

Streaming Algorithms for Submodular Function Maximization

Chandra Chekuri* Shalmoli Gupta† Kent Quanrud‡

Dept. of Computer Science, Univ. of Illinois, Urbana IL 61801, USA
 {chekuri,sgupta49,quanrud2}@illinois.edu

May 1, 2015

We consider the problem of maximizing a nonnegative submodular set function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$ subject to a p -matchoid constraint in the single-pass streaming setting. Previous work in this context has considered streaming algorithms for modular functions and monotone submodular functions. The main result is for submodular functions that are *non-monotone*. We describe deterministic and randomized algorithms that obtain a $\Omega(\frac{1}{p})$ -approximation using $O(k \log k)$ -space, where k is an upper bound on the cardinality of the desired set. The model assumes value oracle access to f and membership oracles for the matroids defining the p -matchoid constraint.

1. Introduction

Let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ be a set function defined over a ground set \mathcal{N} . f is *submodular* if it exhibits decreasing marginal values in the following sense: if $e \in \mathcal{N}$ is any element, and $A, B \subseteq \mathcal{N}$ with $A \subseteq B$ are any two nested sets, then $f(A + e) - f(A) \geq f(B + e) - f(B)$. The gap $f(A + e) - f(A)$ is called the *marginal value* of e with respect to f and A , and denoted $f_A(e)$. An equivalent characterization for submodular functions is that for any two sets $A, B \subseteq \mathcal{N}$, $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$.

Submodular functions play a fundamental role in classical combinatorial optimization where rank functions of matroids, edge cuts, coverage, and others are instances of submodular

*Work on this paper supported in part by NSF grant CCF-1319376.

†Work on this paper supported in part by NSF grant CCF-1319376.

‡Work on this paper supported in part by NSF grants CCF-1319376, CCF-1421231, and CCF-1217462.

functions (see [Sch03, Fuj05]). More recently, there is a large interest in constrained submodular function optimization driven both by theoretical progress and a variety of applications in computer science. The needs of the applications, and in particular the sheer bulk of large data sets, have brought into focus the development of fast algorithms for submodular optimization. Recent work on the theoretical side include the development of faster worst-case approximation algorithms in the traditional sequential model of computation [BV14, IJB13, CJV15], algorithms in the streaming model [BMKK14, CK14] as well as in the map-reduce model of computation [KMVV13].

In this paper we consider constrained submodular function *maximization*. The goal is to find $\max_{S \in \mathcal{I}} f(S)$ where $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a *downward-closed* family of sets; i.e., $A \in \mathcal{I}$ and $B \subseteq A$ implies $B \in \mathcal{I}$. \mathcal{I} is also called an *independence family* and any set $A \in \mathcal{I}$ is called an *independent set*. Submodular maximization under various independence constraints has been extensively studied in the literature. The problem can be easily seen to be NP-hard even for a simple cardinality constraint as it encompasses standard NP-hard problems like the Max- k -cover problem. Constrained submodular maximization has found several new applications in recent years. Some of these include data summarization [LB11, SSSJ12, DKR13], influence maximization in social networks [KKT03, CWY09, CWW10, GBL11, SS13], generalized assignment [CCPV07], mechanism design [BIK07], and network monitoring [LKG⁺07].

In some of these applications, the amount of data involved is much larger than the main memory capacity of individual computers. This motivates the design of space-efficient algorithms which can process the data in *streaming* fashion, where only a small fraction of the data is kept in memory at any point. There has been some recent work on submodular function maximization in the streaming model, focused on *monotone* functions (i.e. $f(A) \leq f(B)$, whenever $A \subseteq B$). This assumption is restrictive from both a theoretical and practical point of view.

In this paper we present streaming algorithms for non-monotone submodular function maximization subject to various combinatorial constraints, the most general being a *p-matchoid*. *p-matchoid*'s generalize many basic combinatorial constraints such as the cardinality constraint, the intersection of p matroids, and matchings in graphs and hyper-graphs. A formal definition of a *p-matchoid* is given in Section 2. We consider the abstract *p-matchoid* constraint for theoretical reasons, and most constraints in practice should be simpler. We explicitly consider the cardinality constraint and obtain an improved bound.

We now describe the problem formally. We are presented a groundset of elements $\mathcal{N} = \{e_1, e_2, \dots, e_n\}$, with no assumption made on the order or the size of the datastream. The goal is to select an independent set $S \subseteq \mathcal{N}$ (where

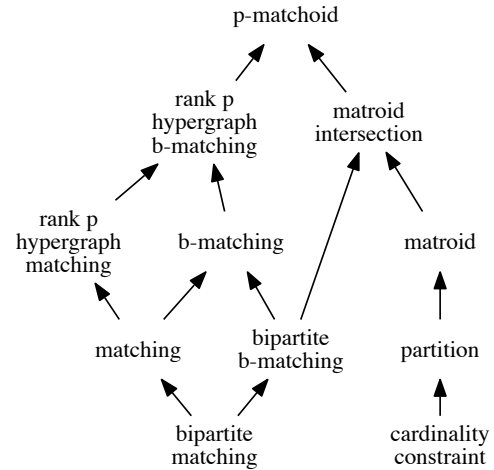


Figure 1: Hierarchy of set systems

independence is defined by the [p-matchoid](#)), which maximizes a nonnegative submodular function f while using as little space as possible. We make the following assumptions: (i) the function f is available via a value oracle, that takes as input a set $S \subseteq \mathcal{N}$ and returns the value $f(S)$; (ii) the independence family \mathcal{I} is available via a membership oracle with some additional information needed in the [p-matchoid](#) setting; and (iii) the constraints specify explicitly, and a priori, an upper bound k on the number of elements to be chosen. We discuss these in turn. The availability of a value oracle for f is a reasonable and standard assumption in the sequential model of computation, but needs some justification in restrictive models of computation such as streaming where the goal is to store at any point of time only a small subset of the elements of \mathcal{N} . Can $f(S)$ be evaluated without having access to all of \mathcal{N} ? This of course depends on f . [BMKK14] gives several examples of interesting and useful functions where this is indeed possible. The second assumption is also reasonable if, as we remarked, the [p-matchoid](#) constraint is in practice going to be a simple one that combines basic matroids such as cardinality, partition and laminar matroid constraints that can be specified compactly and implicitly. Finally, the third assumption is guided by the fact that an abstract model of constraints can in principle lead to every element being chosen. In many applications the goal is to select a small and important subset of elements from a much larger set; and it is therefore reasonable to expect knowledge of an upper bound on how many can be chosen. Submodular set functions are ubiquitous and arise explicitly and implicitly in a variety of settings. The model we consider in this paper may not be useful directly in some important scenarios of interest. Nevertheless, the ideas underlying the analysis in the streaming model that we consider here may still be useful in speeding up existing algorithms and/or reduce their space usage.

As is typical for streaming algorithms, we measure performance in four basic dimensions: (i) the approximation ratio $f(S)/\text{OPT}$, where S is the output of the algorithm and OPT is the value of an optimal solution; (ii) the space usage of the algorithm; (iii) the update time or the time required to process each stream element; and (iv) the number of passes the algorithm makes over the data stream.

Our results. We develop randomized and deterministic algorithms that yield an $\Omega(1/p)$ -approximation for maximizing a non-negative submodular function under a [p-matchoid](#) constraint in the one-pass streaming setting. The space usage is $O(k \log k)$, essentially matching recent algorithms for the simpler setting of maximizing a monotone submodular function subject to a cardinality constraint [BMKK14]. The randomized algorithm achieves better constants than the deterministic algorithm. As far as we are aware, we present the first streaming algorithms for non-monotone submodular function maximization under constraints beyond cardinality. We give an improved bound of $\frac{1-\epsilon}{2+\epsilon}$ for the cardinality constraint. For the monotone case our bounds match those of Chakrabarti and Kale [CK14] for a single pass; we give a self-contained algorithm and analysis. [Table 1](#) summarizes our results for a variety of constraints.

	offline		streaming	
constraint	monotone	nonnegative	monotone	nonnegative
cardinality	$1 - 1/e$ [NWF78]	$1/e + .004$ [BFNS14]	$\frac{1-\epsilon}{2}$ [BMKK14]	$\frac{1-\epsilon}{2+\epsilon}$ (R,★)
matroid	$1 - 1/e$ (R) [CCPV11]	$\frac{1-\epsilon}{e}$ (R) [FNS11]	$1/4$ [CK14]	$\frac{1-\epsilon}{4+\epsilon}$ (R,★)
matchings	$\frac{1}{2+\epsilon}$ [FNSW11]	$\frac{1}{4+\epsilon}$ [FNSW11]	$4/31$ [CK14]	$\frac{1-\epsilon}{12+\epsilon}$ (R,★)
b -matchings	$\frac{1}{2+\epsilon}$ [FNSW11]	$\frac{1}{4+\epsilon}$ [FNSW11]	$1/8$ (★)	$\frac{1-\epsilon}{12+\epsilon}$ (R,★)
rank p hypergraph b -matching	$\frac{1}{p+\epsilon}$ (R) [FNSW11]	$\frac{p-1}{p^2+\epsilon}$ [FNSW11]	$1/4p$ (★)	$\frac{(1-\epsilon)(p-1)}{5p^2-4p+\epsilon}$ (R,★)
intersection of p matroids	$\frac{1}{p+\epsilon}$ [LSV10]	$\frac{p-1}{p^2+(p-1)\epsilon}$ [LSV10]	$1/4p$ [CK14]	$\frac{(1-\epsilon)(p-1)}{5p^2-4p}$ (R,★)
p -matchoids	$\frac{1}{p+1}$ [FNW78, CCPV11]	$\frac{(1-\epsilon)(2-o(1))}{ep}$ (R) [FNS11, CVZ11]	$1/4p$ (★)	$\frac{(1-\epsilon)(2-o(1))}{(8+\epsilon)p}$ (R,★)

Table 1: Best known approximation bounds for submodular maximization. Bounds for randomized algorithms that hold in expectation are marked (R). For hypergraph b -matchings and matroid intersection, p is fixed. In the results for p -matchoids, $o(1)$ goes to zero as p increases. New bounds attained in this paper are marked (★). All new bounds except for the cardinality constraint are the first bounds for their class. The best previous bound for the cardinality constraint is about .0893, by [BFS15].

A brief overview of techniques. Streaming algorithms for constrained modular and submodular function optimization are usually clever variations of the greedy algorithm, which picks elements in iterations to maximize the gain in each iteration locally while maintaining feasibility. For monotone functions, in the offline setting, greedy gives a $1/(p+1)$ -approximation for the p -matchoid constraint and a $(1-1/e)$ -approximation for the cardinality constraint [FNW78]. The offline greedy algorithm cannot be directly implemented in streams, but we outline two different strategies that are still greedy in spirit. For the cardinality constraint, Badanidiyuru *et al.* [BMKK14] designed an algorithm that adds an element to its running solution S only if the marginal gain is at least a threshold of about $\text{OPT}/2k$. Although the quantity $\text{OPT}/2k$ is not known a priori, they show that it lies in a small and identifiable range, and can be approximated with $O(\log k)$ well-spaced guesses. The algorithm then maintains $O(\log k)$ solutions in parallel, one for each guess. Another strategy from Chakrabarti and Kale [CK14], based on previous work for matchings [FKM⁺05, McG05] and matroid constraints [Bad11] with modular weights, will consider deleting elements from S when adding a new element to S is infeasible. More specifically, when a new element e is encountered, the algorithm finds a subset $C \subseteq S$ such that $(S \setminus C) + e$ is feasible, and compare the gain $f((S \setminus C) + e) - f(S)$ to a quantity representing the value that C adds to S . In the modular case, this may be the sum of weights of elements in C ; for monotone

submodular functions, Chakrabarti and Kale used marginal values, fixed for each element when the element is added to S , as proxy weights instead.

The non-monotone case is harder because marginal values can be negative even when f is non-negative. The natural greedy algorithm fails for even the simple cardinality constraint, and the best offline algorithms for nonnegative submodular maximization are uniformly weaker (see Table 1). To this end, we adapt techniques from the recent work of Buchbinder *et al.* [BFNS14] in our randomized algorithm, and techniques from Gupta *et al.* [GRST10] for the deterministic version. Buchbinder *et al.* randomized the standard greedy algorithm (for cardinality) by repeatedly gathering the top (say) k remaining elements, and then randomly picking only one of them. We adapt this to the greedy setting by adding the top elements to a buffer B as they appear in the stream, and randomly adding an element from B to S only when B fills up. What remains of B at the end of the stream is post-processed by an offline algorithm. Gupta *et al.* gave a framework for adapting any monotone submodular maximization algorithm to nonnegative submodular functions, by first running the algorithm once to generate one independent set S_1 , then running the algorithm again on the complement of S_1 to generate a second set S_2 , and running an unconstrained maximization algorithm on S_1 to produce a third set S_3 , finally returning the best of S_1 , S_2 , and S_3 . Our deterministic streaming algorithm is a natural adaptation, piping the rejected elements of one instance of a streaming algorithm directly into a second instance of the same algorithm, and post-processing all the elements taken by the first streaming instance. Both of our algorithms require that we limit the number of elements ever added to S , which then limits the size of the input for the post-processor. This limit is enforced by the idea of additive thresholds from [BMKK14] and a simple but subtle notion of value that ensures the properties we desire.

Related work. There is substantial literature on constrained submodular function optimization, and we only give a quick overview. Many of the basic problems are NP-Hard, so we will mainly focus on the development of approximation algorithms. The (offline) problem $\max_{S \in \mathcal{I}} f(S)$ for various constraints has been extensively explored starting with the early work of Fisher, Nemhauser, Wolsey on greedy and local search algorithms [NWF78, FNW78]. Recent work has obtained many new and powerful results based on a variety of methods including variants of greedy [GRST10, BFNS14, BFNS12], local search [LMNS10, LSV10, FW14], and the multilinear relaxation [CCPV11, KST13, BKNS12, CVZ11]. Monotone submodular functions admit better bounds than non-monotone functions (see Table 1). For a p -matchoid constraint, which is our primary consideration, an $\Omega(1/p)$ -approximation can be obtained for non-negative functions. Recent work has also obtained new lower bounds on the approximation ratio achievable in the oracle model via the so-called symmetry gap technique [Von13]; this also yields lower bounds in the standard computational models [DV12].

Streaming algorithms for submodular functions are a very recent phenomenon with algorithms developed recently for monotone submodular functions [BMKK14, CK14]. [BMKK14]

gives a $1/2 - \epsilon$ approximation for monotone functions under cardinality constraint using $O(k \log k/\epsilon)$ space. [CK14] focuses on more general constraints like intersections of p -matroids and rank p hypergraphs, giving an approximation of $1/4p$ using a single pass. Their algorithm extends to multiple passes, with an approximation bound of $1/(p + 1 + \epsilon)$ with $O(\epsilon^{-3} \log p)$ passes. The main focus of [KMVV13] is on the map-reduce model although they claim some streaming results as well.

Related to the streaming models are two *online* models where elements arrive in an online fashion and the algorithm is required to maintain a feasible solution S at all times; each element on arrival has to be processed and any element which is discarded from S at any time cannot be added back later. Strong lower bounds can be shown in this model and two relaxations have been considered. In the *secretary model*, the elements arrive according to a random permutation of the ground set and an element added to S cannot be discarded later. In the secretary model, constant factor algorithms are known for the cardinality constraint and some special cases of a single matroid constraint [GRST10, BHZ13]. These algorithms assume the stream is randomly ordered and their performance degrades badly against adversarial streams; the best competitive ratio for a single general matroid is $O(\log k)$ (where k is the rank of the matroid). Recently, Buchbinder *et al.* [BFS15] considered a different relaxation of the online model where *preemptions* are allowed: elements added to S can be discarded later. Algorithms in the preemptive model are usually streaming algorithms, but the converse is not true (although the one-pass algorithms in [CK14] are preemptive). For instance, the algorithm in [BMKK14] maintains multiple feasible solutions and our algorithms maintain a buffer of elements neither accepted nor rejected. The space requirement of an algorithm in the online model is not necessarily constrained since in principle an algorithm is allowed to keep track of all the past elements seen so far. The main result in [BFS15], as it pertains to this work, is a randomized 0.0893-competitive algorithm for cardinality constraints using $O(k)$ -space. As Table 1 shows, we obtain a $(1 - \epsilon)/(2 + \epsilon)$ -competitive algorithm for this case using $O(k \log k/\epsilon^2)$ -space.

Paper organization. Section 2 reviews combinatorial definitions and introduces the notion of incremental values. Section 3 analyzes an algorithm that works for monotone submodular functions, and Section 4 adapts this algorithm to the non-monotone case. In Section 5, we give a deterministic streaming algorithm with slightly weaker guarantees.

2. Preliminaries

Matroids. A *matroid* is a finite set system $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, where \mathcal{N} is a set and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is a family of subsets such that: (i) $\emptyset \in \mathcal{I}$, (ii) If $A \subseteq B \subseteq \mathcal{N}$, and $B \in \mathcal{I}$, then $A \in \mathcal{I}$, (iii) If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is an element $b \in B \setminus A$ such that $A + b \in \mathcal{I}$. In a matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, \mathcal{N} is called the *ground set* and the members of \mathcal{I} are called *independent sets* of the matroid. The bases of \mathcal{M} share a common cardinality, called the *rank* of \mathcal{M} .

Matchoids. Let $\mathcal{M}_1 = (\mathcal{N}_1, \mathcal{I}_1), \dots, \mathcal{M}_q = (\mathcal{N}_q, \mathcal{I}_q)$ be q matroids over overlapping ground-sets. Let $\mathcal{N} = \mathcal{N}_1 \cup \dots \cup \mathcal{N}_q$ and $\mathcal{I} = \{S \subseteq \mathcal{N} : S \cap \mathcal{N}_\ell \in \mathcal{I}_\ell \text{ for all } \ell\}$. The finite set system $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$ is a *p -matchoid* if for every element $e \in \mathcal{N}$, e is a member of \mathcal{N}_ℓ for at most p indices $\ell \in [q]$. p -matchoids generalize matchings and intersections of matroids, among others (see Figure 1).

Maximizing submodular functions under a p -matchoid constraint. Let \mathcal{N} be a set of elements, $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}_{\geq 0}$ a nonnegative submodular function on \mathcal{N} , and $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$ a p -matchoid for some integer p . We want to approximate $\text{OPT} = \max_{S \in \mathcal{I}} f(S)$. There are several polynomial-time approximation algorithms that give an $\Omega(1/p)$ -approximation for this problem, with better bounds for simpler constraints (see Table 1). These algorithms are used as a black box called **Offline**, with approximation ratio denoted by γ_p : if **Offline** returns $S \in \mathcal{I}$, then $\mathbf{E}[f(S)] \geq \gamma_p \text{OPT}$ (possibly without expectation, if **Offline** is deterministic).

Incremental Value. Let \mathcal{N} be a ground set, and let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ be a submodular function. For a set $S \subseteq \mathcal{N}$ and an element $e \in S$, what is the value that e adds to S ? One idea is to take the margin $f_{S-e}(e) = f(S) - f(S - e)$ of adding e to $S - e$. However, because f is not necessarily modular, we can only say that $\sum_{e \in S} f_{S-e}(e) \leq f(S)$ without equality. It is natural to ask for a different notion of value where the values of the parts sum to the value of the whole.

Let \mathcal{N} be an *ordered* set and $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ be a set function. For a set $S \subseteq \mathcal{N}$ and element $e \in \mathcal{N}$, the *incremental value* of e in S , denoted $\nu(f, S, e)$, is defined as

$$\nu(f, S, e) = f_{S'}(e), \text{ where } S' = \{s \in S : s < e\}.$$

The key point of incremental values is that they capture the entire value of a set. The following holds for *any* set function.

Lemma 1. *Let \mathcal{N} be an ordered set, $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ a set function, and $S \subseteq \mathcal{N}$ a set. Then $f(S) = \sum_{e \in S} \nu(f, S, e)$.*

Proof. Enumerate $S = \{e_1, \dots, e_\ell\}$ in order, and let $S_i = \{e_1, \dots, e_i\}$ denote the first i elements in S . We have,

$$\sum_{e_i \in S} \nu(f, S, e_i) = \sum_{e_i \in S} f_{S_{i-1}}(e_i) = f(S).$$

□

When f is submodular, we have decreasing incremental values analogous (and closely related) to decreasing marginal returns of submodular function.

Lemma 2. *Let $S \subseteq T \subseteq \mathcal{N}$ be two nested subsets of an ordered set \mathcal{N} , let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ be submodular, and let $e \in \mathcal{N}$. Then $\nu(f, T, e) \leq \nu(f, S, e)$.*

Proof. Let $S' = \{s \in S : s < e\}$ and $T' = \{t \in T : t < e\}$. Since $S \subseteq T$, clearly $S' \subseteq T'$. We have,

$$\nu(f, T, e) = f_{T'}(e) \leq f_{S'}(e) = \nu(f, S, e)$$

where the inequality follows by submodularity. \square

The following is also an easy consequence of submodularity.

Lemma 3. *Let \mathcal{N} be an ordered set of elements, let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ be a submodular function, $S, Z \subseteq \mathcal{N}$ two sets, and $e \in S$. Then $\nu(f_Z, S, e) \leq \nu(f, Z \cup S, e)$.*

Proof. Let $Z' = \{z \in Z : z < e\}$ and $S' = \{s \in S : s < e\}$. By submodularity, we have,

$$\nu(f_Z, S, e) = f_{Z \cup S'}(e) \leq f_{Z' \cup S'}(e) = \nu(f, Z \cup S, e).$$

\square

3. Streaming Greedy

Let $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$ be a p -matchoid and f a submodular function. The elements of \mathcal{N} are presented in a stream, and we order \mathcal{N} by order of appearance. We assume value oracle access to f , that given $S \subseteq \mathcal{N}$, returns the value $f(S)$. We also assume membership oracles for each of the q matroids defining \mathcal{M}^p : given $S \subseteq \mathcal{N}_\ell$, there is an oracle for \mathcal{M}_ℓ that returns whether or not $S \in \mathcal{I}_\ell$.

We first present a deterministic streaming algorithm **Streaming-Greedy** that yields an $\Omega(1/p)$ -approximation for monotone submodular functions, but performs poorly for non-monotone functions. The primary motivation in presenting **Streaming-Greedy** is as a building block for a randomized algorithm **Randomized-Streaming-Greedy** presented in Section 4, and a deterministic algorithm **Iterated-Streaming-Greedy** presented in Section 5. The analysis for these algorithms relies crucially on properties of **Streaming-Greedy**.

Streaming-Greedy maintains an independent set $S \in \mathcal{I}$; as an element arrives in the stream, it is either discarded or added to S in exchange for a well-chosen subset of S . The threshold for exchanging is tuned by two nonnegative parameters α and β . At the end of the stream, **Streaming-Greedy** outputs S .

The overall strategy is similar to previous algorithms developed for matchings [FKM⁺05, McG05] and intersections of matroids [Bad11] when f is modular, and generalized by [CK14] to monotone submodular functions. There are two main differences. One is the use of the additive threshold α . The second is the use of the incremental value ν . By using incremental value, the value of an element $e \in S$ is not fixed statically when e is first added to S , and increases over time as other elements are dropped from S . These two seemingly minor modifications are crucial to the eventual algorithms for non-monotone functions.

We remark that **Streaming-Greedy** also fits the online preemptive model.


```

Streaming-Greedy( $\alpha, \beta$ )
 $S \leftarrow \emptyset$ 
while (stream is not empty)
   $e \leftarrow$  next element in the stream
   $C \leftarrow$  Exchange-Candidates( $S, e$ )
  //  $C$  satisfies  $S - C + e \in \mathcal{I}$ 
  if  $f_S(e) \geq \alpha + (1 + \beta) \sum_{c \in C} \nu(f, S, c)$ 
     $S \leftarrow S \setminus C + e$ 
  end while
return  $S$ 

```

```

Exchange-Candidates( $S, e$ )
 $C \leftarrow \emptyset$ 
for  $\ell = 1, \dots, q$ 
  if  $e \in \mathcal{N}_\ell$  and  $(S + e) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell$ 
     $S_\ell = S \cap \mathcal{N}_\ell$ 
     $X \leftarrow \{s \in S_\ell : (S_\ell - s + e) \in \mathcal{I}_\ell\}$ 
    //  $X + e$  is a circuit
     $c_\ell \leftarrow \arg \min_{x \in X} \nu(f, S, x)$ 
     $C \leftarrow C + c_\ell$ 
  end if
end for
return  $C$ 

```

Outline of the analysis: Let $T \in \mathcal{I}$ be some fixed feasible set (we can think of T as an optimum set). In the offline analysis of the standard greedy algorithm one can show that $f(S \cup T) \leq (p+1)f(S)$, where S is the output of greedy; for the monotone case this implies that $f(S) \geq f(T)/(p+1)$. The analysis here hinges on the fact that each element of $T \setminus S$ is available to greedy when it chooses each element. In the streaming setting, this is no longer feasible and hence the need to remove elements in favor of new high-value elements. To relate \tilde{S} , the final output, to T , we consider U , the set of all elements ever added to S . The analysis proceeds in two steps.

First, we upper bound $f(U)$ by $f(\tilde{S})$ as $f(U) \leq (1 + \frac{1}{\beta}) \cdot f(\tilde{S}) - \frac{\alpha}{\beta}|U|$. Second, we upper bound $f(T \cup U)$ as $f(T \cup U) \leq k\alpha + \frac{(1+\beta)^2}{\beta} \cdot p \cdot f(\tilde{S})$. For $\alpha = 0$, we obtain $f(T \cup U) \leq \frac{(1+\beta)^2}{\beta} \cdot p \cdot f(\tilde{S})$, which yields $f(T) \leq 4pf(\tilde{S})$ when f is monotone (for $\beta = 1$); this gives the same bound as [CK14]. The crucial difference is that we are able to prove an upper bound on the size of U , namely, $|U| \leq \text{OPT}/\alpha$; hence, if we choose the threshold α to be $c\text{OPT}/k$ for some parameter c we have $|U| \leq k/c$. This will play a critical role in analyzing the non-monotone case in the subsequent sections that use **Streaming-Greedy** as a black box. The upper bound on $|U|$ is achieved by the definition of ν and the threshold α ; we stress that this is not as obvious as it may seem because the function f can be non-monotone and the marginal values can be negative.

Some notation for the analysis:

- \tilde{S} denotes the final set returned by **Streaming-Greedy**.
- For each element $e \in \mathcal{N}$, S_e^- denotes the set held by S just before e is processed, and S_e^+ the set held by S just after e is processed. Note that if e is rejected, then $S_e^- = S_e^+$.
- U denotes the set of all elements added to S at any point in the stream. Note that $U = \bigcup_{e \in \mathcal{N}} S_e^+$.

- For $e \in \mathcal{N}$, $C_e = \text{Exchange-Candidates}(S_e^-, e) \subseteq S_e^-$ denotes the set of elements that **Streaming-Greedy** considers exchanging for e . Observe that $\{C_u, u \in U\}$ forms a partition of $U \setminus \tilde{S}$.
- For $e \in \mathcal{N}$, $\delta_e = f(S_e^+) - f(S_e^-)$ denotes the *gain* from processing e . Note that $\delta_e = 0$ for all $e \in \mathcal{N} \setminus U$, and $\sum_{e \in \mathcal{N}} \delta_e = f(\tilde{S})$.

3.1. Relating $f(U)$ to $f(\tilde{S})$

When **Streaming-Greedy** adds an element e to S , it only compares the marginal $f_S(e)$ to the incremental values in its exchange candidates C , and does not directly evaluate the gain $f(S \setminus C + e) - f(S)$ realized by the exchange. The first lemma derives a lower bound for this gain.

Lemma 4. *Let $e \in U$ be added to S when processed by **Streaming-Greedy**. Then*

$$\delta_e \geq \alpha + \beta \sum_{c \in C_e} \nu(f, S_e^-, c)$$

Proof. Since e replaced C_e , by design of **Streaming-Greedy**, we have,

$$f_{S_e^-}(e) \geq \alpha + (1 + \beta) \sum_{c \in C_e} \nu(f, S_e^-, c),$$

which, after rearranging, gives

$$f_{S_e^-}(e) - \sum_{c \in C_e} \nu(f, S_e^-, c) \geq \alpha + \beta \sum_{c \in C_e} \nu(f, S_e^-, c).$$

To prove the lemma, it suffices to show that

$$\delta_e \geq f_{S_e^-}(e) - \sum_{c \in C_e} \nu(f, S_e^-, c).$$

Let $Z = S_e^- \setminus C_e = S_e^+ - e$. Note that $S_e^+ = Z + e$ and $S_e^- = Z \cup C_e$. We have,

$$\begin{aligned} \delta_e &= f(Z + e) - f(Z \cup C_e) \\ &= f_Z(e) - f_Z(C_e) && \text{by adding and subtracting } f(Z), \\ &\geq f_{S_e^-}(e) - f_Z(C_e) && \text{by submodularity of } f, \\ &= f_{S_e^-}(e) - \sum_{c \in C_e} \nu(f_Z, C_e, c), && \text{by definition of } \nu. \\ &\geq f_{S_e^-}(e) - \sum_{c \in C_e} \nu(f, S_e^-, c) && \text{by Lemma 3 and } S_e^- = Z \cup C_e, \end{aligned}$$

as desired. □

One basic consequence of [Lemma 4](#) is that every element in U adds a positive and significant amount α to the value to S . This will be crucial later, when taking α proportional to OPT limits the size of U .

Lemma 5. *For all $e \in U$, $\delta_e \geq \alpha$ and hence $|U| \leq \text{OPT}/\alpha$.*

Proof. We claim that at any point in the algorithm, $\nu(f, S, e) \geq 0$ for all $e \in S$, from which the lemma follows [Lemma 4](#) immediately.

When an element $e \in U$ is added to S , it has incremental value

$$\nu(f, S_e^+, e) = f_{S_e^+ - e}(e) \geq f_{S_e^-}(e) \geq \alpha + (1 + \beta) \sum_{c \in C_e} \nu(f, S, c).$$

As the algorithm continues, elements preceding e in S may be deleted while elements after e are added, so $\nu(f, S, e)$ can only increase with time. \square

Returning to the original task of bounding $f(U)$, the difference $U \setminus \tilde{S}$ is the set of deleted elements, and the only handle on these elements is their incremental value at the point of deletion. For $d \in U \setminus \tilde{S}$, let $e(d)$ be the element that d was exchanged for; that is, $e(d) > d$ and $d \in S_{e(d)}^- \setminus S_{e(d)}^+ = C_{e(d)}$. For deleted elements $d \in U \setminus \tilde{S}$, the *exit value* $\chi(d)$ of d is the incremental value of d evaluated when d is removed from S , defined formally as

$$\chi(d) = \nu(f, S_{e(d)}^-, d).$$

Here we bound the sum of exit values of $U \setminus \tilde{S}$.

Lemma 6.

$$\sum_{d \in U \setminus \tilde{S}} \chi(d) \leq \frac{1}{\beta} \cdot (f(\tilde{S}) - \alpha|U|).$$

Proof. Indeed,

$$\begin{aligned} \sum_{d \in U \setminus \tilde{S}} \chi(d) &= \sum_{u \in U} \sum_{d \in C_u} \chi(d) && \text{since } \{C_u : u \in U\} \text{ partitions } U \setminus \tilde{S}, \\ &\leq \sum_{u \in U} \frac{1}{\beta} \cdot (\delta_u - \alpha) && \text{by [Lemma 4](#),} \\ &= \frac{1}{\beta} \cdot (f(\tilde{S}) - \alpha|U|) \end{aligned}$$

\square

Now we bound $f(U)$.

Lemma 7.

$$f(U) \leq \left(1 + \frac{1}{\beta}\right) \cdot f(\tilde{S}) - \frac{\alpha}{\beta}|U|.$$

Proof. Recall, for each element $d \in U \setminus \tilde{S}$, $e(d)$ denotes the element added in exchange of d . We have,

$$\begin{aligned} f(U) - f(\tilde{S}) &= f_{\tilde{S}}(U) = \sum_{d \in U \setminus \tilde{S}} \nu(f_{\tilde{S}}, U, d) && \text{by Lemma 1,} \\ &\leq \sum_{d \in U \setminus \tilde{S}} \chi(d) && \text{by Lemma 2,} \\ &\leq \frac{1}{\beta} \cdot f(\tilde{S}) - \frac{\alpha}{\beta}|U| && \text{by Lemma 6.} \end{aligned}$$

□

Remark 8. The preceding lemmas relating $f(U)$ and $f(\tilde{S})$ do not rely on the structure of $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$.

3.2. Upper bounding $f(U \cup T)$

Let $T \in \mathcal{I}$ be any feasible solution. The goal is to upper bound $f(U \cup T)$. Here we use the fact that \mathcal{I} is a p -matchoid to frame an exchange argument between T and U .

Lemma 9. *Let $T \in \mathcal{I}$ be a feasible solution disjoint from U . There exists a mapping $\varphi : T \rightarrow 2^U$ such that*

- (a) *Every $s \in \tilde{S}$ appears in the set $\varphi(t)$ for at most p choices of $t \in T$.*
- (b) *Every $d \in U \setminus \tilde{S}$ appears in the set $\varphi(t)$ for at most $(p-1)$ choices of $t \in T$.*
- (c) *For each $t \in T$,*

$$\sum_{c \in C_t} \nu(f, S_t^-, c) \leq \sum_{d \in \varphi(t) \setminus \tilde{S}} \chi(d) + \sum_{s \in \varphi(t) \cap \tilde{S}} \nu(f, \tilde{S}, s). \quad (1)$$

Proof. The high level strategy is as follows. For each matroid $\mathcal{M}_\ell = (\mathcal{N}_\ell, \mathcal{I}_\ell)$ in the p -matchoid \mathcal{M}^p , we construct a directed acyclic graph \mathcal{G}_ℓ on \mathcal{N}_ℓ , where a subset of T forms the source vertices and arrows preserve inequality (1). Applying Lemma 30 we get an injection from a subset of T into $U \cap \mathcal{N}_\ell$. With care, the union of these injections will produce the mapping we seek.

Let us review and annotate the subroutine [Exchange-Candidates\(\$S, e\$ \)](#). For each matroid $\mathcal{M}_\ell = (\mathcal{N}_\ell, \mathcal{I}_\ell)$ in which S spans e (i.e., $(S+e) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell$), we assemble a subset $X_{e,\ell} \subseteq S \cap \mathcal{N}_\ell$ that spans e in \mathcal{M}_ℓ . Of these, we choose the element $c_{e,\ell} \in X_{e,\ell}$ with the smallest incremental value with respect to S .

Fix a matroid $\mathcal{M}_\ell = (\mathcal{N}_\ell, \mathcal{I}_\ell)$. Let

$$T_\ell = \{t \in T \cap \mathcal{N}_\ell : (S_t^- + t) \cap \mathcal{N}_\ell \notin \mathcal{I}_\ell\}$$

be the set of elements in T obstructed by \mathcal{M}_ℓ , so to speak. For each $x \in X_{t,\ell}$, add a directed edge (t, x) from t to x . Observe that for all $t \in T_\ell$, $N_{\mathcal{G}_\ell}^+(t) = X_{t,\ell}$ spans t .

Let

$$D_\ell = \{d : d = c_{e,\ell} \text{ for some } e \in U\}$$

be the elements deleted specifically for \mathcal{M}_ℓ . Observe that $D_\ell \subseteq (U \setminus \tilde{S}) \cap \mathcal{N}_\ell$. For $d \in D_\ell$, the set $Y_{d,\ell} = X_{e(d),\ell} - d + e(d)$ spans d , and for all $y \in Y_{d,\ell}$,

$$\chi(d) = \nu(f, S_{e(d)}^-, d) \leq \nu(f, S_{e(d)}^-, y) \leq \begin{cases} \chi(y) & \text{if } y \in U \setminus \tilde{S}, \\ \nu(f, \tilde{S}, y) & \text{if } y \in \tilde{S}. \end{cases}$$

For each $y \in Y_{d,\ell}$, add the directed edge (d, y) from d to y . Observe that for all $d \in D_\ell$, $N_{\mathcal{G}_\ell}^+(d) = Y_{d,\ell}$ spans d .

Clearly, \mathcal{G}_ℓ is a directed acyclic graph. The elements of T_ℓ are sources in \mathcal{G}_ℓ , and the elements of D_ℓ are never sinks. By [Lemma 30](#), there exists an injection φ_ℓ from T_ℓ to $(U \cap \mathcal{N}_\ell) \setminus D_\ell$ such that for each $t \in T_\ell$, there is a path in \mathcal{G}_ℓ from t to $\varphi_\ell(t)$. If we write out the path $t \rightarrow x_1 \rightarrow \dots \rightarrow x_z \rightarrow \varphi_\ell(t)$, we have $x_1, \dots, x_z \in U \setminus \tilde{S}$ and

$$\nu(f, S_t^-, c_{t,\ell}) \leq \nu(f, S_t^-, x_1) \leq \chi(x_1) \leq \dots \leq \chi(x_z) \leq \begin{cases} \chi(\varphi_\ell(t)) & \text{if } \varphi_\ell(t) \in U \setminus \tilde{S}, \\ \nu(f, \tilde{S}, \varphi_\ell(t)) & \text{if } \varphi_\ell(t) \in \tilde{S}. \end{cases}$$

After constructing φ_ℓ for each matroid \mathcal{M}_ℓ , define $\varphi : T \rightarrow 2^U$ by

$$\varphi(t) = \bigcup_{\ell: t \in T_\ell} \varphi_\ell(t).$$

For each $t \in T$, we have

$$\sum_{c \in C_t} \nu(f, S_t^-, c) \leq \sum_{d \in \varphi(t) \setminus \tilde{S}} \chi(d) + \sum_{s \in \tilde{S} \cap \varphi(t)} \nu(f, \tilde{S}, s).$$

Each $u \in U$ belongs to at most p matroids, so each $u \in U$ appears in $\varphi(t)$ for at most p values of $t \in T$. Since $\{D_\ell\}$ covers $U \setminus \tilde{S}$, and φ_ℓ avoids D_ℓ , each $d \in U \setminus \tilde{S}$ appears in $\varphi(t)$ at most $p - 1$ times. \square

Remark 10. A similar exchange lemma is given by Badanidiyuru for the intersection of p matroids with modular weights [[Bad11](#)], and used implicitly by Chakrabarti and Kale in their extension to submodular weights. Here we extend the argument to p -matchoids and frame it in terms of incremental values.

Now we bound $f(T \cup U)$.

Lemma 11. *Let $T \in \mathcal{I}$ be an independent set. Then*

$$f(T \cup U) \leq k\alpha + \frac{(1+\beta)^2}{\beta} \cdot p \cdot f(\tilde{S}).$$

Proof. Let $T' = T \setminus U$. By submodularity, we have,

$$f_U(T) \leq \sum_{t \in T'} f_U(t) \leq \sum_{t \in T'} f_{S_t^-}(t).$$

Since each $t \in T'$ is rejected, and $|T'| \leq k$, we have

$$\sum_{t \in T'} f_{S_t^-}(t) \leq (1+\beta) \sum_{t \in T'} \sum_{c \in C_t} \nu(f, S_t^-, c) + \alpha k.$$

Apply [Lemma 9](#) to generate a mapping $\varphi : T' \rightarrow 2^U$. We have,

$$\begin{aligned} & (1+\beta) \sum_{t \in T'} \sum_{c \in C_t} \nu(f, S_t^-, c) \\ & \leq (1+\beta) \sum_{t \in T'} \left(\sum_{d \in \varphi(t) \setminus \tilde{S}} \chi(d) + \sum_{s \in \tilde{S} \cap \varphi(t)} \nu(f, \tilde{S}, s) \right) && \text{by construction of } \varphi, \\ & \leq (1+\beta) \cdot (p-1) \sum_{d \in U \setminus \tilde{S}} \chi(d) + (1+\beta) \cdot p \sum_{s \in \tilde{S}} \nu(f, \tilde{S}, s) && \text{by construction of } \varphi, \\ & \leq \frac{1+\beta}{\beta} \cdot (p-1) \cdot f(\tilde{S}) + p \cdot (1+\beta) \cdot f(\tilde{S}) && \text{by [Lemma 6](#),} \\ & = \left(\frac{(1+\beta)^2}{\beta} \cdot p - \frac{1+\beta}{\beta} \right) f(\tilde{S}). \end{aligned}$$

To bound $f(U \cup T)$, we have,

$$\begin{aligned} f(U \cup T) &= f_U(T) + f(U) \\ &\leq k\alpha + \left(\frac{(1+\beta)^2}{\beta} \cdot p - \frac{1+\beta}{\beta} \right) f(\tilde{S}) + f(U) && \text{by the above,} \\ &\leq k\alpha + \frac{(1+\beta)^2}{\beta} \cdot p \cdot f(\tilde{S}) && \text{by [Lemma 7](#),} \end{aligned}$$

as desired. □

3.3. A bound for the monotone case

If f is monotone, then $f(T) \leq f(U \cup T)$ for any set T . If we take T to be the set T^* achieving OPT, $\alpha = 0$, and $\beta = 1$, we obtain the followings.

Corollary 12. *Let $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$ be a p -matchoid of rank k , and let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}_{\geq 0}$ be a nonnegative monotone submodular function. Given a stream over \mathcal{N} , **Streaming-Greedy(0,1)** is an online algorithm that returns a set $\tilde{S} \in \mathcal{I}$ such that*

$$f(\tilde{S}) \geq \frac{1}{4p} \cdot \text{OPT} = \frac{1}{4p} \cdot \max\{f(T) : T \in \mathcal{I}\}.$$

Remark 13. Although these bounds match those of Chakrabarti and Kale for the intersection of p matroids [CK14], **Randomized-Greedy** requires more calls to the submodular value oracle as the incremental value of an element in S updates over time. That said, the number of times a taken element $e \in U$ reevaluates its incremental value is proportional to the number of times an element in S_e^- is deleted, which is at most the rank of \mathcal{M}^p and generally considered small compared to the size of the stream. Furthermore, by taking α proportional to OPT (a procedure for which is discussed in Section 4.6), we can limit the size of U and thereby the number of additional oracle calls generated by shifting incremental values.

4. Randomized Streaming Greedy

Randomized-Streaming-Greedy adapts **Streaming-Greedy** to nonnegative submodular functions by employing a randomized buffer B to limit the probability that any element is added to the running solution S . Like **Streaming-Greedy**, **Randomized-Streaming-Greedy** maintains the invariant $S \in \mathcal{I}$. However, when a “good” element would have been added to S by **Streaming-Greedy**, it is instead placed in B . Once the number of elements in B hits a limit K , we pick one element in B uniformly at random and add it to S just as **Streaming-Greedy** would.

Modifying S may break the invariant that the buffer only contains good elements. Since f is submodular, the incremental value $\nu(f, S, e)$ of each $e \in S$ may increase if a preceding element is deleted. Furthermore, the marginal value $f_S(b)$ of each buffered element $b \in B$ may decrease as elements are added to S . Thus, after modifying S , we reevaluate each $b \in B$ and discard elements that are no longer good.

Let \tilde{B} be the set of elements remaining in the buffer B when the stream ends. We process \tilde{B} with an offline algorithm to produce a second solution S' , and finally return the set \hat{S} which is the better of S and S' .

Outline of the analysis. Let $T \in \mathcal{I}$ be an arbitrary independent set. Let $T' = T \setminus \tilde{B}$ be the portion fully processed by the online portion and $T'' = T \cap \tilde{B}$ the remainder left over in the buffer and processed offline.


```

Randomized-Streaming-Greedy( $\alpha, \beta$ )
 $S \leftarrow \emptyset, B \leftarrow \emptyset$ 
while (stream is not empty)
   $e \leftarrow$  next element in the stream
  if Is-Good( $S, e$ ) then  $B \leftarrow B + e$ 
  if  $|B| = K$  then
     $e \leftarrow$  uniformly random from  $B$ 
     $C \leftarrow$  Exchange-Candidates( $S, e$ )
     $B \leftarrow B - e, S \leftarrow (S \setminus C) + e$ 
    for all  $e' \in B$ 
      unless Is-Good( $S, e'$ )
         $B \leftarrow B - e'$ 
  end if
end while
 $S' \leftarrow$  Offline( $B$ )
return  $\arg \max_{Z \in \{S, S'\}} f(Z)$ 

```

```

Is-Good( $S, e$ )
 $C \leftarrow$  Exchange-Candidates( $S, e$ )
if  $f_S(e) \geq \alpha + (1 + \beta) \sum_{e' \in C} \nu(f, S, e')$ 
  return TRUE
else return FALSE

```

In Section 4.1, we first show that the analysis for T' largely reduces to that of Section 3. In particular, this gives us a bound on $f(U \cup T')$. In Section 4.2, we combine this with a bound on $f(T'')$, guaranteed by the offline algorithm, to obtain an overall bound on $f(U \cup T)$ by $f(\hat{S})$. In Section 4.3, we finally bound $f(T)$ with respect to $f(U)$, leveraging the fact that the buffer limits the probability of elements being added to S . In Section 4.4, we tie together the analysis to bound $f(T)$ by $f(\hat{S})$ for fixed α and β .

The analysis reveals that the optimal choice for β is 1, and that α should be chosen in proportion to OPT/k , where k is the rank of the \mathcal{M}^p . Since OPT is not known a priori, in Section 4.6, we leverage a technique by Badanidiyuru *et al.* [BMKK14] that efficiently guesses the α to within a constant factor of the target value. The final algorithm is then $\log k$ copies of Randomized-Streaming-Greedy run in parallel, each instance corresponding to a “guess” for α . One of these guesses is approximately correct, and attains the bounded asserted in Theorem 14.

Theorem 14. *Let $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$ be a p -matchoid of rank k , let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}_{\geq 0}$ a nonnegative submodular function over \mathcal{N} , and let $\epsilon > 0$ be fixed. Suppose there exists an algorithm for the offline instance of the problem with approximation ratio γ_p . Then there exists a streaming algorithm using total space $O\left(\frac{k \log k}{\epsilon^2}\right)$ that, given a stream over \mathcal{N} , returns a set $\hat{S} \in \mathcal{I}$ such that*

$$(1 - \epsilon)\text{OPT} \leq \left(4p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})].$$

Some notation for the analysis

- Let \tilde{S} be the state of S at the end of the stream.

```

Randomized-Streaming-Greedy( $\alpha, \beta$ )
Let  $G$  be an instance of Streaming-Greedy( $\alpha, \beta$ )
Let  $S_G$  refer to the set  $S$  maintained by  $G$ .
 $B \leftarrow \emptyset$  //  $B$  is a buffer of size  $K > 0$ 
while (stream is not empty)
   $e \leftarrow$  next element in the stream
  if Is-Good( $S_G, e$ ) then  $B \leftarrow B + e$  // buffers the “good” elements
  else send  $e$  downstream to  $G$  //  $G$  will reject  $e$ 
  if  $|B| = K$  then
     $e \leftarrow$  an element from  $B$  picked uniformly at random //  $e$  is “good”
     $B \leftarrow B - e$ 
    send  $e$  downstream to  $G$  //  $G$  will add  $e$  to  $S_G$ 
    for all  $e' \in B$  such that (not Is-Good( $S_G, e'$ ))
       $B \leftarrow B - e'$ 
      send  $e'$  downstream to  $G$  //  $G$  will reject  $e'$ 
    end for
  end if
end while
 $S' \leftarrow$  Offline( $B$ )
Return  $\arg \max_{Z \in \{S_G, S'\}} f(Z)$ 

```

Figure 2: Randomized-Streaming-Greedy rewritten with the deterministic portion reduced to Streaming-Greedy.

- Let U be the set of all elements to pass through S during the stream.
- Let \tilde{B} be the set held by B at the end of the stream.
- Let $\hat{S} = \arg \max_{Z \in \{\tilde{S}, S'\}} f(Z)$ be the set output by Randomized-Streaming-Greedy.

\tilde{S} , U , and \tilde{B} are random sets depending on the random selection process from B . \hat{S} is a random variable depending on \tilde{B} , \tilde{S} , and the offline algorithm’s own internal randomization.

4.1. Reducing to Streaming-Greedy

If we set the buffer limit K to 1, eliminating the role of the buffer B , then Randomized-Streaming-Greedy reduces to the deterministic Streaming-Greedy algorithm from Section 3. In Figure 2, we refactor Randomized-Streaming-Greedy as a buffer placed upstream from a running instance of Streaming-Greedy. The buffer only filters and reorders the stream, and the analysis of Streaming-Greedy holds with respect to this scrambled stream. More precisely, if r denotes the random bits dictating B , then for any fixed r , the analysis of Section 3 still applies. We recap the preceding analysis for Streaming-Greedy as it applies here.

Lemma 15. Let $\alpha, \beta \geq 0$ be fixed parameters. For any $T \in \mathcal{I}$, we have

$$f(U \cup (T \setminus \tilde{B})) \leq \frac{(1 + \beta)^2}{\beta} \cdot p \cdot f(\tilde{S}) + k\alpha.$$

Furthermore, $|U| \leq \text{OPT}/\alpha$.

4.2. Upper bounding $f(U \cup T)$ by $f(\hat{S})$

To bound $f(U \cup T)$, where $T \in \mathcal{I}$ is any independent set, we split $f_U(T)$ into the portion $f_U(T \setminus B)$ that escapes the buffer, and the remainder $f_U(T \cap B)$ captured by the buffer. We bound the former with [Lemma 15](#) and the latter by guarantees for [Offline](#) to obtain the following.

Lemma 16. For any $T \in \mathcal{I}$, we have

$$\mathbf{E}[f(U \cup T)] \leq k\alpha + \left(\frac{(1 + \beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p} \right) \mathbf{E}[f(\hat{S})]$$

Proof. For ease of exposition, let r denote the random bits that dictate the random selections from B , and let us subscript variables by r to highlight their dependence. Let $T'_r = T \setminus \tilde{B}_r$ and $T'' = T \cap \tilde{B}_r$. For any fixed r , we have,

$$\begin{aligned} f(U_r \cup T) &= f(U_r) + f_{U_r}(T) \leq f(U_r) + f_{U_r}(T'_r) + f_{U_r}(T''_r) && \text{by submodularity,} \\ &\leq k\alpha + \frac{(1 + \beta)^2}{\beta} \cdot p \cdot f(\tilde{S}_r) + f_{U_r}(T''_r) && \text{by Lemma 15,} \\ &\leq k\alpha + \frac{(1 + \beta)^2}{\beta} \cdot p \cdot f(\tilde{S}_r) + f(T''_r) && \text{by submodularity,} \\ &\leq k\alpha + \frac{(1 + \beta)^2}{\beta} \cdot p \cdot f(\tilde{S}_r) + \frac{1}{\gamma_p} \mathbf{E}[f(S'_r)]. \end{aligned}$$

Here, the expectation surrounding $f(S'_r)$ is generated by the offline algorithm [Offline](#), which may be randomized (see, for example, [Table 1](#)). Taking expectations of both sides over r , we have

$$\begin{aligned} \mathbf{E}[f(U \cup T)] &\leq k\alpha + \frac{(1 + \beta)^2}{\beta} \cdot p \cdot \mathbf{E}[f(\tilde{S})] + \frac{1}{\gamma_p} \mathbf{E}[f(S')] \\ &\leq k\alpha + \left(\frac{(1 + \beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p} \right) \mathbf{E}[f(\hat{S})], \end{aligned}$$

as desired. □

4.3. Upper bounding $f(T)$ by $f(U \cup T)$

The remaining challenge is to bound $\mathbf{E}[f(U \cup T)]$ from below by some fraction of $f(T)$. The following technical lemma, used similarly by Buchbinder *et al.*, gives us a handle on $\mathbf{E}[f(U \cup T)]$.

Lemma 17 ([BFNS14]). *Let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}_{\geq 0}$ be a nonnegative submodular function. Suppose R is a random set according to a distribution μ on $2^{\mathcal{N}}$ where no element $e \in \mathcal{N}$ is picked with probability more than ρ . Then $\mathbf{E}[f(R)] \geq (1 - \rho)f(\emptyset)$. Moreover, for any set $Y \subseteq \mathcal{N}$, $\mathbf{E}[f(R \cup Y)] \geq (1 - \rho)f(Y)$.*

In this case, U is a random set, and we want to upper bound the probability of an element $e \in \mathcal{N}$ appearing in U . Intuitively, taking K large limits the probability of an element being selected from the buffer, while by Lemma 15, taking α large decreases the number of elements in U .

Lemma 18. *For any element $e \in \mathcal{N}$,*

$$\mathbf{P}[e \in U] \leq 1 - \left(1 - \frac{1}{K}\right)^{\text{OPT}/\alpha}.$$

Proof. An element is added to S (and therefore U) if and only if it is selected from B when $|B|$ reaches K . By Lemma 15, we select from B at most OPT/α times, and each selection is made uniformly and independently at random from K elements. \square

With this, we apply Lemma 17 to give the following.

Lemma 19. *Let $T \in \mathcal{I}$ be a fixed independent set. Then*

$$\mathbf{E}[f(U \cup T)] \geq \left(1 - \frac{1}{K}\right)^{\text{OPT}/\alpha} f(T).$$

Proof. By Lemma 18, for all $e \in \mathcal{N}$, $\mathbf{P}[e \in U] \leq \left(1 - (1 - 1/K)^{\text{OPT}/\alpha}\right) \equiv \rho$. The claim then follows Lemma 17. \square

4.4. Overall Analysis

Tying together Lemma 16 and Lemma 19, we have the following.

Lemma 20. *For any $T \in \mathcal{I}$, we have*

$$\left(1 - \frac{\text{OPT}}{\alpha K}\right) f(T) \leq k\alpha + \left(\frac{(1 + \beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})]$$

Proof. Composing [Lemma 16](#) and [Lemma 19](#), we have,

$$\left(1 - \frac{1}{K}\right)^{\text{OPT}/\alpha} f(T) \leq k\alpha + \left(\frac{(1+\beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})].$$

By Bernoulli's inequality,

$$\left(1 - \frac{1}{K}\right)^{\text{OPT}/\alpha} \geq 1 - \frac{\text{OPT}}{\alpha K},$$

and the claim follows. \square

4.5. A bound for approximate α

We would like to fix α as a constant fraction of OPT. For example, taking $\alpha = \epsilon \text{OPT}/2k$, where $\epsilon > 0$, and plugging into [Lemma 20](#) gives the cleaner bound,

$$\left(1 - \frac{2k}{\epsilon K}\right) f(T) \leq \frac{\epsilon}{2} \text{OPT} + \left(\frac{(1+\beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})].$$

However, the algorithm does not know OPT, and instead we will try to estimate OPT approximately. Let us lay out the bound when α is within a factor of 2 of $\epsilon \text{OPT}/2k$.

Lemma 21. *Let $\epsilon > 0$ be a fixed parameter. If $\epsilon \cdot \text{OPT}/4k \leq \alpha \leq \epsilon \cdot \text{OPT}/2k$, then*

$$\left(1 - \frac{4k}{\epsilon K}\right) \text{OPT} \leq \frac{\epsilon}{2} \text{OPT} + \left(\frac{(1+\beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})].$$

In particular, for $K = 4k/\epsilon^2$, we have

$$(1 - \epsilon) \text{OPT} \leq \left(\frac{(1+\beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})].$$

4.6. Efficiently estimating α

Badanidiyuru *et al.* showed how to “guess” OPT space-efficiently and in a single pass [\[BMKK14\]](#). Let $z = \arg \max_{x \in \mathcal{N}} f(x)$. Clearly, $\text{OPT} \geq f(z)$, and by submodularity of f ,

$$\text{OPT} = \max_{T \in \mathcal{I}} f(T) \leq \max_{T \in \mathcal{I}} \sum_{t \in T} f(t) \leq \max_{T \in \mathcal{I}} |T| \cdot f(z) \leq k \cdot f(z).$$

Fix $\epsilon > 0$, and suppose we run a parallel copy of [Randomized-Streaming-Greedy](#) for each α in

$$\mathcal{A}(z) = \{2^i : i \in \mathbb{Z}\} \cap \left[\frac{\epsilon}{4k} \cdot f(z), \frac{\epsilon}{2} f(z)\right],$$

and at the end of the stream return the best solution among the $\log(2k)$ copies. For some $\alpha \in \mathcal{A}(z)$, we have

$$\frac{\epsilon}{4k} \cdot \text{OPT} \leq \alpha \leq \frac{\epsilon}{2k} \cdot \text{OPT},$$

where we get the approximation guarantee in [Lemma 20](#).

This strategy requires two passes: one to identify z , and the second running $\log(2k)$ copies of [Randomized-Streaming-Greedy](#) in parallel. We can reduce the number of passes to 1 by updating z and $\mathcal{A}(z)$ on the fly. Enumerate the stream e_1, e_2, \dots , and for each i , let

$$z_i = \arg \max_{e_j: j \in [i]} f(e_j)$$

be the single element maximizing f among the first i elements seen thus far. $\mathcal{A}(z_i)$ shifts up over through the stream as z_i is updated. At each step i , we maintain parallel solutions for each choice of $\alpha \in \mathcal{A}(z_i)$, deleting instances with α below $\mathcal{A}(z_i)$ and instantiating new instances with larger values of α .

To ensure correctness, it suffices to show that when we instantiate an instance of [Randomized-Streaming-Greedy](#) for a new threshold α , we haven't skipped over any elements that we would want to include. Let $\alpha \in \mathcal{A}(z_i) - \mathcal{A}(z_{i-1})$, i.e. $f(z_{i-1}) < \alpha \leq f(z_i)$. If $f(e_j) \geq \alpha$ for some $j < i$, then

$$\alpha \leq f(e_j) \leq f(z_{i-1}),$$

a contradiction.

4.7. Simpler algorithm and better bound for cardinality constraint

When the p -matchoid is simply a cardinality constraint with rank k , we can do better. If we set $\beta = \infty$ in [Randomized-Streaming-Greedy\(\$\alpha, \beta\$ \)](#), then the algorithm will only try to add to S without exchanging while $|S| < k$, effectively halting once we meet the cardinality constraint $|S| = k$. In [Figure 3](#), we rewrite [Randomized-Streaming-Greedy\(\$\alpha, \infty\$ \)](#) with the unnecessary logic removed.

Lemma 22. *If $|\tilde{S}| = k$, then $f(\tilde{S}) \geq k\alpha$.*

Lemma 23. *If $|\tilde{S}| < k$, then for any set $T \subseteq \mathcal{N}$,*

$$f(\tilde{S} \cup T) \leq f(\tilde{S}) + f(T \cap B) + \alpha|T|.$$

Proof. Fix $t \in T \setminus (\tilde{S} \cup B)$, and let S_t^- be the set held by \tilde{S} when t is processed. Since t is rejected, and $S_t^- \subseteq \tilde{S}$, we have

$$f_{\tilde{S}}(t) \leq f_{S_t^-}(t) \leq \alpha.$$

```

Randomized-Streaming-Greedy( $\alpha, \infty$ )
 $B \leftarrow \emptyset, S \leftarrow \emptyset$ 
while (stream is not empty)
   $e \leftarrow$  next element in the stream
  if  $|S| \leq k$  and  $f_S(e) > \alpha$  then
     $B \leftarrow B + e$ 
  if  $|B| = K$  then
     $e \leftarrow$  uniformly random from  $B$ 
     $B \leftarrow B - e, S \leftarrow S + e$ 
    for all  $e' \in B$  s.t.  $f_S(e') \leq \alpha$ 
       $B \leftarrow B - e'$ 
    end if
  end while
 $S' \leftarrow \text{Offline}(B)$ 
return  $\arg \max_{Z \in \{S, S'\}} f(Z)$ 

```

Figure 3

Summed over all $t \in T \setminus (\tilde{S} \cup B)$, we have

$$f_{\tilde{S}}(T \setminus B) \leq \sum_{t \in T \setminus B} f_{\tilde{S}}(t) \leq \alpha |T|.$$

Finally, we write

$$\begin{aligned} f(\tilde{S} \cup T) &= f_{\tilde{S}}(T) + f(\tilde{S}) \leq f_{\tilde{S}}(T \setminus B) + f_{\tilde{S}}(T \cap B) + f(\tilde{S}) \\ &\leq f(\tilde{S}) + f(T \cap B) + \alpha |T| \end{aligned}$$

to attain the desired bound. \square

Lemma 24. For $K = k/\epsilon$, and α such that $(1 - \epsilon)\text{OPT} \leq (2 + e)k\alpha \leq (1 + \epsilon)\text{OPT}$, we have

$$\mathbf{E}[f(\hat{S})] \geq \frac{1 - 2\epsilon}{2 + e} \cdot \text{OPT}.$$

Proof. Let $T \subseteq \mathcal{N}$ be an optimal set with $|T| = k$ and $\text{OPT} = f(T)$.

If $|\tilde{S}| = k$, then the claim follows [Lemma 22](#). Otherwise, $|\tilde{S}| < k$ and by [Lemma 23](#), we have

$$f(\tilde{S} \cup T) \leq f(\tilde{S}) + f(T \cap B) + k\alpha.$$

By [Lemma 17](#), we also have

$$\mathbf{E}[f(\tilde{S} \cup T)] \geq \left(1 - \frac{1}{K}\right)^k f(T) \geq \left(1 - \frac{k}{K}\right) f(T) \geq (1 - \epsilon) f(T).$$

Finally, by the bound for [Offline](#), we have,

$$f(T \cap B) \leq e \mathbf{E}[f(S')]$$

Together, we have

$$(1 - \epsilon)f(T) \leq \mathbf{E}[f(\tilde{S}) + ef(S')] + k\alpha \leq (1 + e) \mathbf{E}[f(\hat{S})] + k\alpha.$$

Solving for $\mathbf{E}[f(\hat{S})]$ and plugging in $f(T) = \text{OPT}$ and $k\alpha \leq \frac{1+\epsilon}{2+e} \text{OPT}$, we have

$$\mathbf{E}[f(\hat{S})] \geq \frac{1}{1+e}((1 - \epsilon)\text{OPT} - k\alpha) \geq \frac{(1 - 2\epsilon)}{2+e} \text{OPT},$$

as desired. \square

The preceding analysis reveals that the appropriate choice for α is $\text{OPT}/(2+e)k$, where $\text{OPT} = \max\{f(T) : |T| \leq k\}$ is the maximum value attainable by a set of k elements, and that a sufficiently large choice for K is k/ϵ . As in [Section 4.6](#), we can efficiently approximate α by guessing α in increasing powers of $(1 + \epsilon)$, maintaining at most $O(\log_{1+\epsilon} k) = O(\epsilon^{-1} \log k)$ instances of [Randomized-Streaming-Greedy\(\$\alpha, \infty\$ \)](#) at any instant. The resulting bound is stronger than previously derived for a 1-matchoid.

Theorem 25. *Let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}_{\geq 0}$ be a nonnegative submodular function over a ground set \mathcal{N} , and let $\epsilon > 0$ be fixed. Then there exists a streaming algorithm using total space $O\left(\frac{k \log k}{\epsilon^2}\right)$ that, given a stream over \mathcal{N} , returns a set \hat{S} such that $|\hat{S}| \leq k$ and $f(\hat{S}) \geq \frac{1-\epsilon}{2+e} \cdot \text{OPT}$, where $\text{OPT} = \max\{f(T) : |T| \leq k\}$ is the maximum value attainable by a set of k elements.*

5. A Deterministic Algorithm via Iterated Greedy

Gupta *et al.* gave a framework that takes an offline algorithm for maximizing a monotone submodular functions and, by running the algorithm as a black box multiple times over different groundsets, produces an algorithm for the nonnegative case [[GRST10](#)]. Here we adapt the framework to the streaming setting, employing [Streaming-Greedy](#) as the blackbox for the monotone case.

We first present [Iterated-Streaming-Greedy](#) as an algorithm making two passes over \mathcal{N} . In the first pass we run [Streaming-Greedy\(\$\alpha, \beta\$ \)](#) over \mathcal{N} as usual. Let S_1 denote the set output, and U_1 the set of all elements added to S_1 at any intermediate point of the algorithm, as per [Section 3](#). In the second pass, we run [Streaming-Greedy\(\$0, \beta\$ \)](#) over the set $\mathcal{N} \setminus U_1$ of elements that were immediately rejected in the first pass to produce another independent set S_2 . Lastly, we run our choice of offline algorithm over U_1 to produce a third independent set $S_3 \in \mathcal{I}$. At the end, return the best set among S_1 , S_2 , and S_3 .

If we pipeline the two instances of [Streaming-Greedy](#), then [Iterated-Streaming-Greedy](#) becomes a true streaming algorithm with only one pass over \mathcal{N} . When the first instance

```

Iterated-Streaming-Greedy( $\alpha, \beta, \mathcal{N}$ )
// run Streaming-Greedy over  $\mathcal{N}$ 
( $S_1, U_1$ )  $\leftarrow$  Streaming-Greedy( $\alpha, \beta, \mathcal{N}$ )
//  $U_1$  denotes the set  $U$  in Section 3.
 $S_2 \leftarrow$  Streaming-Greedy( $0, \beta, \mathcal{N} \setminus U_1$ )
 $S_3 \leftarrow$  Offline( $U_1$ )
return  $\arg \max_{\hat{S} \in \{S_1, S_2, S_3\}} f(\hat{S})$ 

```

rejects an element e outright, e is sent downstream to the second instance of **Streaming-Greedy**. Note that the running time of **Offline** depends on the size of its input U_1 , which by [Lemma 5](#) is at most OPT/α .

Lemma 26. *Let $T \in \mathcal{I}$ be any independent set. Then*

$$\left(2 \cdot \frac{(1+\beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})] \geq f(T) - k\alpha$$

Furthermore, if **Offline** is a deterministic algorithm, then **Iterated-Streaming-Greedy** is deterministic and the above holds without taking expectations.

Proof. By submodularity, we have

$$f(U_1 \cup T) + f(U_2 \cup (T \setminus U_1)) \geq f(U_1 \cup S_2) + f(T \setminus U_1) \quad (2)$$

and

$$f(T \setminus U_1) + f(U_1 \cap T) \geq f(T) + f(\emptyset). \quad (3)$$

By nonnegative of f and equations (2) and (3), we have,

$$f(T) \leq f(T) + f(\emptyset) + f(U_1 \cup S_2) \leq f(U_1 \cup T) + f(U_2 \cup (T \setminus U_1)) + f(U_1 \cap T).$$

By [Corollary 12](#), we have $f(U_1 \cup T) \leq \frac{(\beta+1)^2}{\beta} \cdot p \cdot f(S_1) + k\alpha$ and $f(U_2 \cup (T \setminus U_1)) \leq \frac{(\beta+1)^2}{\beta} \cdot p \cdot f(S_2)$, and $f(U_1 \cap T) \leq \frac{1}{\gamma_p} \cdot f(S_3)$ by assumption. Plugging into the above, and noting that $f(S_1), f(S_2), f(S_3) \leq f(\hat{S})$ gives the bounds we seek. \square

Corollary 27. *Let $\epsilon > 0$ be given. If $\alpha \leq \epsilon \text{OPT}/k$, then*

$$\left(2 \frac{(1+\beta)^2}{\beta} \cdot p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})] \geq (1 - \epsilon) \text{OPT},$$

and the inequality holds without taking expectations if **Offline** is a deterministic algorithm.

The appropriate value of α is guessed efficiently exactly as described in [Section 4.6](#). Here, if $|U_1|$ grows too large in an instance of [Iterated-Streaming-Greedy\(\$\alpha, \beta\$ \)](#) for some fixed α , then α must be too small and we can terminate the instance immediately.

Theorem 28. *Let $\mathcal{M}^p = (\mathcal{N}, \mathcal{I})$ be a p -matchoid of rank k , let $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}_{\geq 0}$ be a nonnegative submodular function over \mathcal{N} , let $\epsilon > 0$ be fixed. Suppose there exists an offline algorithm for finding the largest value independent set in p -matchoid with approximation ratio γ_p . Then there exists a streaming algorithm using total space $O\left(\frac{k \log k}{\epsilon}\right)$ that, given a stream of \mathcal{N} , returns a set $\hat{S} \in \mathcal{I}$ such that*

$$\left(8p + \frac{1}{\gamma_p}\right) \mathbf{E}[f(\hat{S})] \geq (1 - \epsilon) \text{OPT}.$$

If the offline algorithm is deterministic, then the claimed algorithm is deterministic and the above bound holds without expectation.

References

- [Bad11] A. Badanidiyuru Varadaraja. Buyback problem: Approximate matroid intersection with cancellation costs. In *Proc. 38th Internat. Colloq. Automata Lang. Prog. (ICALP)*, volume 1, pages 379–390, 2011.
- [BFNS12] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. In *Proc. 53rd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 649–658, 2012.
- [BFNS14] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. In *Proc. 25th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 1433–1452, 2014.
- [BFS15] N. Buchbinder, M. Feldman, and R. Schwartz. Online submodular maximization with preemption. In *Proc. 26th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 1202–1216, 2015.
- [BHZ13] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam. Submodular secretary problem and extensions. *ACM Trans. Algs.*, 9(4):32:1–32:23, October 2013.
- [BIK07] M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. 18th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 434–443, Philadelphia, PA, USA, 2007.
- [BKNS12] N. Bansal, N. Korula, V. Nagarajan, and A. Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theo. Comput.*, 8(1):533–565, 2012.

- [BMKK14] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular optimization: Massive data summarization on the fly. In *Proc. 20th ACM Conf. Knowl. Disc. and Data Mining (KDD)*, pages 671–680, 2014.
- [BV14] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proc. 25th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 1497–1514, 2014.
- [CCPV07] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Proc. 12th Int. Conf. Int. Prog. Comb. Opt. (IPCO)*, pages 182–196, 2007.
- [CCPV11] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.
- [CJV15] C. Chekuri, T.S. Jayram, and J. Vondrák. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of ITCS*, 2015.
- [CK14] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: matchings, matroids and more. In *Proc. 17th Int. Conf. Int. Prog. Comb. Opt. (IPCO)*, pages 210–221, 2014.
- [CVZ11] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proc. 43th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 783–792, 2011.
- [CWW10] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. 16th ACM Conf. Knowl. Disc. and Data Mining (KDD)*, pages 1029–1038, 2010.
- [CWY09] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proc. 15th ACM Conf. Knowl. Disc. and Data Mining (KDD)*, pages 199–208, New York, NY, USA, 2009.
- [DKR13] A. Dasgupta, R. Kumar, and S. Ravi. Summarization through submodularity and dispersion. In *Proc. 51st Ann. Meet. Assoc. for Comp. Ling. (ACL)*, volume 1, pages 1014–1022, 2013.
- [DV12] S. Dobzinski and J. Vondrak. From query complexity to computational complexity. In *Proc. 44th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 1107–1116, 2012.
- [FKM⁺05] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theo. Comp. Sci.*, 348(2–3):207–216, 2005.

- [FNS11] M. Feldman, J. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Proc. 52nd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 570–579, 2011.
- [FNSW11] M. Feldman, J. Naor, R. Schwartz, and J. Ward. Improved approximations for k -exchange systems. In *Proc. 19th Annu. European Sympos. Algs. (ESA)*, pages 784–798, 2011.
- [FNW78] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – II. *Math. Prog. Studies*, 8:73–87, 1978.
- [Fuj05] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.
- [FW14] Y. Filmus and J. Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM J. Comput.*, 43(2):514–542, 2014.
- [GBL11] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A data-based approach to social influence maximization. *Proc. VLDB Endow.*, 5(1):73–84, September 2011.
- [GRST10] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proc. 6th Int. Conf. Internet and Network Economics (WINE)*, pages 246–257, 2010.
- [IJB13] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *Proc. 30th Int. Conf. Mach. Learning (ICML)*, volume 28, pages 855–863, 2013.
- [KKT03] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM Conf. Knowl. Disc. and Data Mining (KDD)*, pages 137–146, New York, NY, USA, 2003.
- [KMVV13] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In *Proc. 25th Ann. ACM Sympos. Parallelism Alg. Arch. (SPAA)*, pages 1–10, 2013.
- [KST13] A. Kulik, H. Shachnai, and T. Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Math. Oper. Res.*, 38(4):729–739, 2013.
- [LB11] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proc. 49th Ann. Meet. Assoc. Comput. Ling.: Human Lang. Tech. (HLT)*, volume 1, pages 510–520, 2011.
- [LKG⁺07] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proc. 13th ACM Conf. Knowl. Disc. and Data Mining (KDD)*, pages 420–429, New York, NY, USA, 2007.

- [LMNS10] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.*, 23(4):2053–2078, 2010.
- [LSV10] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 35:795–806, 2010.
- [McG05] A. McGregor. Finding graph matchings in data streams. In *8th Intl. Work. Approx. Algs. Combin. Opt. Problems*, pages 170–181, 2005.
- [NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Math. Prog.*, 14(1):265–294, 1978.
- [Sch03] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Verlag, 2003.
- [SS13] L. Seeman and Y. Singer. Adaptive seeding in social networks. In *Proc. 54th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 459–468, 2013.
- [SSSJ12] R. Sipos, A. Swaminathan, P. Shivaswamy, and T. Joachims. Temporal corpus summarization using submodular word coverage. In *Proc. 21st ACM Int. Conf. Inf. and Know. Management (CIKM)*, pages 754–763, 2012.
- [Von13] J. Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013.

A. Exchange lemmas for matroids

Lemma 29. *Let $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ be a matroid, let $S, T \subseteq \mathcal{N}$ be two subsets, and let $x, y \in \mathcal{N}$ be two elements. If S spans x , and $T + x$ spans y , then $S \cup T$ spans y .*

Proof. It suffices to assume that S and T are independent sets.

Extend S to a base B in $S \cup T$. Since B extends S , B spans x , and B is a base in $(S \cup T) + x$. Since $(S \cup T) + x$ spans y , B spans y . \square

Let \mathcal{G} be a directed graph. For $v \in \mathcal{V}(\mathcal{G})$, let $N^+(v) = \{w : (v, w) \in \mathcal{E}(\mathcal{G})\}$ denote the set of outgoing neighbors of v , and $N^-(v) = \{u : (u, v) \in \mathcal{E}(\mathcal{G})\}$ the set of incoming neighbors of v .

The following lemma is implicit in Badanidiyuru [Bad11].

Lemma 30. *Let $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ be a matroid, and \mathcal{G} a directed acyclic graph over \mathcal{N} such that for every non-sink vertex $e \in \mathcal{N}$, the outgoing neighbors $N^+(e)$ of e span e . Let $I \in \mathcal{I}$ be an independent set such that no path in \mathcal{G} goes from one element in I to another. Then there exists an injection from I to sink vertices in \mathcal{G} such that each $e \in I$ maps into an element reachable from e .*

Proof. Restricting our attention to elements reachable from I in \mathcal{G} , let us assume that the elements of I are sources (i.e., have no incoming neighbors) in \mathcal{G} . Let us call an element $e \in \mathcal{N}$ an “internal” element if it is neither a sink nor a source in \mathcal{G} .

We prove by induction on the number of internal vertices reachable from I . In the base case, the outgoing neighbors of each $i \in I$ are all sinks, and \mathcal{G} is bipartite. For any subset $J \subseteq I$, $N^+(J) = \bigcup_{i \in J} N^+(i)$ spans J . J is independent, so we have $|J| \leq |N^+(J)|$. Thus, by Hall’s matching theorem, there exists an injection $I \hookrightarrow N^+(I)$ such that each $i \in I$ maps into $N^+(i)$.

In the general case, let $e \in \mathcal{N} \setminus I$ be an internal vertex in \mathcal{G} . Consider the graph \mathcal{H} removing e and preserving all paths through e , defined by,

$$\begin{aligned}\mathcal{V}(\mathcal{H}) &= \mathcal{V}(\mathcal{G}) - e = \mathcal{N} - e, \\ \mathcal{E}(\mathcal{H}) &= (\mathcal{E}(\mathcal{G}) \setminus \{(a, b) : a = e \text{ or } e = a\}) \cup \{(a, b) : (a, e), (e, b) \in \mathcal{E}(\mathcal{G})\}\end{aligned}$$

\mathcal{H} has one less internal vertex than \mathcal{G} , the same sink vertices as \mathcal{G} , and a vertex $a \in \mathcal{V}(\mathcal{H})$ is reachable from $i \in I$ in \mathcal{H} iff it is reachable from i in \mathcal{G} . For any vertex $a \in N_{\mathcal{G}}^-(e)$ that had an outgoing arc into e , we have

$$N_{\mathcal{H}}^+(a) = (N_{\mathcal{G}}^+(a) - e) \cup N_{\mathcal{G}}^+(e),$$

which spans a by [Lemma 29](#). Since any other vertices has the same outgoing arcs, we conclude that $N_{\mathcal{H}}^+(a)$ spans a for any non-sink vertex of \mathcal{H} .

By induction, there exists an injection from I into the sinks of \mathcal{H} such that every $i \in I$ is mapped to a sink vertex reachable from i in \mathcal{H} . By construction, these vertices are also reachable sinks in \mathcal{G} , as claimed. \square