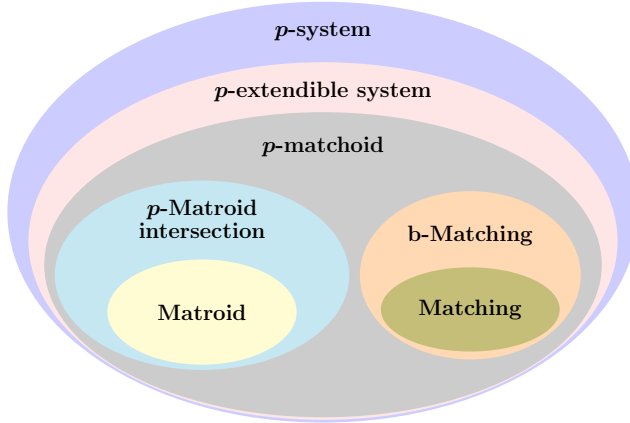
Figure 1: A visualization of the hierarchy of independence systems.

no more than p ground sets $\mathcal{N}_1, \ldots, \mathcal{N}_m$ and $\mathcal{I} = \{S \in \mathcal{N} \mid \forall_{i=1,\ldots,m} \, \mathcal{N}_i \cap S \in \mathcal{I}_i\}$. The hierarchy of the classes of independence systems mentioned above is presented below. We note that all the inclusions between these classes are known to be strict.

$$\text{matroid} \subset \text{intersection of } p \text{ matroids} \subset p\text{-matchoid} \subset p\text{-extendible} \subset p\text{-system} \ .$$

This hierarchy of independence systems is quite rich and expressive, containing many classic examples which are useful for modeling applications. The simplest example is the *k-cardinality constraint*, where $\mathcal{I} = \{S \mid S \subseteq \mathcal{N} \text{ and } |S| \leq k\}$, which is also referred to as the *uniform matroid*. The *partition matroid* is specified by a partition $P_1, P_2, \ldots P_\ell \subseteq \mathcal{N}$ ($\cup_{i=1}^\ell P_i = \mathcal{N}$ and $P_i \cap P_j = \varnothing$ for $i \neq j$) and integers $k_1, k_2 \ldots k_\ell$ such that $\mathcal{I} = \{S \mid |P_i \cap S| \leq k_i, i = 1, 2 \ldots \ell\}$. The *graphic matroid* is specified by an undirected graph $G = (V, E)$ where $\mathcal{N} = E$ and $\mathcal{I} = \{S \subseteq E \mid S \text{ does not contain a cycle}\}$. Matching constraints on subsets of edges of a graph—and more generally, b-matchings—form 2-matchoids. Moreover, a variety of scheduling constraints may be represented as p-extendible systems [Mestre, 2006]. Similarly to a partition matroid, the independence system given by subsets $P_1, P_2 \ldots P_\ell \subseteq \mathcal{N}$ (not necessarily a partition) and integers $k_1, k_2 \ldots k_\ell$ such that $\mathcal{I} = \{S \mid |P_i \cap S| \leq k_i \ i = 1, 2 \ldots \ell\}$ is the intersection of ℓ matroids (and thus, also an ℓ-matchoid). Finally, we remark that an example of an independence system which is p-extendible but not a p-matchoid is a knapsack constraint in which the sizes of all the elements are between 1 and p. A visualization of the hierarchy of independence systems is presented in Figure 1.

The problem we are interested in in this paper is the mathematical program

$$\max_{S \in \mathcal{I}} f(S) \ , \tag{2}$$

where f is non-negative and submodular on a ground set $\mathcal{N}$ and $(\mathcal{N}, \mathcal{I})$ is a p-extendible system. We denote an arbitrary set achieving this maximum by OPT. We say that a set $S \in \mathcal{I}$ is an *α-approximation* for some $\alpha \geq 1$ if

$$f(S) \geq \frac{1}{\alpha} \cdot f(\text{OPT}) \ .$$

We assume that both the objective function f and the independence system $\mathcal{I}$ are accessed by the algorithms through oracles; that is, given a set S there is a *value oracle* which returns the

value $f(S)$ and an *independence oracle* which returns whether or not $S \in \mathcal{I}$. Our goal is to design an algorithm which makes few queries to the value and independence oracles and produces a set that is an α-approximation for as small $\alpha \geq 1$ as possible.

In many practical applications, when the data is too large to be randomly accessed, offline algorithms are impractical. The *streaming model* of computation is an alternative computational paradigm for such settings. In this model, data points arrive in an arbitrary order and only a small amount of memory may be kept. More formally, let the elements of the ground set be arbitrarily ordered as $\mathcal{N} = \{u_1, \ldots, u_n\}$, and let $\mathcal{N}_t = \{u_1, \ldots, u_t\}$ be the first t elements in this ordering. An algorithm $\mathcal{A}$ for the model is presented with each element $u_1, u_2 \ldots u_n$ in a sequential manner. The algorithm $\mathcal{A}$ maintains a set M of the elements it currently keeps in its memory. Let us denote by M_t the set M immediately after the processing of u_t by the algorithm. We note that M_t must be a subset of $\mathcal{N}_t$, and also that it must be a subset of $M_{t-1} \cup \{u_t\}$ because once an element leaves M, it is forgotten and cannot be added to M later on. Naturally, $\mathcal{A}$ is allowed to query the value and independence oracle queries only with respect to subsets S of the current M, because M includes all the elements kept in $\mathcal{A}$'s memory.

To be an α-approximation algorithm, the algorithm $\mathcal{A}$ must be able—after viewing every element e_t—to produce a set $S_t \subseteq M_t$ that is independent ($S_t \in \mathcal{I}$) and is an α-approximation, i.e.,

$$f(S_t) \geq \frac{1}{\alpha} \cdot \max_{\substack{S \subseteq \mathcal{N}_t \\ S \in \mathcal{I}}} f(S) \ .$$

The performance of a streaming algorithm is judged based on its approximation ratio α, its update cost (which is the number of evaluation and independence oracle queries it makes after viewing each element), and the memory size $\max_t |M_t|$. In this paper, we consider streaming algorithms only for the case in which the constraint is defined by a p-matchoid, and we denote by m the number of matroids used to defined the p-matchoid. As is standard in the literature, we assume the streaming algorithm has access to an independence oracle for each of the m defining matroids when considering a p-matchoid.

In the context of streaming algorithms, given an element $u_i \in N$ and sets $S, T \subseteq \mathcal{N}$, we use the shorthands $f(u_i : S) = f(u_i \mid S \cap \{u_1, u_2, \ldots, u_{i-1}\})$ and $f(T : S) = \sum_{u \in T} f(u : S)$. Intuitively, $f(u : S)$ is the marginal contribution of u with respect to the part of S that arrived before u itself.

4 Main Results

In this section, we present two algorithms which use the subsampling technique for constrained submodular maximization. The subsampling technique is simple: the algorithm only considers a random subset $\mathcal{N}' \subseteq \mathcal{N}$ of the ground set, where each element appears independently with some probability q, which is determined solely by the complexity of the constraint set p. SAMPLE-GREEDY is proposed for the usual offline setting where random access to the data is assumed and SAMPLE-STREAMING is proposed for the streaming setting, where elements arrive one at a time and only a small dataset is maintained in memory. Our theoretical guarantees on the performance of these algorithms are given below.

Theorem 1. *When $q = (p+1)^{-1}$, SAMPLEGREEDY achieves a $\frac{(p+1)^2}{p}$-approximation ratio for the problem of maximizing a non-negative submodular function f subject to a p-extendible system. Moreover, this approximation ratio improves to $p + 1$ when f is monotone. When f is linear and $q = p^{-1}$, SAMPLEGREEDY achieves a further improved p-approximation ratio. In all cases, SAMPLEGREEDY uses in expectation $O(nk/p)$ calls to the evaluation and independence oracles.*

Theorem 2. *When $c = 1$ and $q = ((1 + c)p + 1)^{-1}$, SAMPLE-STREAMING achieves an approximation ratio of at most $(2p + 2\sqrt{p(p + 1)} + 1) = 4p + 2 - o(1)$ for maximizing a non-negative submodular function f subject to a p-matchoid system $(\mathcal{N}, \mathcal{I})$ in the streaming setting. When $c = \sqrt{1 + 1/p}$ and f is monotone, SAMPLE-STREAMING achieves an improved approximation ratio of at most $4p$. In both cases, SAMPLE-STREAMING requires $O(k)$ memory and $O(km/p)$ evaluation and independence oracle queries in expectation when processing each arriving element.*

We remark that the $(p+1)^2/p = p + 2 + o(1)$ approximation ratio of SAMPLEGREEDY is nearly tight, due to Theorem 4 of [Feldman et al., 2017], which shows that no randomized algorithm can achieve an approximation better than $p + 1/2$ for the problem using polynomially many queries to the evaluation and independence oracles. Although SAMPLE-STREAMING has an approximation guarantee which is worse than SAMPLEGREEDY and applies to a subclass of constraints, its approximation ratio is currently the best among known algorithms for the problem setting. Moreover, there is no known non-trivial streaming algorithm for maximizing even a monotone submodular function subject to a p-extendible constraint. The difference between the guarantees of SAMPLEGREEDY and SAMPLE-STREAMING is of course aligned with the intuitive expectation that the streaming setting should be more challenging than the offline setting.

In addition to its simplicity, one of the more attractive aspects of the subsampling technique is the way in which it can be analyzed to provide approximation guarantees for both the monotone and non-monotone settings in a unified manner. The main technical result which makes this possible is due to Buchbinder et al. [2014].

Lemma 3 (Lemma 2.2 of [Buchbinder et al., 2014]). *Let $g : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function, and let B be a random subset of $\mathcal{N}$ containing every element of $\mathcal{N}$ with probability at most q (not necessarily independently). Then, $\mathbb{E}[g(B)] \geq (1 - q) \cdot g(\varnothing)$.*

Suppose that S is the random set returned by an algorithm. When using the subsampling technique, this lemma provides a nontrivial lower bound on the term $\mathbb{E}[f(S \cup \text{OPT})]$. In particular, we can apply the lemma to the function defined by $g(T) = f(T \cup \text{OPT}) \ \forall \ T \subseteq \mathcal{N}$, whose submodularity and non-negativity is guaranteed by the same conditions on f. The subsampling technique implies that each element of $\mathcal{N}$ appears in S with probability at most q, and therefore,

$$\mathbb{E}[f(S \cup \text{OPT})] = \mathbb{E}[g(S)] \geq (1 - q) \cdot g(\varnothing) = (1 - q) \cdot f(\text{OPT}) \ .$$

This allows for a basically interchangeable step when lower bounding the term $\mathbb{E}[f(S \cup \text{OPT})]$ in the algorithmic analysis for monotone and non-monotone functions. If f is monotone, then we have $\mathbb{E}[f(S \cup \text{OPT})] \geq f(\text{OPT})$ by monotonicity; otherwise, we invoke the subsampling technique to obtain $\mathbb{E}[f(S \cup \text{OPT})] \geq (1 - q) \cdot f(\text{OPT})$ by Lemma 3. Note that Lemma 3 alone is not enough to guarantee any approximation factor. Rather, Lemma 3 shows that *subsampling* algorithms for monotone optimization can be converted into subsampling algorithms for non-monotone optimization with a controlled loss in the approximation ratio. The powerful—and arguably, shocking—result is that this controlled loss in approximation yields nearly optimal approximation ratios in the offline setting.

Our subsampling technique is different from that proposed by [Mirzasoleiman et al., 2015], which was developed for the problem of maximizing a monotone submodular function under a cardinality constraint. The subsampling technique of [Mirzasoleiman et al., 2015] works by sampling a new subset at each iteration and greedily choosing the element in the random subset with highest marginal gain. The analysis works by guaranteeing that if the sample size is large enough, then it is likely that an element with sufficiently large marginal gain is chosen by the algorithm. On the other hand, our subsampling technique in SAMPLEGREEDY is quite different in that it requires only one subsampling of the ground set at the beginning of the algorithm. Moreover, our

proposed subsampling technique admits guarantees in more general problem settings, including non-monotone objectives and p-extendible system constraints.

4.1 Offline Algorithm: SampleGreedy

In this section, we present SAMPLEGREEDY, a subsampling algorithm for the offline setting. The idea is simple: first independently sample elements to obtain a subsampled ground set, then run the vanilla greedy algorithm. We present SAMPLEGREEDY as Algorithm 1 here.

Algorithm 1: SAMPLEGREEDY $(q, f, \mathcal{N}, \mathcal{I})$

---

1   Let $\mathcal{N}' \leftarrow \varnothing$ and $S \leftarrow \varnothing$.
2   **for** *each $u \in \mathcal{N}$* **do**
3   **with** *probability q* **do**
4   Add u to $\mathcal{N}'$.
5   **end**
6   **while** *there exists $u \in \mathcal{N}'$ such that $S + u \in \mathcal{I}$ and $f(u \mid S) > 0$* **do**
7   Let $u \in \mathcal{N}'$ be the element of this kind maximizing $f(u \mid S)$.
8   Add u to S.
9   **end**
10   **return** S.

For analysis purposes, we introduce an auxiliary algorithm, Algorithm 2. While SAMPLE-GREEDY first independently samples elements from the ground set $\mathcal{N}$ and then runs a greedy maximization, Algorithm 2 runs a greedy maximization over the entire ground set and independently samples the greedily chosen element at each iteration. We will show that both algorithms produce the same distribution over their output sets $S \subseteq \mathcal{N}$ because the sampling of elements is independent from the greedy maximization. Thus, approximation guarantees obtained for Algorithm 2 also hold for SAMPLEGREEDY.

Algorithm 2: Equivalent-SAMPLEGREEDY $(q, f, \mathcal{N}, \mathcal{I})$

---

1   Let $\mathcal{N}' \leftarrow \mathcal{N}$, $S \leftarrow \varnothing$, $O \leftarrow \mathrm{OPT}$, and $O_u \leftarrow \varnothing$, $S_u \leftarrow \varnothing$ for each $u \in \mathcal{N}$.
2   **while** *there exists an element $u \in \mathcal{N}'$ such that $S + u \in \mathcal{I}$ and $f(u \mid S) > 0$* **do**
3   Let $u \in \mathcal{N}'$ be the element of this kind maximizing $f(u \mid S)$, and let $S_u \leftarrow S$.
4   **with** *probability q* **do**
5   Add u to S and O.
6   Let $O_u \subseteq O \setminus S$ be the smallest set such that $O \setminus O_u \in \mathcal{I}$.
7   **otherwise**
8   **if** $u \in O$ **then** let $O_u \leftarrow \{u\}$.
9   **else** let $O_u \leftarrow \varnothing$.
10   Remove the elements of O_u from O.
11   Remove u from $\mathcal{N}'$.
12   **end**
13   **return** S.

As in SAMPLEGREEDY, S is the current solution to which elements are incrementally added. Algorithm 2 also maintains several auxiliary sets for analysis purposes, such as O and the sets O_u, S_u for each element u in $\mathcal{N}$. We use these sets in the analysis of the algorithm below, however, they do not affect the output distribution (or other behavior) of Algorithm 2.

We now establish the equivalence between SAMPLEGREEDY and Algorithm 2. It is important, however, to note that for the equivalence to hold we must make some technical assumption about the tie-breaking rule used by the two algorithms. In the proof below we assume that this tie-breaking rule is based on an ordering of the ground set. In other words, suppose that the ground set is (arbitrarily) ordered $\mathcal{N} = \{u_1, u_2 \ldots u_n\}$ and that both algorithms break ties in favor of the element appearing earlier in this order.[2]

Lemma 4. *Let S^1 and S^2 be the random sets produced by* SAMPLEGREEDY *and Algorithm 2, respectively. These random sets have the same probability distribution.*

Proof. Proof Let $u_1^1, u_2^1 \ldots u_{k_1}^1$ be the random sequence of elements chosen by SAMPLEGREEDY, and let $u_1^2, u_2^2 \ldots u_{k_2}^2$ be the random sequence of elements chosen by Algorithm 2. We prove that these two random sequences have the same distribution, which implies the lemma since $S^1 = \{u_1^1, u_2^1 \ldots u_{k_1}^1\}$ and $S^2 = \{u_1^2, u_2^2 \ldots u_{k_2}^2\}$.

Observe that both SAMPLEGREEDY and Algorithm 2 require at most n random (biased) bits, as each element from the ground set is sampled at most once. This occurs at Line 3 in SAMPLEGREEDY and at Line 4 in Algorithm 2. For each element $u_i \in \mathcal{N}$, let $b_i \in \{0, 1\}$ be the corresponding random bit that takes the value 1 if u_i is accepted in the sampling step and 0 if u_i is rejected. Because the random sampling at Line 3 in SAMPLEGREEDY and at Line 4 in Algorithm 2 are independent of the algorithms' previous and current states, the bits $b_1, b_2, \ldots b_n \in \{0,1\}^n$ are independent. Thus, we may assume for the sake of the proof that these bits are chosen before the execution of the algorithms. Note also that, by definition of the sampling probabilities in the algorithms, each bit b_i takes the value 1 with probability q and 0 with probability $1 - q$.

We show below that conditioned on any fixed realization of the random bits $b_1, b_2 \ldots b_n$, the two sequences $u_1^1, u_2^1 \ldots u_{k_1}^1$ and $u_1^2, u_2^2 \ldots u_{k_2}^2$, produced by SAMPLEGREEDY and Algorithm 2 are the same. We do this by induction on the index of the element in the sequence. In other words, let us denote for every $i \in \{1, 2\}$ and $0 \leq j \leq k_i$,

$$S_j^i = \{u_1^i, u_2^i \ldots u_j^i\} \ .$$

Then, we prove below by induction that $j \leq k_1$ if and only if $j \leq k_2$, and that $S_j^1 = S_j^2$ whenever $j \leq k_1$. Before starting the proof by induction, however, let us note that it implies the lemma by the law of total probability, i.e., summing over all fixed realizations of the bits $b_1, b_2, \ldots b_n$.

Clearly, $0 \leq k_1, k_2$ and $S_0^1 = S_0^2 = \varnothing$, which establishes the base of the induction. Assume now that the induction hypothesis holds for some $j - 1 \geq 0$, and let us prove it for j. If $j - 1 > k_1$, then the induction hypothesis implies $j > j - 1 > \max\{k_1, k_2\}$, and there is nothing left to prove. Thus, let us assume $j - 1 \leq k_1$, which by the induction hypothesis implies $j - 1 \leq k_2$ and $S_{j-1}^1 = S_{j-1}^2$. We denote the common value of the last two sets by A, i.e., $A = S_{j-1}^1 = S_{j-1}^2$. We note that SAMPLEGREEDY chooses its next element from the set of elements obeying $u \in \mathcal{N}'$, $A + u \in \mathcal{I}$ and $f(u \mid A)$ is positive and maximal. Since $u \in \mathcal{N}'$ if and only if $b_u = 1$, this implies that SAMPLEGREEDY chooses u_j to be the earliest element in the tie-breaking order from the set

$$\underset{\substack{u_i \in \mathcal{N} \\ b_i = 1 \\ A + u_i \in \mathcal{I}}}{\arg\max} \{f(u \mid A) \mid f(u \mid A) > 0\} \ , \tag{3}$$

whenever this set is non-empty. Otherwise, if the set is empty, then SAMPLEGREEDY terminates. Now, consider the element which is chosen next by Algorithm 2. This element is chosen from the

---

[2]There are of course other natural ways in which the tie-breaking rule can be defined, and the proof can be made to work with many of them. For example, the proof goes through if the tie-breaking rule picks at every step a uniformly random element out of all the elements that can be picked at this point and maximize the marginal gain.

set of elements obeying $u \in \mathcal{N}'$, $A + u \in \mathcal{I}$ and $f(u \mid A)$ is positive and maximal. However, if $b_u = 0$, then the element is rejected and the algorithm continues to the next element in the set (according to the tie-breaking order). Thus, Algorithm 2 also chooses the earliest element from the set (3) if this set is non-empty; and terminates if the set is empty. Therefore, if the set (3) is empty, then $k_1 = k_2 = j - 1 < j$, and if it is not empty, then $j \leq \min\{k_1, k_2\}$ and $S_j^1 = S_j^2$. In either case the induction step holds, which completes the proof by induction. $\qquad\square$

Now we return to the task of analyzing Algorithm 2. Let us explain the intuition behind the auxiliary sets O, O_u and S_u appearing in this algorithm. The set O begins as an optimal solution and is updated throughout the algorithm to maintain independence. We say that an element $u \in \mathcal{N}$ is *considered* by Algorithm 2 if it is chosen in Line 3 at some iteration. An element $u \in \mathcal{N}$ is considered at most once, and perhaps not at all. If u is considered, then S_u is the solution at the iteration in which this happens and O_u is the subset of O which must be removed at this iteration to maintain independence and a few other properties; otherwise, if u is not considered, then S_u and O_u are empty. More formally, Lemma 5 gives several key properties of these auxiliary sets.

Lemma 5. *The following three properties hold throughout Algorithm 2.*

(P1) O is an independent set.

(P2) Every element of S is an element of O.

(P3) Every element of $O \setminus S$ is an element not yet considered by Algorithm 2.

Proof. Proof It is clear that all three properties hold at the beginning of the algorithm when $O = \mathrm{OPT}$ and $S = \varnothing$. Let us now show that these properties are maintained throughout Algorithm 2 by induction over the iterations of the algorithm. Suppose that u is the element being considered at some iteration, S and O are the sets at the beginning of this iteration (satisfying properties *(P1)*, *(P2)*, and *(P3)* by the inductive hypothesis) and S' and O' are these sets at the end of the iteration.

Case 1 Suppose that u is chosen to be added to the current solution. By *(P1)* and *(P2)*, O is an extension of S, and by Line 3, $S \cup \{u\} \in \mathcal{I}$. Thus, the algorithm is able to find a set $O_u \subseteq (O \cup \{u\}) \setminus (S \cup \{u\})$ such that $O' = (O \cup \{u\}) \setminus O_u \in \mathcal{I}$ ($O_u = O \setminus (S \cup \{u\})$ is one possible option). Thus, O' remains independent, and *(P1)* is maintained. We have that $S \subseteq O$ by *(P2)*. The element u is added to both S and O, and the only elements which are removed from O are not in S. Thus, $S' \subseteq O'$ and so *(P2)* is maintained at well. By *(P3)*, all elements in $O \setminus S$ had not yet been considered by Algorithm 2 at the beginning of the iteration. The only element which is considered in this iteration is u, and it is in both O' and S', so it is not in $O' \setminus S'$. Hence, the elements in $O' \setminus S'$ have still not been considered by Algorithm 2 at the end of iteration, and *(P3)* is maintained.

Case 2 Consider now the case that u is not added to the current solution. In this case, $O' \subseteq O$, and therefore, it remains independent and *(P1)* is maintained. Because $S \subseteq O$ by *(P2)*, $u \notin S' = S$, and the only element which is possibly removed from O is u, we have that all elements in S' belong to O', and thus, *(P2)* is maintained. Finally, by *(P3)*, none of the elements in $O \setminus S$ were considered prior to this iteration. By Line 8, u does not appear in $O' \setminus S'$. Since u is the only element considered during the current iteration and $O' \setminus S' = (O \setminus S) \setminus \{u\}$, property *(P3)* is maintained. $\qquad\square$

Throughout the remainder of this section, every expression involving S or O is assumed to refer to the final values of these sets. The following lemma provides a deterministic lower bound on $f(S)$. Intuitively, this lemma follows from the observation that, when an element u is considered by Algorithm 2, its marginal contribution is at least as large as the marginal contribution of any element of $O \setminus S$.

Lemma 6. $f(S) \geq f(S \cup OPT) - \sum\limits_{u \in \mathcal{N}} |O_u \setminus S| \cdot f(u \mid S_u)$.

Proof. Proof We first show that $f(S) \geq f(O)$, then we lower bound $f(O)$ to complete the proof. By *(P1)* and *(P2)* of Lemma 5, we have $O \in \mathcal{I}$ and $S \subseteq O$, and thus, $S + v \in \mathcal{I}$ for all $v \in O \setminus S$ because $(\mathcal{N}, \mathcal{I})$ is an independence system. Consequently, the termination condition of Algorithm 2 guarantees that $\Delta f(v \mid S) \leq 0$ for all $v \in O \setminus S$. To use these observations, let us denote the elements of $O \setminus S$ by $v_1, v_2, \ldots, v_{|O \setminus S|}$ in an arbitrary order. Then

$$f(O) = f(S) + \sum_{i=1}^{|O \setminus S|} f\left(v_i \mid S \cup \{v_1, \ldots, v_{i-1}\}\right) \leq f(S) + \sum_{i=1}^{|O \setminus S|} f\left(v_i \mid S\right) \leq f(S) \ ,$$

where the first inequality follows by the submodularity of f, and the second inequality follows from the termination condition.

It remains to prove the lower bound on $f(O)$. By definition, O is the set obtained from OPT after the elements of $\cup_{u \in \mathcal{N}} O_u$ are removed and the elements of S are added. Additionally, an element that is removed from O is never added to O again, unless it becomes a part of S. This implies that the sets $\{O_u \setminus S\}_{u \in \mathcal{N}}$ are disjoint, and that O can also be written as

$$O = (S \cup \text{OPT}) \setminus \cup_{u \in \mathcal{N}} (O_u \setminus S) \ . \tag{4}$$

Denoting the elements of $\mathcal{N}$ by $u_1, u_2, \ldots, u_n$ in an arbitrary order, and using the above, we get

$$\begin{aligned}
f(O) &= f(S \cup \text{OPT}) - \sum_{i=1}^{n} f\left(O_{u_i} \setminus S \mid (S \cup \text{OPT}) \setminus \cup_{1 \leq j \leq i}(O_{u_j} \setminus S)\right) && \text{(Equality (4))} \\
&\geq f(S \cup \text{OPT}) - \sum_{i=1}^{n} f(O_{u_i} \setminus S \mid S_{u_i}) \\
&\geq f(S \cup \text{OPT}) - \sum_{i=1}^{n} \sum_{v \in O_{u_i} \setminus S} f(v \mid S_{u_i}) \\
&= f(S \cup \text{OPT}) - \sum_{u \in \mathcal{N}} \sum_{v \in O_u \setminus S} f(v \mid S_u) \ ,
\end{aligned}$$

where the first inequality follows from the submodularity of f because $S_{u_i} \subseteq S \subseteq (S \cup \text{OPT}) \setminus \cup_{u \in \mathcal{N}} (O_u \setminus S)$, and the second inequality follows from the submodularity of f as well.

To complete the proof of the lemma, we need one more observation. Consider an element u for which O_u is not empty. Since O_u is not empty, we know that u was considered by the algorithm at some iteration. Moreover, every element of O_u was also a possible candidate for consideration at this iteration, and thus, it must be the case that u was selected for consideration because its marginal contribution with respect to S_u is at least as large as the marginal contribution of every element of O_u. Plugging this observation into the last inequality, we get the following desired

lower bound on $f(O)$.

$$f(O) \geq f(S \cup \text{OPT}) - \sum_{u \in \mathcal{N}} \sum_{v \in O_u \setminus S} f(v \mid S_u)$$

$$\geq f(S \cup \text{OPT}) - \sum_{u \in \mathcal{N}} \sum_{v \in O_u \setminus S} f(u \mid S_u)$$

$$= f(S \cup \text{OPT}) - \sum_{u \in \mathcal{N}} |O_u \setminus S| \cdot f(u \mid S_u) \; . \qquad \square$$

While the previous lemma was true deterministically, the next two lemmata are statements about expected values. At this point, it is convenient to define some random variables. For every element $u \in \mathcal{N}$, let X_u be an indicator for the event that u is considered by Algorithm 2 in one of its iterations.

Lemma 7. *Suppose that the sampling probability is $q = \frac{1}{p+1}$. Then, for every element $u \in \mathcal{N}$,*

$$\mathbb{E}[|O_u \setminus S| \cdot f(u \mid S_u)] \leq \frac{p}{p+1} \cdot \mathbb{E}[X_u f(u \mid S_u)] \; . \tag{5}$$

Proof. Proof Let $\mathcal{E}_u$ be an arbitrary event specifying all random decisions made by Algorithm 2 up until the iteration in which it considers u if u is considered, or all random decisions made by Algorithm 2 throughout its execution if it never considers u. By the law of total probability, since these events are disjoint, it is enough to prove Inequality (5) conditioned on every such event $\mathcal{E}_u$. If $\mathcal{E}_u$ implies that u is not considered, then both $|O_u|$ and X_u are 0 conditioned on $\mathcal{E}_u$, and thus, the inequality holds as an equality. Thus, we may assume in the rest of the proof that $\mathcal{E}_u$ implies that u is considered by Algorithm 2. Notice that conditioned on $\mathcal{E}_u$ the set S_u is deterministic and X_u takes the value 1. Denoting the deterministic value of S_u conditioned on $\mathcal{E}_u$ by S'_u, Inequality (5) reduces to

$$\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] \cdot f(u \mid S'_u) \leq \frac{p}{p+1} \cdot f(u \mid S'_u) \; .$$

Since u is being considered, it must hold that $f(u \mid S'_u) > 0$, and thus, it suffices to show that $\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] \leq \frac{p}{p+1}$. There are now two cases to consider.

Case 1 If $\mathcal{E}_u$ implies that $u \in O$ at the beginning of the iteration in which Algorithm 2 considers u, then $O_u = \varnothing$ if u is added to S and $O_u = \{u\}$ if u is not added to S. As u is added to S with probability $\frac{1}{p+1}$, this gives

$$\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] = \frac{1}{p+1} \cdot |\varnothing| + \left(1 - \frac{1}{p+1}\right) \cdot |\{u\}| = \frac{p}{p+1} \; ,$$

and we are done.

Case 2 Consider now the case that $\mathcal{E}_u$ implies that $u \notin O$ at the beginning of the iteration in which Algorithm 2 considers u. Because u is being considered, $S \cup \{u\}$ is independent and by *(P2)* and *(P1)* of Lemma 5, O is an extension of S. This implies that O_u has size at most p because $(\mathcal{N}, \mathcal{I})$ is p-extendible. As u is added to S with probability $\frac{1}{p+1}$, we get in this case

$$\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] \leq \frac{1}{p+1} \cdot p + \left(1 - \frac{1}{p+1}\right) \cdot |\varnothing| = \frac{p}{p+1} \; . \qquad \square$$

The next lemma relates the expected marginal gains of considered elements in individual iterations to the final expected value of $f(S)$ produced by the algorithm.

Lemma 8. $q \cdot \sum_{u \in \mathcal{N}} \mathbb{E}[X_u f(u \mid S_u)] \leq \mathbb{E}[f(S)]$.

Proof. Proof For each $u \in \mathcal{N}$, let G_u be a random variable whose value is equal to the increase in the value of S when u is added to S by Algorithm 2. If u is never added to S by Algorithm 2, then the value of G_u is simply 0. Clearly,

$$f(S) = f(\varnothing) + \sum_{u \in \mathcal{N}} G_u \geq \sum_{u \in \mathcal{N}} G_u \ ,$$

where $f(\varnothing) \geq 0$ follows from non-negativity of f. By the linearity of expectation, it suffices to show that

$$\mathbb{E}[G_u] = q \cdot \mathbb{E}[X_u f(u \mid S_u)] \ . \tag{6}$$

As in the proof of Lemma 7, let $\mathcal{E}_u$ be an arbitrary event specifying all random decisions made by Algorithm 2 up until the iteration in which it considers u if u is considered, or all random decisions made by Algorithm 2 throughout its execution if u is never considered. By the law of total probability, since these events are disjoint, it is enough to prove that Equality (6) holds when conditioned on every such event $\mathcal{E}_u$. If $\mathcal{E}_u$ is an event that implies that Algorithm 2 does not consider u, then, by conditioning on $\mathcal{E}_u$, we obtain

$$\mathbb{E}[G_u \mid \mathcal{E}_u] = 0 = q \cdot \mathbb{E}[0 \cdot f(u \mid S_u) \mid \mathcal{E}_u] = q \cdot \mathbb{E}[X_u f(u \mid S_u) \mid \mathcal{E}_u] \ .$$

On the other hand, if $\mathcal{E}_u$ implies that Algorithm 2 does consider u, then we observe that S_u is a deterministic set given $\mathcal{E}_u$. Denoting this set by S'_u, we obtain

$$\mathbb{E}[G_u \mid \mathcal{E}_u] = \Pr[u \in S \mid \mathcal{E}_u] \cdot f(u \mid S'_u) = q \cdot f(u \mid S'_u) = q \cdot \mathbb{E}[X_u f(u \mid S_u) \mid \mathcal{E}_u] \ ,$$

where the second equality holds since an element considered by Algorithm 2 is added to S with probability q. □

With these lemmata, we are now ready to prove Theorem 1 in the case of submodular (not necessarily linear) objectives.

4.1.1 Proof of Theorem 1, Submodular Objectives.

We prove the first part of Theorem 1 concerning submodular functions in this section. The improved approximation guarantees for linear functions requires a few tighter lemmas, and so we prove the case of linear functions in the next section (Section 4.1.2).

Theorem 1. As discussed earlier, Algorithms 1 and 2 have identical output distributions, and so it suffices to show that Algorithm 2 achieves the desired approximation ratios. Note that $q = \frac{1}{p+1}$, and therefore,

$$\mathbb{E}[f(S)] \geq \mathbb{E}[f(S \cup \mathrm{OPT})] - \sum_{u \in \mathcal{N}} \mathbb{E}[|O_u \setminus S| \cdot f(u \mid S_u)] \qquad \text{(Lemma 6)}$$

$$\geq \mathbb{E}[f(S \cup \mathrm{OPT})] - \frac{p}{p+1} \sum_{u \in \mathcal{N}} \mathbb{E}[X_u \cdot f(u \mid S_u)] \qquad \text{(Lemma 7)}$$

$$\geq \mathbb{E}[f(S \cup \mathrm{OPT})] - p \cdot \mathbb{E}[f(S)] \ . \qquad \text{(Lemma 8)}$$

14

If f is monotone, then by monotonicity we have that $\mathbb{E}[f(S \cup \text{OPT})] \geq f(\text{OPT})$. Substituting this in the expression above yields

$$\mathbb{E}[f(S)] \geq f(\text{OPT}) - p \cdot \mathbb{E}[f(S)]$$

and rearranging this expression yields the desired approximation ratio of $p + 1$. Suppose now that f is non-monotone. Note that each element appears in S with probability at most $q = \frac{1}{p+1}$, and hence, by Lemma 3, we have that $\mathbb{E}[f(S \cup \text{OPT})] \geq \left(1 - \frac{1}{p+1}\right) f(\text{OPT})$. Substituting this into the inequalities above yields

$$\mathbb{E}[f(S)] \geq \left(1 - \frac{1}{p+1}\right) f(\text{OPT}) - p \cdot \mathbb{E}[f(S)] \ ,$$

and rearranging this expression yields the desired approximation ratio of $(p+1)^2/p$.

It remains to bound the number of oracle calls required by SampleGreedy. Because $\mathbb{E}[|\mathcal{N}'|] = n \cdot q$, iterating over each $u \in \mathcal{N}'$ and testing $S + u \in \mathcal{I}$ and $f(u \mid S) > 0$ requires $O(nq)$ calls to the evaluation and independence oracle. Moreover, because $|S|$ increases at each iteration, the while loop (Line 6 in Algorithm 1) is repeated at most k times. Using that $q = O(p^{-1})$, we have shown that Algorithm 1 requires $O(knq) = O(nk/p)$ calls to evaluation and independence oracles in expectation.

4.1.2 Proof of Theorem 1, Linear Objectives.

The method for proving the improved approximation guarantees for linear objectives uses essentially the same ideas as in the general submodular setting. However, further care is required to obtain the p-approximation guarantee. In this section, we prove two lemmata which are analogous to Lemmata 6 and 7, but tighter in the case of linear functions.

We begin with the following lemma, which corresponds to Lemma 6. For every $u \in \mathcal{N}$, let Y_u be a random variable which takes the value 1 if $u \in S$ and, in addition, u does not belong to O at the beginning of the iteration in which u is considered. In every other case the value of Y_u is 0.

Lemma 9. $f(S) \geq f(OPT) - \sum\limits_{u \in \mathcal{N}} [|O_u| - Y_u] f(u)$.

Proof. Proof The proof of Lemma 6 begins by showing that $f(S) \geq f(O)$. This part of the proof is of course still true. Thus, we only need to show that

$$f(O) \geq f(\text{OPT}) - \sum_{u \in \mathcal{N}} [|O_u| - Y_u] f(u) \ .$$

Recall that O begins as equal to OPT. Thus, to prove the last inequality it is enough to show that the second term on its right hand side is an upper bound on the decrease in the value of O over time. In the rest of the proof we do this by showing that $[|O_u| - Y_u] f(u)$ is an upper bound on the decrease in the value of O in the iteration in which u is considered, and is equal to 0 when u is not considered at all.

Let us first consider the case that u is not considered at all. In this case, by definition, $O_u = \varnothing$ and $Y_u = 0$, which imply together $[|O_u| - Y_u] f(u) = 0 \cdot f(u) = 0$. Consider now the case that u is considered by Algorithm 2. In this case, O is changed during the iteration in which u is considered in two ways. First, the elements of O_u are removed from O, and second, u is added

15

to O if it is added to S and it does not already belong to O. Thus, the decrease in the value of O during this iteration can be written as

$$\sum_{v \in O_u} f(v) - Y_u \cdot f(u) \ .$$

To see why this expression is upper bounded by $[|O_u| - Y_u]f(u)$, we recall that in the proof of Lemma 6 we showed that $f(v \mid S_u) \leq f(u \mid S_u)$ for every $v \in O_u$, which implies, since f is linear, that $f(v) \leq f(u)$ for every such element v. □

We need one more lemma which corresponds to Lemma 7.

Lemma 10. *Suppose that the sampling probability is $q = \frac{1}{p}$. Then, for every element $u \in \mathcal{N}$,*

$$\mathbb{E}[|O_u| - Y_u] \leq \frac{p-1}{p} \cdot \mathbb{E}[X_u] \ . \tag{7}$$

Proof. Proof As in the proof of Lemma 7, let $\mathcal{E}_u$ be an arbitrary event specifying all random decisions made by Algorithm 2 up until the iteration in which it considers u if u is considered, or all random decisions made by Algorithm 2 throughout its execution if it never considers u. By the law of total probability, since these events are disjoint, it is enough to prove Inequality (7) conditioned on every such event $\mathcal{E}_u$. If $\mathcal{E}_u$ implies that u is not considered, then $|O_u|$, X_u and Y_u are all 0 conditioned on $\mathcal{E}_u$, and thus, the inequality holds as an equality. Thus, we may assume in the rest of the proof that $\mathcal{E}_u$ implies that u is considered by Algorithm 2. Notice that, conditioned on $\mathcal{E}_u$, X_u takes the value 1. Hence, Inequality (7) reduces to

$$\mathbb{E}[|O_u| - Y_u \mid \mathcal{E}_u] \leq \frac{p-1}{p} \ .$$

There are now two cases to consider.

Case 1 The first case is that $\mathcal{E}_u$ implies that $u \in O$ at the beginning of the iteration in which Algorithm 2 considers u. In this case $Y_u = 0$, and in addition, O_u is empty if u is added to S, and is $\{u\}$ if u is not added to S. As u is added to S with probability $\frac{1}{p}$, this gives

$$\mathbb{E}[|O_u| - Y_u \mid \mathcal{E}_u] \leq \frac{1}{p} \cdot |\varnothing| + \left(1 - \frac{1}{p}\right) \cdot |\{u\}| = \frac{p-1}{p} \ ,$$

and we are done.

Case 2 Consider now the case that $\mathcal{E}_u$ implies that $u \notin O$ at the beginning of the iteration in which Algorithm 2 considers u. In this case, if u is not added to S, then we get $Y_u = 0$ and $O_u = \varnothing$. In contrast, if u is added to S, then $Y_u = 1$ by definition and $|O_u| \leq p$ as in the proof of Lemma 7. As u is added to S with probability $\frac{1}{p}$, we get in this case

$$\mathbb{E}[|O_u| - Y_u| \mid \mathcal{E}_u] \leq \frac{1}{p} \cdot (p-1) + \left(1 - \frac{1}{p}\right) \cdot |\varnothing| = \frac{p-1}{p} \ . \qquad \square$$

We are now ready to prove the guarantee of Theorem 1 for linear objectives.

Proof. Proof of Theorem 1 for linear objectives. We prove here that the approximation ratio guaranteed by Theorem 1 for linear objectives is obtained by Algorithm 1 for $q = 1/p$. As discussed earlier, Algorithms 1 and 2 have identical output distributions, and so it suffices to show that Algorithm 2 achieves this approximation ratio. Since we assume $q = \frac{1}{p}$,

$$\mathbb{E}[f(S)] \geq f(\text{OPT}) - \sum_{u \in \mathcal{N}} \mathbb{E}[|O_u| - Y_u] f(u) \qquad \text{(Lemma 9)}$$

$$\geq f(\text{OPT}) - \frac{p-1}{p} \sum_{u \in \mathcal{N}} \mathbb{E}[X_u] f(u) \qquad \text{(Lemma 10)}$$

$$\geq f(\text{OPT}) - (p-1)\mathbb{E}[f(S)] \ . \qquad \text{(Lemma 8)}$$

Rearranging the above inequality completes the proof, as the oracle complexity is unchanged. $\square$

4.2   Streaming Algorithm

In this section, we present SAMPLE-STREAMING, a subsampling algorithm for the streaming setting. SAMPLE-STREAMING has two parameters: a sampling probability $q \in (0, 1]$ and an acceptance parameter $c > 0$. At a given iteration $i = 1, \ldots, n$, the arriving element u_i is considered for exchange with probability q, and rejected without being considered for an exchange with probability $1 - q$. This step acts as an independent subsampling of elements in the stream, in an analogous manner to the subsampling in SAMPLEGREEDY. If the element u_i is considered for exchange, a subroutine EXCHANGE-CANDIDATE[3] produces a set $U_i \subseteq S_i$ of low marginal contribution such that $(S \setminus U_i) \cup \{u_i\}$ is independent. If the marginal contribution of adding u_i to the current solution is large enough compared to the value of the elements of U, then u is added to the solution and the elements of U are removed. SAMPLE-STREAMING and the subroutine EXCHANGE-CANDIDATE are presented here as Algorithms 3 and 4, respectively.

Algorithm 3: SAMPLE-STREAMING $(q, f, \mathcal{I}_1, \ldots, \mathcal{I}_m)$

---

1   Let $S_0 \leftarrow \varnothing$.
2   **for** *every arriving element u_i* **do**
3 $\quad$ Let $S_i \leftarrow S_{i-1}$.
4 $\quad$ **with** *probability q* **do**
5 $\quad\quad$ Let $U_i \leftarrow$ EXCHANGE-CANDIDATE(S_{i-1}, u_i).
6 $\quad\quad$ **if** $f(u_i \mid S_{i-1}) \geq (1 + c) \cdot f(U_i : S_{i-1})$ **then** let $S_i \leftarrow (S_{i-1} \setminus U_i) \cup \{u_i\}$.
7 $\quad$ **end**
8   **end**
9   **return** S_n.

SAMPLE-STREAMING adds an element u to the current solution if two conditions are satisfied: first, the element is randomly sampled from the stream in Line 4 and second, the element has sufficient marginal contribution. These two conditions are checked in this order because it is more computationally efficient as it avoids unnecessary oracle calls. However, the order that these conditions are checked may be swapped without affecting the distribution of outcomes of the algorithm. In fact, it is easier to analyze the algorithm when these conditions are reversed. It is also convenient to assume that elements which have sufficiently large marginal contributions

---

[3]The subroutine EXCHANGE-CANDIDATE has appeared in a previous work [Chekuri et al., 2015] as a method for exchanging in a p-matchoid.

Algorithm 4: EXCHANGE-CANDIDATE (S, u)
---
1   Let $U \leftarrow \varnothing$.
2   **for** $\ell = 1$ **to** m **do**
3   **if** $(S + u) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell$ **then**
4   Let $X_\ell \leftarrow \{x \in S \mid ((S - x + u) \cap \mathcal{N}_\ell) \in \mathcal{I}_\ell\}$.
5   Let $x_\ell \leftarrow \arg\min_{x \in X_\ell} f(x : S)$.
6   Add x_ℓ to U.
7   **end**
8   **end**
9   **return** U.
---

but are not subsampled from the stream are put into a set R. We present Algorithm 5 with these changes for the purpose of analysis.

Algorithm 5: Equivalent-SAMPLE-STREAMING $(q, f, \mathcal{I}_1, \ldots, \mathcal{I}_m)$
---
1   Let $S_0 \leftarrow \varnothing$ and $R \leftarrow \varnothing$.
2   **for** *every arriving element* u_i **do**
3   Let $S_i \leftarrow S_{i-1}$.
4   Let $U_i \leftarrow$ EXCHANGE-CANDIDATE(S_{i-1}, u_i).
5   **if** $f(u_i \mid S_{i-1}) \geq (1 + c) \cdot f(U_i : S_{i-1})$ **then**
6   **with** *probability* q **do** Let $S_i \leftarrow (S_{i-1} \setminus U_i) \cup \{u_i\}$.
7   **otherwise** Add u_i to R.
8   **end**
9   **end**
10   **return** S_n.

We now formally show that SAMPLE-STREAMING and Algorithm 5 have the same distribution of returned sets. The proof of equivalence in the streaming setting is simpler than in the offline setting. This is due to the fact that an ordering of the ground set does not need to be chosen by the algorithm—it is already determined by the order of the stream. We assume that the procedure EXCHANGE-CANDIDATE uses a tie-breaking rule at Line 5 of Algorithm 4 which depends only on the set S and the element u. For simplicity, we also assume that this rule is deterministic, so that the set U_i is deterministic conditioned on the current solution S_{i-1} and the new element u_i which are given as input to EXCHANGE-CANDIDATE. This is useful because it implies that the event $f(u_i \mid S_{i-1}) \geq (1 + c) \cdot f(U_i : S_{i-1})$ is also deterministic conditioned on S_{i-1} and u_i. We note, however, that the proof can be easily made to work also with a randomized tie-breaking rule in the procedure EXCHANGE-CANDIDATE, as long as this rule only depends on the set S and the element u.

Lemma 11. *Let S_i^1 and S_i^2 be the random solution sets maintained by* SAMPLE-STREAMING *and Equivalent-*SAMPLE-STREAMING *at iterations $i = 0, 1, \ldots, n$, respectively. At each iteration $i = 0, 1, \ldots, n$, the random sets S_i^1 and S_i^2 have the same distribution.*

Proof. Proof We prove the lemma by induction on the iteration i. For $i = 0$ the lemma is trivial since both S_0^1 and S_0^2 are initialized to be empty. Suppose now that S_k^1 and S_k^2 have the same distributions for all iterations $k = 0, 1, \ldots, i - 1$ and let us prove that S_i^1 and S_i^2 also share the same distribution. In fact, we show the even stronger property that for every set $A \subseteq \mathcal{N}$

such that $\Pr[S_{i-1}^1 = A] = \Pr[S_{i-1}^2 = A] > 0$, the sets S_i^1 and S_i^2 have the same distributions conditioned on the events $S_{i-1}^1 = A$ and $S_{i-1}^2 = A$, respectively.

Let u_i be the ith element in the stream, encountered by both algorithms. Recall that since we condition on $S_{i-1}^1 = A$ or $S_{i-1}^2 = A$, the same set U_i is chosen by both algorithms (in the case of SAMPLE-STREAMING, we mean here the set that is chosen if the algorithm decides to pick an element in this iteration). Suppose now that A is such that $f(u_i \mid S_{i-1}) \geq (1 + c) \cdot f(U_i : S_{i-1})$. Then in SAMPLE-STREAMING, the probability of updating $S_i^1 \leftarrow (A \setminus U_i) \cup \{u_i\}$ is q and the probability of keeping $S_i^1 \leftarrow S_{i-1}^1$ is $1-q$. This is also true for Algorithm 5; that is, the probability of updating $S_i^2 \leftarrow (A \setminus U_i) \cup \{u_i\}$ is q and the probability of keeping $S_i^2 \leftarrow S_{i-1}^2$ is $1 - q$. This is due to the fact that sampling and exchange procedures are independent in both algorithms. If A is such that $f(u_i \mid S_{i-1}) < (1 + c) \cdot f(U_i : S_{i-1})$, then both algorithms keep the current solution (that is, $S_i \leftarrow S_{i-1}$) with probability 1. Thus, we have shown that for every $A, B \subseteq \mathcal{N}$, if $\Pr[S_{i-1}^1 = A] > 0$, then

$$\Pr\left[S_i^1 = B \,\middle|\, S_{i-1}^1 = A\right] = \Pr\left[S_i^2 = B \,\middle|\, S_{i-1}^2 = A\right] \ .$$

The lemma now follows by the law of total probability and the inductive hypothesis. $\qquad\square$

Now that the equivalence of SAMPLE-STREAMING and Algorithm 5 has been established, we are guaranteed that any approximation guarantee for Algorithm 5 also holds for SAMPLE-STREAMING. Accordingly, in the remainder of the section, we analyze Algorithm 5. The following technical lemma shows that, for every two sets A and B, the sum of the marginal contributions of the elements of B (as they arrive) to the already arrived elements of A is larger than the total marginal contribution of B to A.

Observation 1. *For every two sets $A, B \subseteq \mathcal{N}$, $f(B \mid A \setminus B) \leq f(B : A)$.*

Proof. Proof Let us denote the elements of B by $u_{i_1}, u_{i_2}, \ldots, u_{i_{|B|}}$, where $i_1 < i_2 < \cdots < i_{|B|}$. Then,

$$
\begin{aligned}
f(B \mid A \setminus B) &= \sum_{j=1}^{|B|} f(u_{i_j} \mid (A \cup B) \setminus \{u_{i_j}, u_{i_{j+1}} \ldots, u_{i_{|B|}}\}) \\
&\leq \sum_{j=1}^{|B|} f(u_{i_j} \mid A \setminus \{u_{i_j}, u_{i_{j+1}} \ldots, u_n\}) \\
&= \sum_{j=1}^{|B} f(u_{i_j} \mid A \cap \{u_1, u_2, \ldots, u_{i_j-1}\}) \\
&= \sum_{j=1}^{|B|} f(u_{i_j} : A) \\
&= f(B : A) \ ,
\end{aligned}
$$

where the inequality follows from the submodularity of f. $\qquad\square$

Let us denote from this point on by A the set of elements that ever appeared in the solution maintained by Algorithm 5—formally, $A = \bigcup_{i=1}^n S_i$. The following lemma and corollary show that the elements of $A \setminus S_n$ cannot contribute much to the output solution S_n of Algorithm 5, and thus, their absence from S_n does not make S_n much less valuable than A.

Lemma 12. $f(A \setminus S_n : S_n) \leq \frac{f(S_n)}{c}$.

Proof. Proof Fix an element $u_i \in A$, then

$$
\begin{aligned}
f(S_i) - f(S_{i-1}) &= f(S_{i-1} \setminus U_i + u_i) - f(S_{i-1}) \\
&= f(u_i \mid S_{i-1} \setminus U_i) - f(U_i \mid S_{i-1} \setminus U_i) \\
&\geq f(u_i \mid S_{i-1}) - f(U_i : S_{i-1}) \\
&\geq c \cdot f(U_i : S_{i-1}) \ ,
\end{aligned}
$$

where the first inequality follows from the submodularity of f and Observation 1, and the second inequality holds since the fact that Algorithm 5 accepted u_i into its solution implies $f(u_i \mid S_{i-1}) \geq (1+c) \cdot f(U_i : S_{i-1})$.

Because every element of $A \setminus S_n$ has been removed exactly once from the solution of Algorithm 5, the sets U_i such that $u_i \in A$ form a disjoint partition of $A \setminus S_n$. Thus,

$$
f(A \setminus S_n : S_n) = \sum_{u_i \in A} f(U_i : S_n) \leq \sum_{u_i \in A} \frac{f(S_i) - f(S_{i-1})}{c} = \frac{f(S_n) - f(\varnothing)}{c} \leq \frac{f(S_n)}{c} \ ,
$$

where the first inequality follows from the inequalities above, the second equality holds since $S_i = S_{i-1}$ whenever $u_i \notin A$ and the second inequality follows from the non-negativity of f. $\qquad\square$

Corollary 13. $f(A) \leq \frac{c+1}{c} \cdot f(S_n)$.

Proof. Proof Observe that

$$
f(A) = f(A \setminus S_n \mid S_n) + f(S_n) \leq f(A \setminus S_n : S_n) + f(S_n) \leq \frac{f(S_n)}{c} + f(S_n) = \frac{c+1}{c} \cdot f(S_n) \ ,
$$

where the first equality follows from $S_n \subseteq A$, the first inequality follows from Observation 1, and the second inequality follows from Lemma 12. $\qquad\square$

Our next objective is to show that the value of the elements of the optimal solution that do not belong to A is not too large compared to the value of A itself. To this end, we need a mapping from the elements of the optimal solution to elements of A. Such a mapping is given by Proposition 14. However, before stating Proposition 14, we need to present a simplification given by Reduction 1.

Reduction 1. *For the sake of analyzing the approximation ratio of Algorithm 5, one may assume that every element $u \in \mathcal{N}$ belongs to exactly p out of the m ground sets $\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_m$ of the matroids defining the p-matchoid $(\mathcal{N}, \mathcal{I})$.*

Proof. Proof For every element $u \in \mathcal{N}$ that belongs to the ground sets of only $p' < p$ out of the m matroids $(\mathcal{N}_1, \mathcal{N}_1), (\mathcal{N}_2, \mathcal{N}_2), \dots, (\mathcal{N}_m, \mathcal{I}_m)$, we can add u to $p - p'$ additional matroids as a free element (i.e., an element whose addition to an independent set always keeps the set independent). One can observe that the addition of u to these matroids does not affect the behavior of Algorithm 5 at all, but makes u obey the technical property of belonging to *exactly* p out of the ground sets $\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_m$. $\qquad\square$

From this point on we implicitly make the assumption allowed by Reduction 1. In particular, the proof of Proposition 14 relies on this assumption. To state the proposition, we still need some additional notation. For every $1 \leq i \leq n$, we define

$$
d(i) = \begin{cases} 1 + \max\{i \leq j \leq n \mid u_i \in S_j\} & \text{if } u_i \in A \ , \\ i & \text{otherwise} \ . \end{cases}
$$

20

In general, $d(i)$ is the index of the element whose arrival made Algorithm 5 remove u_i from its solution. Two exceptions to this rule are as follows. If u_i was never added to the solution, then $d(i) = i$; and if u_i was never removed from the solution, then $d(i) = n + 1$.

Proposition 14. *For every set $T \in \mathcal{I}$ which does not include elements of R, there exists a mapping ϕ_T from elements of T to multi-subsets of A such that*

- *every element $u \in S_n$ appears at most p times in the multi-sets of $\{\phi_T(u) \mid u \in T\}$.*

- *every element $u \in A \setminus S_n$ appears at most $p - 1$ times in the multi-sets of $\{\phi_T(u) \mid u \in T\}$.*

- *every element $u_i \in T \setminus A$ obeys $f(u_i \mid S_{i-1}) \leq (1 + c) \cdot \sum_{u_j \in \phi_T(u_i)} f(u_j : S_{d(j)-1})$.*

- *every element $u_i \in T \cap A$ obeys $f(u_i \mid S_{i-1}) \leq f(u_j : S_{d(j)-1})$ for every $u_j \in \phi_T(u_i)$, and the multi-set $\phi_T(u_i)$ contains exactly p elements (including repetitions).*

Because the proof of Proposition 14 is lengthy and detailed, we defer it to Section 4.2.2. Instead, we prove now a useful technical observation. Let $Z = \{u_i \in \mathcal{N} \mid f(u_i \mid S_{i-1}) < 0\}$.

Observation 2. *Consider an arbitrary element $u_i \in \mathcal{N}$.*

- *If $u_i \notin Z$, then $f(u_i : S_{i'}) \geq 0$ for every $i' \geq i - 1$. In particular, since $d(i) \geq i$, $f(u_i : S_{d(i)-1}) \geq 0$.*

- *$A \cap (R \cup Z) = \varnothing$.*

Proof. Proof To see why the first part of the observation is true, consider an arbitrary element $u_i \notin Z$. Then,

$$0 \leq f(u_i \mid S_{i-1}) \leq f(u \mid S_{i'} \cap \{u_1, u_2, \ldots, u_{i-1}\}) = f(u : S_{i'}) \ ,$$

where the second inequality follows from the submodularity of f and the inclusion $S_{i'} \cap \{u_1, u_2, \ldots, u_{i-1}\} \subseteq S_{i-1}$ (which holds because elements are only added by Algorithm 5 to its solution at the time of their arrival).

It remains to prove the second part of the observation. Note that Algorithm 5 adds every arriving element to at most one of the sets A and R, and thus, these sets are disjoint; hence, to prove the observation it is enough to show that A and Z are also disjoint. Assume towards a contradiction that this is not the case, and let u_i be the first element to arrive which belongs to both A and Z. Then,

$$f(u_i \mid S_{i-1}) \geq (1 + c) \cdot f(U_i : S_{i-1}) = (1 + c) \cdot \sum_{u_j \in U_i} f(u_j : S_{d(j)-1}) \ .$$

To see why that inequality leads to a contradiction, notice that its leftmost hand side is negative by our assumption that $u_i \in Z$, while its rightmost hand side is non-negative by the first part of this observation since the choice of u_i implies that no element of $U_i \subseteq S_{i-1} \subseteq A \cap \{u_1, u_2, \ldots, u_{i-1}\}$ can belong to Z. $\qquad\square$

We are now ready to show that the value of the elements of the optimal solution that do not belong to A is not too large compared to the value of A itself when the sampling parameter q is chosen appropriately.

Lemma 15. *If $q = ((1 + c)p + 1)^{-1}$, then $\mathbb{E}[f(S_n)] \geq \frac{c}{(1+c)^2 p} \cdot \mathbb{E}[f(A \cup OPT)]$.*

21

Proof. Proof Since $S_i \subseteq A$ for every $0 \leq i \leq n$, the submodularity of f guarantees that

$$f(A \cup \mathrm{OPT}) \leq f(A) + \sum_{u_i \in \mathrm{OPT} \setminus (R \cup A)} f(u_i \mid A) + \sum_{u_i \in (\mathrm{OPT} \setminus A) \cap R} f(u_i \mid A)$$

$$\leq f(A) + \sum_{u_i \in \mathrm{OPT} \setminus (R \cup A)} f(u_i \mid S_{i-1}) + \sum_{u_i \in (\mathrm{OPT} \setminus A) \cap R} f(u_i \mid S_{i-1})$$

$$\leq \frac{1+c}{c} \cdot f(S_n) + \sum_{u_i \in \mathrm{OPT} \setminus (R \cup A)} f(u_i \mid S_{i-1}) + \sum_{u_i \in \mathrm{OPT} \cap R} f(u_i \mid S_{i-1}) \ ,$$

where the third inequality follows from Corollary 13 and the fact that $A \cap R = \varnothing$ by Observation 2. Let us now consider the function $\phi_{\mathrm{OPT} \setminus R}$ whose existence is guaranteed by Proposition 14 when we choose $T = \mathrm{OPT} \setminus R$. The property guaranteed by Proposition 14 for elements of $T \setminus A$ implies

$$\sum_{u_i \in \mathrm{OPT} \setminus (R \cup A)} f(u_i \mid S_{i-1}) \leq (1+c) \cdot \sum_{\substack{u_i \in \mathrm{OPT} \setminus (R \cup A) \\ u_j \in \phi_{\mathrm{OPT} \setminus R}(u_i)}} f(u_j : S_{d(j)-1}) \ .$$

Additionally,

$$\sum_{\substack{u_i \in \mathrm{OPT} \setminus (R \cup A) \\ u_j \in \phi_{\mathrm{OPT} \setminus R}(u_i)}} f(u_j : S_{d(j)-1}) + p \cdot \sum_{u_i \in \mathrm{OPT} \cap A} f(u_i \mid S_{i-1})$$

$$\leq \sum_{\substack{u_i \in \mathrm{OPT} \setminus R \\ u_j \in \phi_{\mathrm{OPT} \setminus R}(u_i)}} f(u_j : S_{d(j)-1})$$

$$\leq p \cdot \sum_{u_j \in S_n} f(u_j : S_n) + (p-1) \cdot \sum_{u_j \in A \setminus S_n} f(u_j : S_{d(j)-1})$$

$$\leq p \cdot f(S_n) + \frac{p-1}{c} \cdot f(S_n)$$

$$= \frac{(1+c) \cdot p - 1}{c} \cdot f(S_n) \ ,$$

where the first inequality follows from the properties guaranteed by Proposition 14 for elements of $T \cap A$ (note that the sets $\mathrm{OPT} \setminus (R \cup A)$ and $OPT \cap A$ are a disjoint partition of $\mathrm{OPT} \setminus R$ by Observation 2), and the second inequality follows from the properties guaranteed by Proposition 14 for elements of $A \setminus S_n$ and S_n because every element u_i in the multisets produced by $\phi_{\mathrm{OPT} \setminus R}$ belongs to A, and thus, obeys $f(u_i : S_{d(i)-1}) \geq 0$ by Observation 2. Finally, the last inequality follows from Lemma 12 and the fact that $f(u_j : S_{d(j)-1}) \leq f(u_j : S_n)$ for every $1 \leq j \leq n$. Combining all the above inequalities, we get

$$f(A \cup \mathrm{OPT})$$

$$\leq \frac{1+c}{c} \cdot f(S_n) + (1+c) \cdot \left[\frac{(1+c) \cdot p - 1}{c} \cdot f(S_n) - p \cdot \sum_{u_i \in \mathrm{OPT} \cap A} f(u_i \mid S_{i-1}) \right] + \sum_{u_i \in \mathrm{OPT} \cap R} f(u_i \mid S_{i-1})$$

$$= \frac{(1+c)^2 \cdot p}{c} \cdot f(S_n) - (1+c)p \cdot \sum_{u_i \in \mathrm{OPT} \cap A} f(u_i \mid S_{i-1}) + \sum_{u_i \in \mathrm{OPT} \cap R} f(u_i \mid S_{i-1}) \ .$$

By the linearity of expectation, to prove the lemma it suffices to show that the expectation of the last two terms is non-positive. We will show the stronger statement that the expectation of

the last two terms is zero. To this end, consider an arbitrary element $u_i \in$ OPT. When u_i arrives, one of two things happens. The first option is that Algorithm 5 discards u_i without adding it to either its solution or to R. The other option is that Algorithm 5 adds u_i to its solution (and thus, to A) with probability q, and to R with probability $1 - q$. The crucial observation here is that at the time of u_i's arrival the set S_{i-1} is already determined, and thus, this set is independent of the decision of the algorithm to add u to A or to R; which implies the following equality (given an event $\mathcal{E}$, we use here $\mathbf{1}[\mathcal{E}]$ to denote an indicator for it).

$$\frac{\mathbb{E}[\mathbf{1}[u_i \in A] \cdot f(u_i \mid S_{i-1})]}{q} = \frac{\mathbb{E}[\mathbf{1}[u_i \in R] \cdot f(u_i \mid S_{i-1})]}{1-q} \enspace .$$

Rearranging the last equality, and summing it up over all elements $u_i \in$ OPT, we get

$$\frac{1-q}{q} \cdot \mathbb{E}\left[\sum_{u_i \in \text{OPT} \cap A} f(u_i \mid S_{i-1}) \right] = \mathbb{E}\left[\sum_{u_i \in \text{OPT} \cap R} f(u_i \mid S_{i-1}) \right] \enspace .$$

By assumption, $q = ((1+c)p + 1)^{-1}$, which implies $(1-q)/q = q^{-1} - 1 = (c+1)p$. Substituting this into the equality above completes the proof. $\qquad\square$

Now we are ready to prove the approximation and efficiency guarantees of Theorem 2.

4.2.1 Proof of Theorem 2.

We first prove that Algorithm 3 achieves the approximation ratios guaranteed by Theorem 2. As discussed earlier, Algorithms 3 and 5 have identical output distributions, and so it suffices to show that Algorithm 5 achieves the desired approximation ratios. Recall that $q = ((1+c)p + 1)^{-1}$, so by Lemma 15,

$$\mathbb{E}[f(S_n)] \geq \frac{c}{(1+c)^2 p} \cdot \mathbb{E}[f(A \cup \text{OPT})] \enspace . \tag{8}$$

Suppose that f is monotone. Setting $c = 1$ yields $\frac{c}{(1+c)^2 p} = 1/4p$. Additionally, the monotonicity of f implies that $\mathbb{E}[f(A \cup \text{OPT})] \geq f(\text{OPT})$. Substituting these two observations into (8) yields

$$\mathbb{E}[f(S_n)] \geq \frac{1}{4p} f(\text{OPT}) \enspace ,$$

which establishes the approximation guarantee in the monotone case. Consider now the more general case in which f is not necessarily monotone. Note that each element appears in A with probability at most q due to subsampling, thus, by Lemma 3, we have $\mathbb{E}[f(A \cup \text{OPT})] \geq (1-q) f(\text{OPT})$. Substituting this into Inequality (8) and setting $c = \sqrt{1 + 1/p}$ yields

$$\begin{aligned}
\mathbb{E}[f(S_n)] &\geq (1-q) \frac{c}{(1+c)^2 p} \cdot f(\text{OPT}) \\
&= \left(1 - \frac{1}{(1+c)p + 1} \right) \frac{c}{(1+c)^2 p} \cdot f(\text{OPT}) \\
&= \left(1 - \frac{1}{(1 + \sqrt{1+1/p})p + 1} \right) \frac{\sqrt{1+1/p}}{(1 + \sqrt{1+1/p})^2 p} \cdot f(\text{OPT}) \\
&= \frac{1}{2p + 2\sqrt{p(p+1)} + 1} \cdot f(\text{OPT}),
\end{aligned}$$

which yields the promised approximation guarantee for this case.

Now we verify that SAMPLE-STREAMING achieves the guaranteed memory and oracle complexities. Algorithm 3 has to keep the following three sets in memory: S_i, U_i and X_ℓ. Since U_i and X_ℓ are subsets of S_{i-1}, they are independent, and so is S_i. Hence, each one of the three sets S_i, U_i and X_ℓ contains at most k elements. Thus, $O(k)$ memory suffices for the algorithm. When an arriving element is not sampled (which happens with probability $1-q$), no queries to the evaluation or independence oracles are required. A sampled element requires $O(km)$ queries. Because $q = O(1/p)$, an arriving element requires $q \cdot O(km) = O(km/p)$ oracle queries in expectation.

4.2.2 Proof of Proposition 14.

In this section we prove Proposition 14. Before doing so, we introduce some terminology regarding matroids which will be used in the proof. A *circuit* is a dependent set which is minimal with respect to inclusion; that is, $C \notin \mathcal{I}$ is a circuit if $A \notin \mathcal{I}$ and $A \subseteq C$ imply $C = A$. An element u is *spanned* by a set S if the maximum size independent subsets of S and $S + u$ are of the same size. Note that it follows from these definitions that every element of u of a circuit C is spanned by $C - u$.

Let us also recall some of the sets involved in Algorithm 5. The sequence $S_1, \ldots S_n$ are the solutions constructed by the algorithm and S_n is the final returned solution. The set $A = \cup_{i=1}^n S_i$ is the set of all elements which were added to some solution and the set R contains elements which had sufficiently large marginal gain but were rejected with probability q. Lastly, $d(i)$ is the index of the element whose arrival made Algorithm 5 remove u_i from its solution. Now, we restate the proposition itself.

Proposition 14. *For every set $T \in \mathcal{I}$ which does not include elements of R, there exists a mapping ϕ_T from elements of T to multi-subsets of A such that*

- *every element $u \in S_n$ appears at most p times in the multi-sets of $\{\phi_T(u) \mid u \in T\}$.*

- *every element $u \in A \setminus S_n$ appears at most $p-1$ times in the multi-sets of $\{\phi_T(u) \mid u \in T\}$.*

- *every element $u_i \in T \setminus A$ obeys $f(u_i \mid S_{i-1}) \leq (1+c) \cdot \sum_{u_j \in \phi_T(u_i)} f(u_j : S_{d(j)-1})$.*

- *every element $u_i \in T \cap A$ obeys $f(u_i \mid S_{i-1}) \leq f(u_j : S_{d(j)-1})$ for every $u_j \in \phi_T(u_i)$, and the multi-set $\phi_T(u_i)$ contains exactly p elements (including repetitions).*

We begin the proof of Proposition 14 by constructing m graphs, one for each of the matroids defining $\mathcal{M}$. For every $1 \leq \ell \leq m$, the graph G_ℓ contains two types of vertices: its internal vertices are the elements of $A \cap \mathcal{N}_\ell$, and its external vertices are the elements of $\{u_i \in \mathcal{N}_\ell \setminus (R \cup A) \mid (S_{i-1} + u_i) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell\}$. Informally, the external elements of G_ℓ are the elements of $\mathcal{N}_\ell$ which were rejected upon arrival by Algorithm 5 and the matroid $\mathcal{M}_\ell = (\mathcal{N}_\ell, \mathcal{I}_\ell)$ can be (partially) blamed for this rejection.

The arcs of G_ℓ are created using the following iterative process that creates some arcs of G_ℓ in response to every arriving element. For every $1 \leq i \leq n$, consider the element x_ℓ selected by the execution of EXCHANGE-CANDIDATE on the element u_i and the set S_{i-1}. From this point on we denote this element by $x_{i,\ell}$. If no $x_{i,\ell}$ element was selected by the above execution of EXCHANGE-CANDIDATE, or $u_i \in R$, then no G_ℓ arcs are created in response to u_i. Otherwise, let $C_{i,\ell}$ be the single circuit of the matroid $\mathcal{M}_\ell$ in the set $(S_{i-1} + u_i) \cap \mathcal{N}_\ell$—there is exactly one circuit of $\mathcal{M}_\ell$ in this set because S_{i-1} is independent, but $(S_{i-1} + u_i) \cap \mathcal{N}_\ell$ is not independent in $\mathcal{M}_\ell$. One can observe that $C_{i,\ell} - u_i$ is equal to the set X_ℓ in the above-mentioned execution of EXCHANGE-CANDIDATE, and thus, $x_{i,\ell} \in C_{i,\ell}$. We now denote by $u'_{i,\ell}$ the vertex out of $\{u_i, x_{i,\ell}\}$ that does not belong to S_i—notice that there is exactly one such vertex since $x_{i,\ell} \in U_i$, which implies that it appears in S_i if $S_i = S_{i-1}$ and does not appear in S_i if $S_i = S_{i-1} \setminus U_i + u_i$.
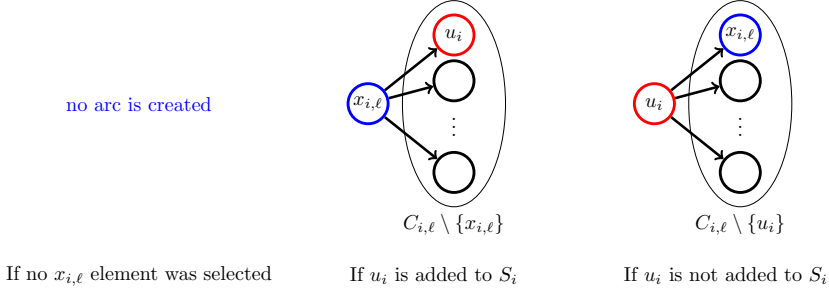
24

Figure 2: A sketch of how arcs are constructed in the graph G_ℓ at each iteration.

Regardless of the node chosen as $u'_{i,\ell}$, the arcs of G_ℓ created in response to u_i are all the possible arcs from $u'_{i,\ell}$ to the other vertices of $C_{i,\ell}$. Observe that these are valid arcs for G_ℓ in the sense that their endpoints (i.e., the elements of $C_{i,\ell}$) are all vertices of G_ℓ (for the elements of $C_{i,\ell} - u_i$ this is true since $C_{i,\ell} - u_i \subseteq S_{i-1} \cap \mathcal{N}_\ell \subseteq A \cap \mathcal{N}_\ell$, and for the element u_i this is true since the existence of $x_{i,\ell}$ implies $(S_{i-1} + u_i) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell$). See Figure 2 for a sketch of how arcs are added to G_ℓ.

Some properties of G_ℓ are given by the following observation. Given a graph G and a vertex u, we denote by $\delta_G^+(u)$ the set of vertices to which there is a direct arc from u in G.

Observation 3. *For every $1 \leq \ell \leq m$,*

- *every non-sink vertex u of G_ℓ is spanned by the set $\delta_{G_\ell}^+(u)$.*

- *for every two indexes $1 \leq i, j \leq n$, if $u'_{i,\ell}$ and $u'_{j,\ell}$ both exist and $i \neq j$, then $u'_{i,\ell} \neq u'_{j,\ell}$.*

- *G_ℓ is a directed acyclic graph.*

Proof. Proof Consider an arbitrary non-sink node u of G_ℓ. Since there are arcs leaving u, u must be equal to $u'_{i,\ell}$ for some $1 \leq i \leq n$. This implies that u belongs to the circuit $C_{i,\ell}$, and that there are arcs from u to every other vertex of $C_{i,\ell}$. Thus, u is spanned by the vertices of $\delta_{G_\ell}^+(u) \supseteq C_{i,\ell} - u$ because the fact that $C_{i,\ell}$ is a circuit containing u implies that $C_{i,\ell} - u$ spans u. This completes the proof of the first part of the observation.

Let us prove now a useful technical claim. Consider an index $1 \leq i \leq n$ such that $u'_{i,\ell}$ exists, and let j be an arbitrary value $i < j \leq n$. We will prove that $u'_{i,\ell}$ does not belong to $C_{j,\ell}$. By definition, $u'_{i,\ell}$ is either u_i or the vertex $x_{i,\ell}$ that belongs to S_{i-1}, and thus, arrived before u_i and is not equal to u_j; hence, in both cases, we have that $u'_{i,\ell} \neq u_j$. Moreover, combining the fact that $u'_{i,\ell}$ is either u_i or arrived before u_i and the observation that $u'_{i,\ell}$ is never a part of S_i, we get that $u'_{i,\ell}$ cannot belong to $S_j \supseteq C_{j,\ell} - u_j$, which implies the claim together with the previous observation that $u'_{i,\ell} \neq u_j$.

The technical claim that we proved above implies the second part of the lemma, namely that for every two indexes $1 \leq i, j \leq n$, if $u'_{i,\ell}$ and $u'_{j,\ell}$ both exist and $i \neq j$, then $u'_{i,\ell} \neq u'_{j,\ell}$. To see why that is the case, assume without loss of generality $i < j$. Then, the above technical claim implies that $u'_{i,\ell} \notin C_{j,\ell}$, which implies $u'_{i,\ell} \neq u'_{j,\ell}$ because $u'_{j,\ell} \in C_{j,\ell}$.

At this point, let us assume towards a contradiction that the third part of the observation is not true, i.e., that there exists a circuit L in G_ℓ. Since every vertex of L has a non-zero out degree, every such vertex must be equal to $u'_{i,\ell}$ for some $1 \leq i \leq n$. Thus, there must be indexes $1 \leq i_1 < i_2 \leq n$ such that L contains an arc from $u'_{i_2,\ell}$ to $u'_{i_1,\ell}$. Since we already proved that $u'_{i_2,\ell}$ cannot be equal to $u'_{j,\ell}$ for any $j \neq i_2$, the arc from $u'_{i_2,\ell}$ to $u'_{i_1,\ell}$ must have been created in response to u_{i_2}, hence, $u'_{i_1,\ell} \in C_{i_2,\ell}$, which contradicts the technical claim we have proved. $\qquad\square$

One consequence of the properties of G_ℓ proved by the last observation is given by the following lemma. A slightly weaker version of this lemma was proved implicitly by Badanidiyuru [2011], and was stated as an explicit lemma by Chekuri et al. [2015].

Lemma 16. *Consider an arbitrary directed acyclic graph $G = (V, E)$ whose vertices are elements of some matroid $\mathcal{M}'$. If every non-sink vertex u of G is spanned by $\delta_G^+(u)$ in $\mathcal{M}'$, then for every set S of vertices of G which is independent in $\mathcal{M}'$ there must exist an injective function $\psi_S : S \to V$ such that, for every vertex $u \in S$, $\psi_S(u)$ is a sink of G which is reachable from u.*

Proof. Proof Let us define the *width* of a set S of vertices of G as the number of arcs that appear on some path starting at a vertex of S (more formally, the width of S is the size of the set $\{e \in E \mid \text{there is a path in } G \text{ that starts in a vertex of } S \text{ and includes } e\}$). We prove the lemma by induction of the width of S. We begin by considering the base case where S is of width 0. In this case, the vertices of S cannot have any outgoing arcs because such arcs would have contributed to the width of S, and thus, they are all sinks of G. Thus, the lemma holds for the trivial function ψ_S mapping every element of S to itself. Assume now that the width w of S is larger than 0, and assume that the lemma holds for every set of width smaller than w. Let u be a non-sink vertex of S such that there is no path in G from any other vertex of S to u. Notice that such a vertex must exist since G is acyclic. By the assumption of the lemma, $\delta^+(u)$ spans u. In contrast, since S is independent, $S - u$ does not span u, and thus, there must exist an element $v \in \delta^+(u) \setminus S$ such that the set $S' = S - u + v$ is independent.

Let us explain why the width of S' must be strictly smaller than the width of S. First, consider an arbitrary arc e which is on a path starting at a vertex $u' \in S'$. If $u' \in S$, then e is also on a path starting in a vertex of S. On the other hand, if $u' \notin S$, then u' must be the vertex v. Thus, e must be on a path P starting in v. Adding uv to the beginning of the path P, we get a path from u which includes e. Hence, in conclusion, we have got that every arc e which appears on a path starting in a vertex of S' (and thus, contributes to the width of S') also appears on a path starting in a vertex of S (and thus, also contributes to the width of S); which implies that the width of S' is not larger than the width of S. To see that the width of S' is actually strictly smaller than the width of S, it only remains to find an arc which contributes to the width of S, but not to the width of S'. Towards this goal, consider the arc uv. Since u is a vertex of S, the arc uv must be on some path starting in u (for example, the path including only this arc), and thus, contributes to the width of S. Assume now towards a contradiction that uv contributes also to the width of S', i.e., that there is a path P starting at a vertex $w \in S'$ which includes uv. If $w = v$, then this leads to a contradiction since it implies the existence of a circuit in G. On the other hand, if $w \neq v$, then this implies a path in G from a vertex $w \neq u$ of S to u, which contradicts the definition of u. This completes the proof that the width of S' is strictly smaller than the width of S.

Using the induction hypothesis, we now get that there exists an injective function $\psi_{S'}$ mapping every vertex of S' to a sink of G. Using $\psi_{S'}$, we can define ψ_S as follows. For every $w \in S$,

$$\psi_S(w) = \begin{cases} \psi_{S'}(v) & \text{if } w = u \ , \\ \psi_{S'}(w) & \text{otherwise} \ . \end{cases}$$

Since u appears in S but not in S', and v appears in S' but not in S, the injectiveness of ψ_S follows from the injectiveness of $\psi_{S'}$. Moreover, ψ_S clearly maps every vertex of S to a sink of G since $\psi_{S'}$ maps every vertex of S' to such a sink. Finally, one can observe that $\psi_S(w)$ is reachable from w for every $w \in S$ because $\psi_S(u) = \psi_{S'}(v)$ is reachable from v by the definition of $\psi_{S'}$, and thus, also from u due to the existence of the arc uv. □

For every $1 \leq \ell \leq m$, let T_ℓ be the set of elements of T that appear as vertices of G_ℓ. Since T is independent and T_ℓ contains only elements of $\mathcal{N}_\ell$, Observation 3 and Lemma 16 imply together the existence of an injective function ψ_{T_ℓ} mapping the elements of T_ℓ to sink vertices of G_ℓ. We can now define the function ϕ_T promised by Proposition 14. For every element $u \in T$, the function ϕ_T maps u to the multi-set $\{\psi_{T_\ell}(u) \mid 1 \leq \ell \leq m \text{ and } u \in T_\ell\}$, where we assume that repetitions are kept when the expression $\psi_{T_\ell}(u)$ evaluates to the same element for different choices of ℓ. Let us explain why the elements in the multi-sets produced by ϕ_T are indeed all elements of A, as is required by the proposition. Consider an element $u_i \notin A$, and let us show that it does not appear in the range of ψ_{T_ℓ} for any $1 \leq \ell \leq m$. If u_i does not appear as a vertex in G_ℓ, then this is obvious. Otherwise, the fact that $u_i \notin A$ implies $u'_{i,\ell} = u_i$, and thus, the arcs of G_ℓ created in response to u_i are arcs leaving u_i, which implies that u_i is not a sink of G_ℓ, and hence, does not appear in the range of ψ_{T_ℓ}.

Recall that every element $u \in \mathcal{N}$ belongs to at most p out of the ground sets $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_m$, and thus, is a vertex in at most p out of the graphs $G_1, G_2, \ldots, G_m$. Since ψ_{T_ℓ} maps every element to vertices of G_ℓ, this implies that u is in the range of at most p out of the functions $\psi_{T_1}, \psi_{T_2}, \ldots, \psi_{T_m}$. Moreover, since these functions are injective, every one of these functions that have u in its range maps at most one element to u. Thus, the multi-sets produced by ϕ_T contain u at most p times. Since this is true for every element of $\mathcal{N}$, it is true in particular for the elements of S_n, which is the first property of ϕ_T that we needed to prove.

Consider now an element $u \in A \setminus S_n$. Our next objective is to prove that u appears at most $p-1$ times in the multi-sets produced by ϕ_T, which is the second property of ϕ_T that we need to prove. Above, we proved that u appears at most p times in these multi-sets by arguing that every such appearance must be due to a function ψ_{T_ℓ} that has u in its range, and that the function ψ_{T_ℓ} can have this property only for the p values of ℓ for which $u \in \mathcal{N}_\ell$. Thus, to prove that u in fact appears at most $p-1$ times in the multi-sets produced by ϕ_T, it is enough to argue that there exists a value ℓ such that $e \in \mathcal{N}_\ell$, but ψ_{T_ℓ} does not have u in its range. Let us prove that this follows from the membership of u in $A \setminus S_n$. Since u was removed from the solution of Algorithm 5 at some point, there must be some index $1 \leq i \leq n$ such that both $u \in U_i$ and u_i was added to the solution of Algorithm 5. Since $u \in U_i$, there must be a value $1 \leq \ell \leq m$ such that $u = x_{i,\ell}$, and since u_i was added to the solution of Algorithm 5, $u'_{i,\ell} = x_{i,\ell}$. These equalities imply together that there are arcs leaving u in G_ℓ (which were created in response to u_i). Thus, the function ψ_{T_ℓ} does not map any element to u because u is not a sink of G_ℓ, despite the fact that $u \in \mathcal{N}_\ell$.

To prove the other guaranteed properties of ϕ_T, we need the following lemma.

Lemma 17. *Consider two vertices u_i and u_j such that u_j is reachable from u_i in G_ℓ. If $u_i \in A$, then $f(u_i : S_{d(i)-1}) \leq f(u_j : S_{d(j)-1})$, otherwise, $f(x_{i,\ell} : S_{i-1}) \leq f(u_j : S_{d(j)-1})$.*

Proof. Proof We begin by proving a weaker version of this lemma that makes the following two simplifying assumptions: (i) $u_i \in A$, and (ii) there is a direct arc from u_i to u_j. The existence of the last arc implies that there is some value $1 \leq h \leq n$ such that $u'_{h,\ell} = u_i$ and $u_j \in C_{h,\ell}$. Since $u_i \in A$ is an internal vertex of G_ℓ, it cannot be equal to u_h because this would have implied that u_h was rejected immediately by Algorithm 5, and is thus, not internal. Thus, $u_i = x_{h,\ell}$. Recall now that $C_{h,\ell} - u_h$ is equal to the set X_ℓ chosen by EXCHANGE-CANDIDATE when it is executed with the element u_h and the set S_{h-1}; and let us consider two cases.

Case 1 In the case of $u_j \neq u_h$, the fact that $u_i = x_{h,\ell}$ and the way $x_{h,\ell}$ is chosen out of X_ℓ imply

$$f(u_i : S_{d(i)-1}) = f(u_i : S_{h-1}) \leq f(u_j : S_{h-1}) \leq f(u_j : S_{d(j)-1}) \ ,$$

where the equality holds since $u'_{h,\ell} = u_i$ implies $d(i) = h$ and the last inequality holds since $f(u_j : S_{r-1})$ is a non-decreasing function of r when $r \geq j$ and the membership of u_j in $C_{h,\ell}$ implies $j \leq h \leq d(j)$.

Case 2 It remains to consider the case $u_j = u_h$. In this case, the fact that $u_j = u_h$ is accepted into the solution of Algorithm 5 implies

$$f(u_j : S_{d(j)-1}) \geq f(u_j : S_{j-1}) = f(u_j \mid S_{j-1} \cap \{u_1, u_2, \ldots, u_{j-1}\}) = f(u_j \mid S_{j-1})$$
$$= f(u_h \mid S_{h-1}) \geq (1+c) \cdot f(U_h : S_{h-1}) \geq f(U_h : S_{h-1})$$
$$\geq f(x_{h,\ell} : S_{h-1}) = f(u_i : S_{h-1}) = f(u_i : S_{d(i)-1}) \; ,$$

where the first inequality holds since $d(j) \geq j$ by definition, the last equality holds since $u'_{h,\ell} = u_i$ implies $d(i) = h$ and the last two inequalities follow from the fact that the elements of $U_h \subseteq A$ do not belong to Z by Observation 2, which implies (again, by Observation 2) that $f(u : S_{h-1}) \geq 0$ for every $u \in U_h$. This completes the proof of the weaker version of the lemma.

We begin the proof of the full version of the lemma by showing that no arc of G_ℓ goes from an internal vertex to an external one. Assume this is not the case, and that there exists an arc uv of G_ℓ from an internal vertex u to an external vertex v. By definition, there must be a value $1 \leq h \leq n$ such that v belongs to the circuit $C_{h,\ell}$ and $u'_{h,\ell} = u$. The fact that u is an internal vertex implies that u_h must have been accepted by Algorithm 5 upon arrival because otherwise we would have gotten $u = u'_{h,\ell} = u_h$, which implies that u is external, and thus, leads to a contradiction. Consequently, we get $C_{h,\ell} \subseteq A$ because every element of $C_{h,\ell}$ must either be u_h or belong to S_{h-1}. In particular, $v \in A$, which contradicts our assumption that v is an external vertex.

We are now ready to prove the lemma for $u_i \in A$. Consider some path P from u_i to u_j, and let us denote the vertices of this path by $u_{r_0}, u_{r_1}, \ldots, u_{r_{|P|}}$. Since u_i is an internal vertex of G_ℓ and we already proved that no arc of G_ℓ goes from an internal vertex to an external one, all the vertices of P must be internal. Thus, by applying the weaker version of the lemma that we have already proved to every pair of adjacent vertices along the path P, we get that the expression $f(u_{r_k} : S_{d(r_k)-1})$ is a non-decreasing function of k, and in particular,

$$f(u_i : S_{d(i)-1}) = f(u_{r_0} : S_{d(r_0)-1}) \leq f(u_{r_k} : S_{d(r_k)-1}) = f(u_j : S_{d(j)-1}) \; .$$

It remains to prove the lemma for $u_i \notin A$. Let u_h denote the vertex immediately after u_i on some path from u_i to u_j in G_ℓ. Since $u_i \notin A$, we get that $u'_{i,\ell} = u_i$, which implies that the arcs of G_ℓ that were created in response to u_i go from u_i to the vertices of $C_{i,\ell} - u_i$. Since Observation 3 guarantees that $u_i = u'_{i,\ell} \neq u'_{j,\ell}$ for every value $1 \leq j \leq n$ which is different from i, there cannot be any other arcs in G_ℓ leaving u_i, and thus, the existence of an arc from u_i to u_h implies $u_h \in C_{i,\ell} - u_i$. Recall now that $C_{i,\ell} - u_i$ is equal to the set X_ℓ in the execution of EXCHANGE-CANDIDATE corresponding to the element u_i and the set S_{i-1}, and thus, by the definition of $x_{i,\ell}$, $f(x_{i,\ell} : S_{i-1}) \leq f(u_h : S_{i-1})$. Additionally, as an element of $C_{i,\ell} - u_i$, u_h must be a member of $S_{i-1} \subseteq A$, and thus, by the part of the lemma we have already proved, we get $f(u_h : S_{d(h)-1}) \leq f(u_j : S_{d(j)-1})$ because u_j is reachable from u_h. Combining the two inequalities we have proved, we get

$$f(x_{i,\ell} : S_{i-1}) \leq f(u_h : S_{i-1}) \leq f(u_h : S_{d(h)-1}) \leq f(u_j : S_{d(j)-1}) \; ,$$

where the second inequality holds since the fact that $u_h \in C_{i,\ell} - u_i \subseteq S_{i-1}$ implies $d(h) \geq i$. $\quad \square$

Consider now an arbitrary element $u_i \in T \setminus A$. Let us denote by u_{r_ℓ} the element $u_{r_\ell} = \psi_{T_\ell}(u_i)$ if it exists, and recall that this element is reachable from u_i in G_ℓ. Thus, the fact that u_i is not

in A implies

$$f(u_i \mid S_{i-1}) \leq (1+c) \cdot \sum_{u \in U_i} f(u : S_{i-1}) = (1+c) \cdot \sum_{\substack{1 \leq \ell \leq m \\ (S_{i-1}+u_i) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell}} f(x_{i,\ell} : S_{i-1})$$

$$\leq (1+c) \cdot \sum_{\substack{1 \leq \ell \leq m \\ (S_{i-1}+u_i) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell}} f(u_{r_\ell} : S_{d(r_\ell)}) = (1+c) \cdot \sum_{u_j \in \phi_T(u_i)} f(u_j : S_{d(j)}) \ ,$$

where the second inequality follows from Lemma 17 and the last equality holds since the values of ℓ for which $(S_{i-1} + u_i) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell$ are exactly the values for which $u_i \in T_\ell$, and thus, they are all also exactly the values for which the multi-set $\phi_T(u_i)$ includes the value of $\psi_{T_\ell}(u_i)$. This completes the proof of the third property of ϕ_T that we need to prove.

Finally, consider an arbitrary element $u_i \in A \cap T$. Every element $u_j \in \phi_T(u_i)$ can be reached from u_i in some graph G_ℓ, and thus, by Lemma 17,

$$f(u_i \mid S_{i-1}) = f(u_i \mid S_{i-1} \cap \{u_1, u_2, \ldots, u_{i-1}\}) = f(u_i : S_{i-1})$$

$$\leq f(u_i : S_{d(i)-1}) \leq f(u_j : S_{d(j)-1}) \ ,$$

where the first inequality holds since $d(i) \geq i$ by definition and $f(u_i : S_{r-1})$ is a non-decreasing function of r for $r \geq i$. Additionally, we observe that u_i, as an element of $T \cap A$, belongs to T_ℓ for every value $1 \leq \ell \leq m$ for which $u_i \in \mathcal{N}_\ell$, and thus, the size of the multi-set $\phi_T(u_i)$ is equal to the number of ground sets out of $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_m$ that include u_i. Since we assume by Reduction 1 that every element belongs to exactly p out of these ground sets, we get that the multi-set $\phi_T(u_i)$ contains exactly p elements (including repetitions), which completes the proof of Proposition 14.

5 Experimental Results

In this section, we compare the performance of our proposed offline and streaming algorithms with the state-of-the-art methods for maximizing non-monotone submodular functions on real-world datasets . We empirically demonstrate that by using the subsampling technique, our algorithms return solutions with similar utility values as other methods, but at a fraction of computational cost. We first compare the performance of SAMPLEGREEDY with other competitive offline algorithms. Then, we evaluate the effectiveness of SAMPLE-STREAMING in terms of utility and scalability.

5.1 Offline Algorithms

In this section, we compare the performance of SAMPLEGREEDY with two other offline algorithms: FANTOM [Mirzasoleiman et al., 2016a] and REPEATEDGREEDY [Feldman et al., 2017], both of which are repeated greedy techniques requiring $O(nkp)$ and $O(nk\sqrt{p})$ oracle queries, respectively. In order to improve the performance of SAMPLEGREEDY in terms of utility, we also consider a boosted version of SAMPLEGREEDY by taking the best of four runs, denoted MAX-SAMPLEGREEDY. We test these algorithms on a personalized movie recommendation system and find that while SAMPLEGREEDY and its boosted variant return comparable solutions to REPEATEDGREEDY and FANTOM, they run orders of magnitude faster.

In the movie recommendation system application, we observe movie ratings from users, and our objective is to recommend movies to users based on their reported favorite genres. In particular, given a user-specified input of favorite genres, we would like to recommend a short list

of movies that are diverse, and yet representative, of those genres. The similarity score between movies that we use is derived from user ratings similar to the methods used in [Lindgren et al., 2015]. Next, we describe the experiment setting in more detail.

Let $\mathcal{N}$ be a set of movies, and $\mathcal{G} = \{G_1, \ldots G_p\}$ be the set of all movie genres, where each genre is a subset $G_i \subseteq \mathcal{N}$. Note that each movie may be identified with multiple genres. Let $s_{i,j}$ be a non-negative similarity score between movies $i, j \in \mathcal{N}$, and suppose a user u seeks a representative set of movies from genres $\mathcal{G}_u \subseteq \mathcal{G}$. Note that the set of movies from these genres is $\mathcal{N}_u = \cup_{G_i \in \mathcal{G}_u} G_i$. A reasonable utility function for choosing a diverse yet representative set of movies S for u is

$$f_u(S) = \sum_{i \in S} \sum_{j \in \mathcal{N}_u} s_{i,j} - \lambda \sum_{i \in S} \sum_{j \in S} s_{i,j}, \tag{9}$$

for some parameter $0 \leq \lambda \leq 1$. Observe that the first term is a sum-coverage function that captures the representativeness of S, and the second term is a dispersion function penalizing similarity within S. When $\lambda > 0$, this function is non-monotone and for $\lambda = 1$, this utility is the usual cut function.

The user may specify an upper limit k on the number of movies in his recommended set S, as well as an upper limit k_i on the number of movies from each genre G_i appearing in S (we call the parameters $k_1, \ldots k_p$ *genre limits*). One can show that these constraints on the recommended movie set S form a p-extendible system.

For our experiments, we use the MovieLens 20M dataset, which features 20 million ratings of 27,000 movies by 138,000 users. To obtain a similarity score between movies, we take an approach developed in [Lindgren et al., 2015]. First, we fill missing entries of an incomplete movie-user matrix $M \in \mathbb{R}^{n \times m}$ via low-rank matrix completion [Candés and Recht, 2008, Hastie et al., 2015], where m is the total number of users. Then we randomly sample to obtain a matrix $\tilde{M} \in \mathbb{R}^{n \times \ell}$ where $\ell \ll m$ and the inner products between rows is preserved. The similarity score between movies i and j is then defined as the inner product of their corresponding rows in $\tilde{M}$. In our experiment, we set the total recommended movies limit to $k = 10$. The genre limits k_g are set to be equal across genres, and we vary them from 1 to 6. For all experiments, we use $\lambda = 0.9$. Finally, we set our favorite genres G_u to be *Adventure*, *Animation* and *Fantasy*; thus, the corresponding constraint set is 3-extendible. For each algorithm and test instance, we record the function value of the returned solution S and the number of evaluations of the objective f, which is a machine-independent measure of run-time.

The result of this experiment is depicted in Figure 3. First, Figure 3a shows the utility value of the solution sets for the various algorithms. As we see from Figure 3a, FANTOM consistently returns a solution set with the highest function value. However, SAMPLEGREEDY and MAX-SAMPLEGREEDY return solution sets with similarly high function values. Even for four runs, MAX-SAMPLEGREEDY noticeably increases the utility of the returned solution. Both SAMPLE-GREEDY and MAX-SAMPLEGREEDY return solutions with larger utility than Greedy. Figure 3b shows the number of function calls made by each algorithm as the genre limit k_g is varied. For each algorithm, the number of function calls appears roughly constant as k_g is varied—this is likely due to the lazy greedy implementation and also to the small values used for k_g. We observe that SAMPLEGREEDY runs roughly two orders magnitude faster than REPEATEDGREEDY and three orders of magnitude faster than FANTOM. Moreover, even after we boost SAMPLEGREEDY by executing it a few times, its computational cost remains much lower than that of the other algorithms.

To better analyze the trade-off between the utility of the solution value and the computational cost, in Figures 3c and 3d, we compare the ratio of these measurements for the various algorithms using FANTOM as a baseline. For both cases of $k_g = 1$ and $k_g = 3$, we see that MAX-SAMPLEGREEDY provides nearly the same utility as FANTOM, while only incurring around
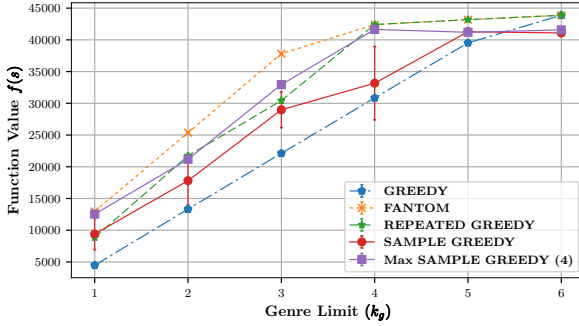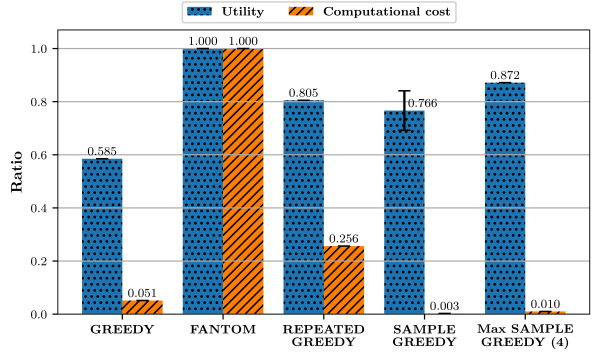
(a) Solution Quality
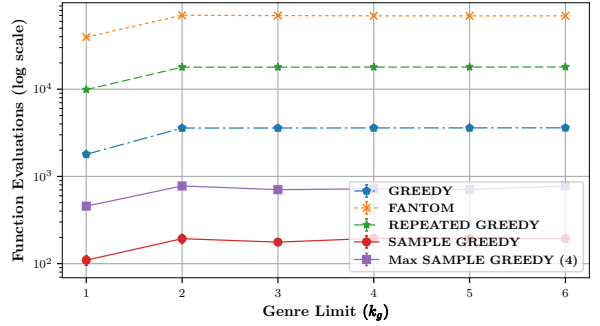
(b) Run Time
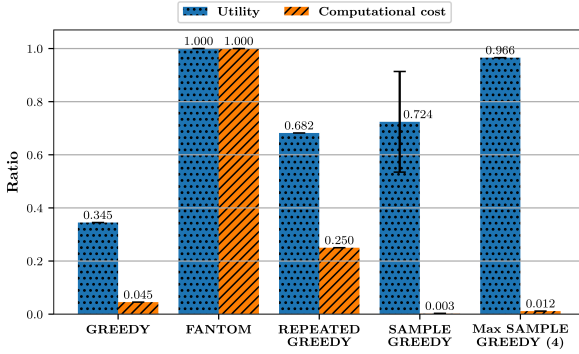
(c) Ratio Comparison $k_g = 1$

(d) Ratio Comparison $k_g = 3$

Figure 3: Performance Comparison. 3a shows the function value of the returned solutions for tested algorithms with varying genre limit k_g. 3b shows the number of function evaluations on a logarithmic scale with varying genre limit k_g. 3c and 3d show the ratio of solution quality and cost with FANTOM serving as a baseline.

1% of the computational cost. For the case of $k_g = 3$, SAMPLEGREEDY achieves 76.6% of the utility of FANTOM, while incurring only 0.3% of the computational cost. Thus, we may conclude that our algorithm provides solutions whose quality is on par with current state-of-the-art, and yet they run in a small fraction of the time.

While Greedy may commit to poor solutions early on which may not be improved, SAMPLE-GREEDY avoids this (in expectation) by only considering a fraction of the ground set. Fortunately, the movie recommendation system has a very interpretable solution so we can observe this phenomenon in a qualitative manner. See Figure 4 for the movies recommended by the different algorithms. Because $k_g = 1$, we are constrained here to have at most one movie from *Adventure*, *Animation* and *Fantasy*. As seen in Figure 4, FANTOM and SAMPLEGREEDY return maximum size solution sets that are both diverse and representative of these genres. On the other hand, Greedy gets stuck choosing a single movie that belongs to all three genres, precluding any other choice of movie from the solution set.

| FANTOM | | | Sample Greedy | | | Repeated Greedy | | Greedy |

Figure 4: Solution sets: the movies in the solution sets for $k_g = 1$ returned by FANTOM, Sample Greedy, Repeated Greedy and Greedy are listed here, along with genre information. The favorite genres (G_u) are in red.

5.2 Streaming Algorithms

In this section, we investigate the performance of our proposed streaming algorithm on two data summarization applications. We demonstrate that by subsampling elements of a stream, it is possible to dramatically reduce the computational cost while providing solutions with similar utility as state-of-the-art algorithms.

For the first task, in Section 5.2.1, we replicate the experiment of Mirzasoleiman et al. [2018] and compare the performance of our algorithm (SAMPLE-STREAMING) with the performance of the algorithm of Mirzasoleiman et al. [2018]. Unfortunately, to allow such a comparison we had to resort to the relatively small datasets that existing algorithms can handle. Despite the small size of these datasets, we could still observe the superiority of our method against this state-of-the-art. In Section 5.2.2, we investigate the scalability of our algorithm to larger datasets in a location summarization task where the other streaming algorithms are not applicable due to their larger computational cost.

5.2.1 Video Summarization.

We evaluate the performance of SAMPLE-STREAMING on a video summarization task and compare it with SEQDPP [Gong et al., 2014][4] and LOCAL-SEARCH [Mirzasoleiman et al., 2018].[5] For our experiments, we use the Open Video Project (OVP) and the YouTube datasets, which have 50 and 39 videos, respectively [De Avila et al., 2011].

Determinantal point process (DPP) is a powerful method to capture diversity in datasets [Macchi, 1975, Kulesza and Taskar, 2012]. Let $\mathcal{N} = \{1, 2, \cdots, n\}$ be a ground set of n items. A DPP defines a probability distribution over all subsets of $\mathcal{N}$, distributed as $\Pr[Y = S] = \frac{\det(L_S)}{\det(I+L)}$ for every set $S \subseteq \mathcal{N}$, where L is a positive semidefinite kernel matrix, L_S is the principal submatrix of L indexed by S, and I is the $n \times n$ identity matrix. The most diverse subset of $\mathcal{N}$ is the one with the maximum probability in this distribution. Although finding this set is NP-hard [Ko et al., 1995], the function $f(S) = \log \det(L_S)$ is a non-monotone submodular function [Kulesza and Taskar, 2012].

We follow the experimental setup of [Gong et al., 2014] for extracting frames from videos, finding a linear kernel matrix L and evaluating the quality of produced summaries based on their F-score. Gong et al. [2014] define a sequential DPP, where each video sequence is partitioned into disjoint segments of equal sizes. For selecting a subset S_t from each segment t (i.e., set $\mathcal{P}_t$), a DPP is defined on the union of the frames in this segment and the selected frames S_{t-1} from

---

[4]https://github.com/pujols/Video-summarization
[5]https://github.com/baharanm/non-mon-stream

the previous segment. Therefore, the conditional distribution of S_t is given by, $\Pr[S_t|S_{t-1}] = \frac{\det(L_{S_t \cup S_{t-1}})}{\det(I_t + L)}$, where L is the kernel matrix defined over $\mathcal{P}_t \cup S_{t-1}$, and I_t is a diagonal matrix of the same size as $\mathcal{P}_t \cup S_{t-1}$ in which the elements corresponding to S_{t-1} are zeros and the elements corresponding to $\mathcal{P}_t$ are 1. For the detailed explanation, please refer to [Gong et al., 2014]. In our experiments, we focus on maximizing the non-monotone submodular function $f(S_t) = \log \det(L_{S_t \cup S_{t-1}})$. We remark that this function can take negative values, violating the non-negativity condition required for our theoretical guarantees.

We first compare the objective values (F-scores) of SAMPLE-STREAMING and LOCAL-SEARCH for different segment sizes over YouTube and OVP datasets. In each experiment, the values are normalized to the F-score of summaries generated by SEQDPP. While SEQDPP has the best performance in terms of maximizing the objective value, in Figures 5a and 5b, we observe that both SAMPLE-STREAMING and LOCAL-SEARCH produce summaries with very high qualities. Figure 6 shows the summary produced by our algorithm for OVP video number 60. Mirzasoleiman et al. [2018] showed that their algorithm (LOCAL-SEARCH) runs three orders of magnitude faster than SEQDPP [Gong et al., 2014]. In our experiments (see Figure 5c), we observed that SAMPLE-STREAMING is 40 and 50 times faster than LOCAL-SEARCH for the YouTube and OVP datasets, respectively, which demonstrates the power of subsampling in reducing the computational cost. Note that for different segment sizes the number of frames remains constant; therefore, the time complexities for both SAMPLE-STREAMING and LOCAL-SEARCH do not change.

In the second experiment, we study the effect of imposing different constraints on video summarization task for YouTube video number 106, which is a part of the television series, "Britain's Got Talent". We consider a constraint on the faces which appear in the summary's frames, using the same methods as described by Mirzasoleiman et al. [2018] for face recognition. Note that a frame may contain more than one face. In the first set of constraints, we restrict to 3 the number of appearances in the summary of each face, which results in a 6-matchoid since there are six unique faces in the video. Figure 7(a) shows the summary produced for this task. We also produce a summary limiting the number of frames which contain each judge. Such a constraint forms a 3-matchoid and this summary is shown in Figure 7(b). Finally, Figure 7(c) shows a summary with a constraint allowing only the singer's face.

5.2.2 Location Summarization.

In the second task, we consider a massive ridesharing dataset, for which current state-of-the-art streaming algorithms are infeasible to run. Given a dataset of 504,247 Uber pick ups in Manhattan, New York in April 2014 UberDataset, our goal is to find a set of the most representative locations. This dataset allows us to study the power of subsampling and the effect of p and k (the size of the largest feasible solution) on the performance of our algorithm.

To do so, the entire area of the given pick ups is covered by $m = 166$ overlapping circular regions of radius r (the centers of these regions provided a 1km-cover of all the area, i.e., for each location in the dataset there was at least one center within a distance of 1km from it), and the algorithm was allowed to choose at most ℓ locations out of each one of these regions. One can observe that by using a single matroid for limiting the number of locations chosen within each one of the regions, the above constraint can be expressed as a p-matchoid constraint, where p is the maximum number of regions a single location can belong to (notice that p could be much smaller than the total number m of regions).

In order to find a representative set S, we use the following monotone submodular objective function: $f(S) = \log \det(I + \alpha K_{S,S})$, where the matrix K encodes the similarities between data points, $K_{S,S}$ is the principal sub-matrix of K indexed by S and $\alpha > 0$ is a regularization parameter [Herbrich et al., 2003, Seeger, 2004, Krause and Guestrin, 2005]. The similarity of

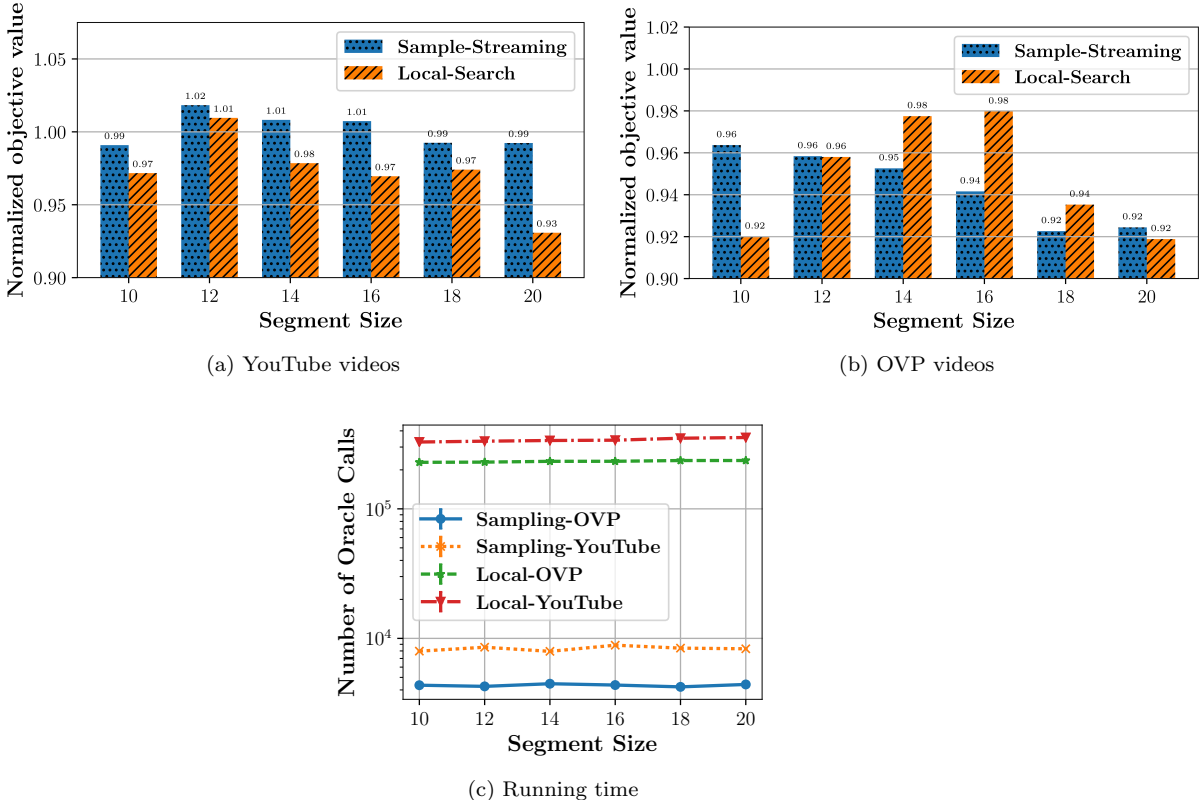(a) YouTube videos

(b) OVP videos

(c) Running time

Figure 5: Comparing the normalized objective value and running time of SAMPLE-STREAMING and LOCAL-SEARCH for different segment sizes.

Figure 6: Summary generated by SAMPLE-STREAMING for OVP video number 60.

two location samples i and j is defined by a Gaussian kernel $K_{i,j} = \exp\left(-d_{i,j}^2/h^2\right)$, where the distance $d_{i,j}$ (in meters) is calculated from the coordinates and h is set to 5000.

In the first location summarization experiment, we set the radius of regions to $r = 1.5$km. In this setting, we observed that a point belongs to at most 7 regions; hence, the constraint is a 7-matchoid. For $\ell = 5$, it took 116 seconds[6] (and 693,717 oracle calls) for our algorithm to find a summary of size $k = 153$. Additionally, for $\ell = 10$ and $\ell = 20$ it took 294 seconds (and 1,306,957 oracle calls) and 1004 seconds (and 2,367,389 oracle calls), respectively, for the algorithm to produce summaries of sizes 301 and 541, respectively.

In the second location summarization experiment, we set the radius of regions to $r = 2.5$km to investigate the performance of our algorithm on p-matchoids with larger values of p. In this

---

[6]In these experiments, we used a machine powered by Intel i5, 3.2 GHz processor and 16 GB of RAM.

Figure 7: Summaries generated by SAMPLE-STREAMING for YouTube video number 106: (a) a 6-matchoid constraint, (b) a 3-matchoid constraint and (c) a partition matroid constraint.

setting, we observed that a point belongs to at most 17 regions, which made the constraint a 17-matchoid. This time, for $\ell = 5$, it took only 35 seconds (and 296,023 oracle calls) for our algorithm to find a summary of size $k = 54$. Additionally, for $\ell = 10$ and $\ell = 20$ it took 80 seconds (and 526,839 oracle calls) and 176 seconds (and 958,549 oracle calls), respectively, for the algorithm to produce summaries of sizes 106 and 198, respectively. As one can observe, our algorithm scales well to larger datasets. Also, for p-matchoids with larger p (which results in a smaller sampling probability q) the performance gets even better.

6   Conclusion

We presented subsampling as an alternative algorithmic technique for constrained submodular maximization. We argued that subsampling provides a unified framework for achieving tight or nearly-tight approximation guarantees for both monotone and non-monotone objectives in several computational settings. To this end, we proposed two algorithms which use the subsampling technique: SAMPLEGREEDY for submodular maximization under a p-extendible system in the offline setting, and SAMPLE-STREAMING for submodular maximization under a p-matchoid constraint in the streaming setting. Our theoretical analysis shows that subsampling can be used to achieve state-of-the-art approximation guarantees for these settings at significantly lower computational cost than previous methods. Our experimental results demonstrate that for a variety of practical problems, algorithms featuring the subsampling technique produce solutions with similar quality to other existing algorithms at a fraction of the computational cost. This work shows that subsampling is a powerful technique which can be used to scale submodular optimization to larger problem instances with more complex constraints.

Acknowledgments.

References

Ashwinkumar Badanidiyuru. Buyback problem - approximate matroid intersection with cancellation costs. In *ICALP*, pages 379–390, 2011.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *KDD*, pages 671–680, 2014.

Ashwinkumar Badanidiyuru, Amin Karbasi, Ehsan Kazemi, and Jan Vondrak. Submodular maximization through barrier functions. In *Advances in Neural Information Processing Systems*, pages 524–534, 2020.

Eric Balkanski, Baharan Mirzasoleiman, Andreas Krause, and Yaron Singer. Learning sparse combinatorial representations via two-stage submodular maximization. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2207–2216, 2016.

Niv Buchbinder and Moran Feldman. Submodular functions maximization problems., 2018.

Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Math. Oper. Res.*, 44(3):988–1005, 2019. URL `https://doi.org/10.1287/moor.2018.0955`.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *SODA*, pages 1433–1452, 2014.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *SODA*, pages 1202–1216, 2015.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing Apples and Oranges: Query Trade-off in Submodular Maximization. *Mathematics of Operations Research*, 2016.

Niv Buchbinder, Moran Feldman, Yuval Filmus, and Mohit Garg. Online submodular maximization: Beating 1/2 made simple. In *IPCO*, pages 101–114, 2019.

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

E. Candés and B. Recht. Exact matrix completion via convex optimization. In *Foundations of Computational Mathematics*, 2008.

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

Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *ICALP*, pages 318–330, 2015.

Sandra Eliza Fontes De Avila, Ana Paula Brandão Lopes, Antonio da Luz Jr, and Arnaldo de Albuquerque Araújo. VSUMM: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68, 2011.

Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *international conference on Knowledge discovery and data mining (KDD)*, pages 289–298, 2009.

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

Alina Ene and Huy L. Nguyen. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. *CoRR*, abs/1811.07464, 2018. URL `http://arxiv.org/abs/1811.07464`.

Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, pages 570–579, 2011a.

Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems - (extended abstract). In *ESA*, pages 784–798, 2011b.

Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is good: Near-optimal submodular maximization via greedy optimization. In *COLT*, pages 758–784, 2017.

Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 730–740, 2018.

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

Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *international conference on Knowledge discovery and data mining (KDD)*, 2010.

Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *NIPS*, pages 2069–2077, 2014.

Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-Optimal Sensor Placements in Gaussian Processes. In *International Conference on Machine Learning (ICML)*, 2005.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms. In *WINE*, pages 246–257, 2010.

Ran Haba, Ehsan Kazemi, Moran Feldman, and Amin Karbasi. Streaming submodular maximization under a k-set system constraint. In *International Conference on Machine Learning*, 2020.

Trevor Hastie, Rahul Mazumder, Jason D. Lee, and Rexa Zadeh. Matrix completion and low-rank svd via fast alternating least squares. In *Journal of Machine Learning Research*, 2015.

Ralf Herbrich, Neil D Lawrence, and Matthias Seeger. Fast sparse gaussian process methods: The informative vector machine. In *Advances in neural information processing systems*, pages 625–632, 2003.

Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints. In *ICML*, pages 2549–2558, 2018.

Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular Streaming in All Its Glory: Tight Approximation, Minimum Memory and Low Adaptive Complexity. In *International Conference on Machine Learning (ICML)*, pages 3311–3320, 2019.

Ehsan Kazemi, Shervin Minaee, Moran Feldman, and Amin Karbasi. Regularized submodular maximization at scale. *arXiv preprint arXiv:2002.03503*, 2020.

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

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

Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

Nitish Korula, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM J. Comput.*, 47(3):1056–1086, 2018. URL https://doi.org/10.1137/15M1051142.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 324–331, 2005.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3), 2012.

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing non-monotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.*, 23(4):2053–2078, 2010a.

Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular Maximization over Multiple Matroids via Generalized Exchange Properties. *Math. Oper. Res.*, 35(4):795–806, 2010b.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *international conference on Knowledge discovery and data mining (KDD)*, pages 420–429, 2007.

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

Hui Lin and Jeff A. Bilmes. A Class of Submodular Functions for Document Summarization. In *HLT*, pages 510–520, 2011.

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

Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

Julián Mestre. Greedy in Approximation Algorithms. In *European Symposium on Algorithms (ESA)*, pages 528–539. 2006.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *AAAI Conference on Artificial Intelligence*, pages 1812–1818, 2015.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast Constrained Submodular Maximization: Personalized Data Summarization. In *ICML*, pages 1358–1367, 2016a.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed Submodular Maximization. *Journal of Machine Learning Research*, 17:238:1–238:44, 2016b.

Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming Non-Monotone Submodular Maximization: Personalized Video Summarization on the Fly. In *AAAI Conference on Artificial Intelligence*, 2018.

Marko Mitrovic, Mark Bun, Andreas Krause, and Amin Karbasi. Differentially private submodular maximization: Data summarization in disguise. In *International Conference on Machine Learning*, pages 2478–2487. PMLR, 2017.

Marko Mitrovic, Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Data Summarization at Scale: A Two-Stage Submodular Approach. In *ICML*, pages 3593–3602, 2018.

Marko Mitrovic, Ehsan Kazemi, Moran Feldman, Andreas Krause, and Amin Karbasi. Adaptive sequence submodularity. In *Advances in Neural Information Processing Systems*, pages 5352–5363, 2019.

Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Conditional gradient method for stochastic submodular maximization: Closing the gap. In *AISTATS*, pages 1886–1895, 2018.

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

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

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, pages 1098–1116, 2011.

Mehraveh Salehi, Amin Karbasi, Dustin Scheinost, and R. Todd Constable. A Submodular Approach to Create Individualized Parcellations of the Human Brain. In *MICCAI*, pages 478–485, 2017.

Matthias Seeger. Greedy forward selection in the informative vector machine. Technical report, Technical report, University of California at Berkeley, 2004.

Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In *International Conference on Information and Knowledge Management, (CIKM)*, pages 754–763, 2012.

Serban Stan, Morteza Zadimoghaddam, Andreas Krause, and Amin Karbasi. Probabilistic Submodular Maximization in Sub-Linear Time. In *International Conference on Machine Learning (ICML)*, pages 3241–3250, 2017.

Ehsan Tohidi, Rouhollah Amiri, Mario Coutino, David Gesbert, Geert Leus, and Amin Karbasi. Submodularity in action: From machine learning to signal processing applications. *IEEE Signal Processing Magazine*, 37(5):120–133, 2020.

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

UberDataset. Uber pickups in new york city, 2014. URL `https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city`.

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

Justin Ward. A (k+3)/2-approximation algorithm for monotone submodular k-set packing and general k-exchange systems. In *STACS*, pages 42–53, 2012.