The Open University of Israel
Department of Mathematics and Computer Science

Streaming Maximization of Submodular Functions Subject to Cardinality Constraint

Thesis submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Computer Science Division

By
Naor Alaluf

Prepared under the supervision of Prof. Moran Feldman

November 2020

Abstract

We study the problem of maximizing a *non-monotone* submodular function subject to a cardinality constraint in the streaming model. Our main contribution is a single-pass (semi-) streaming algorithm that uses roughly $O(k/\varepsilon^2)$ memory, where k is the size constraint. At the end of the stream, our algorithm post-processes its data structure using *any offline algorithm* for submodular maximization, and obtains a solution whose approximation guarantee is $\frac{\alpha}{1+\alpha} - \varepsilon$, where α is the approximation of the offline algorithm. If we use an exact (exponential time) post-processing algorithm, this leads to $\frac{1}{2} - \varepsilon$ approximation (which is nearly optimal). If we post-process with the algorithm of [7], that achieves the state-of-the-art offline approximation guarantee of $\alpha = 0.385$, we obtain 0.2779-approximation in polynomial time, improving over the previously best polynomial-time approximation of 0.1715 due to [24]. It is also worth mentioning that our algorithm is combinatorial and deterministic, which is rare for an algorithm for non-monotone submodular maximization, and enjoys a fast update time of $O(\frac{\log k + \log(1/\alpha)}{\varepsilon^2})$ per element. Additionally, we show that our main algorithm can be naturally modified to maximize continuous DR-submodular functions subject to a support cardinality constraint.

Dedications and Acknowledgements

This thesis is dedicated to my family, in the loving memory of our dearest mother, who passed away earlier this year after fighting cancer for two years. My parents, brother and sister saw potential in me long before I could, and I am forever grateful for their endless love, support and motivation.

I wish to express my deepest gratitude and appreciation to my thesis advisor, Prof. Moran Feldman. Moran gave me the opportunity to prove my learning abilities after a very short acquaintance, and was very sensitive to my delicate situation at home. His guidance throughout the work on our published paper and this thesis was truly inspiring, and it helped me gain confidence and a lot of intuition when approaching complex problems and developing ideas. I am very thankful for his invaluable guidance and patience, and for introducing me to the world of research.

I also wish to thank Prof. Alina Ene, Prof. Huy L. Nguyen and Andrew Suh for the joint work on our published paper.

Finally, the generous financial help of the Open University of Israel is gratefully acknowledged.

Contents

1 Introduction

Submodular functions are a wide class of functions capturing the intuitive notion of diminishing returns. As diminishing returns occurs naturally in many scenarios, the optimization of submodular functions subject to combinatorial constraints has found many applications in diverse fields, including machine learning [20, 30, 39], social networks [31, 34] and algorithmic game theory [21, 40].

In the context of many of the above applications, it is desirable for the submodular optimization algorithm to be a (semi-)streaming algorithm because the input is either very large (e.g., the friendships graph of a social network) or naturally occurs at the form of a long stream (e.g., summarizations of the frames generated by a surveillance camera). In response to this need, Badanidiyuru et al. [5] and Chakrabarti and Kale [13] developed two semi-streaming algorithms for the maximization of non-negative submodular functions that are also *monotone*. A set function $f\colon 2^V \to \mathbb{R}$ over a ground set V is submodular if for every two sets $A \subseteq B \subseteq V$ and element $u \in V \setminus B$ it holds that

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \enspace ,$$

and it is monotone if $f(A) \leq f(B)$ for every two sets $A \subseteq B \subseteq V$. The algorithm of Badanidiyuru et al. [5] maximizes non-negative monotone submodular functions subject to a cardinality constraint up to an approximation ratio of 1/2; and the algorithm of Chakrabarti and Kale [13] achieves a 1/4-approximation for the maximization of the same kind of functions subject to a more general class of constraints known as matroid constraints.

Note again that the two above algorithms work only for monotone submodular functions. To handle non-monotone submodular functions, it is usually necessary to use randomness.[1] The algorithm of Chakrabarti and Kale [13] is local-search based, and easily integrates with randomness, which has lead to multiple works adapting it to non-monotone functions [16, 25, 36]. The best of these adaptations achieves an approximation ratio of $1/(3 + 2\sqrt{2}) \approx 0.1715$ for maximizing a non-negative (not necessarily monotone) submodular function subject to a matroid constraint [25]. In contrast, the algorithm of Badanidiyuru et al. [5] for cardinality constrains, which is known as Sieve-Streaming, is based on a thresholding technique. In a nutshell, the algorithm picks a threshold and then selects every element whose marginal contribution to the current solution of the algorithm exceeds this threshold, until the solution gets to the maximum cardinality allowed. The analysis of Sieve-Streaming then handles in two very different ways the case in which the solution grew all the way to the maximum cardinality allowed and the case in which this did not happen. Unfortunately, most natural ways to add randomness to Sieve-Streaming result in input instances for which both these cases might happen with a non-zero probability, which makes the analysis break down. Due to this hurdle, prior to this work, no random adaptation of Sieve-Streaming managed to improve over the 0.1715-approximation of [25] despite the significant advantage of Sieve-Streaming over the local search approach of Chakrabarti and Kale [13] in the context of monotone submodular functions.[2]

[1]Some works use alternative techniques involving the maintenance of multiple solutions. However, it is often natural to view these solutions as the support of a distribution of solutions. See [6] for an explicit example of this point of view.

[2]Chekuri et al. [16] claimed an improved approximation ratio of 0.212 for cardinality constraints based on

In this thesis, we give a streaming algorithm[3] (named StreamProcess) that is based on the technique of maintaining multiple solutions and treating them as the support of a distribution. Our algorithm is deterministic and combinatorial, has a relatively simple analysis and enjoys a low space complexity of $O(k\varepsilon^{-2}\log\alpha^{-1})$ and a low update time of $O(\varepsilon^{-2}\log(k/\alpha))$ marginal value computations per element. Furthermore, the algorithm performs a single pass over the stream and outputs sets of size $O(k/\varepsilon)$ that can be post-processed using *any offline algorithm* for submodular maximization. The post-processing is itself quite straightforward: we simply run the offline algorithm on the output set to obtain a solution of size at most k. We show that, if the offline algorithm achieves α-approximation, then we obtain $\left(\frac{\alpha}{1+\alpha}-\varepsilon\right)$-approximation. One can note that if we post-process using an exact (exponential time) algorithm, we obtain $(\frac{1}{2}-\varepsilon)$-approximation. This matches the inapproximability result proven by [27] for the special case of a monotone objective function, which shows that any streaming algorithm guaranteeing $(\frac{1}{2}+\varepsilon)$-approximation for some positive constant ε must use $\Omega(n/k^3)$ space. Furthermore, we show that in the non-monotone case the lower bound on the space complexity required by any algorithm can be improved to $\Omega(n)$.[4] Thus, we essentially settle the approximability of the problem if exponential-time computation is allowed.

The best (polynomial-time) approximation guarantee that is currently known in the offline setting is $\alpha = 0.385$ [7]. If we post-process using this algorithm, we obtain 0.2779-approximation in polynomial time, improving over the previously best polynomial-time approximation of 0.1715 due to [24]. The offline algorithm of [7] is based on the multilinear extension, and thus is quite slow. One can obtain a more efficient overall algorithm by using the combinatorial random Greedy algorithm of [10] that achieves $\frac{1}{e}$-approximation. Furthermore, any existing heuristic for the offline problem can be used for post-processing, exploiting their effectiveness beyond the worst case.

Variants of our algorithm. As mentioned above, every algorithm for non-monotone submodular maximization includes a randomized component. Perhaps the most straightforward way to introduce a randomized component into the single-threshold Greedy algorithm is to use the multilinear extension as the objective function and include only fractions of elements in the solution (which corresponds to including the elements in the solution only with some bounded probability). This has the advantage of keeping the algorithm almost deterministic (in fact, completely deterministic when the multilinear extension can be evaluated deterministically), which allows for a relatively simple analysis of the algorithm and a low space complexity of $O(k\log\alpha^{-1}/\varepsilon^2)$. However, the time complexity of an algorithm obtained via this approach depends on the complexity of evaluating the multilinear extension, which in general can be quite high. In Section 5, we describe and analyze a variant of our algorithm (named StreamProcessExtension) which is based on the multilinear

such an adaptation of Sieve-Streaming, but an error was later found in the proof of this improved ratio [15]. See Appendix A for more detail.

[3]Formally, all the algorithms we present are semi-streaming algorithms, i.e., their space complexity is nearly linear in k. Since this is unavoidable for algorithms designed to output an approximate solution (as opposed to just estimating the value of the optimal solution), we ignore the difference between streaming and semi-streaming algorithms in this thesis and use the two terms interchangeably.

[4]This result is a simple adaptation of a result due to Buchbinder et al. [11]. For completeness, we include the proof in Appendix B.

extension.

Although it is most common to discuss submodularity in the context of set functions, optimization of continuous functions that have the diminishing returns (DR) property has many applications in real-world problems. In Section 6 we present a variant of our main algorithm for maximizing a subclass of continuous submodular functions, known as DR-submodular functions, subject to a support cardinality constrained.

It is also worth mentioning that while the asymptotic approximation guarantees of all three variants of our algorithm are identical given a black box offline α-approximation algorithm, they might differ when the offline algorithm has additional properties. For example, the approximation ratio of the offline algorithm might depend on the ratio between k and the number of elements in its input (see [10] for an example of such an algorithm). Given such an offline algorithm, it might be beneficial to set the parameters controlling the number of elements passed to the offline algorithm so that only a moderate number of elements is passed, which is a regime in which the three variants of our algorithm produce different approximation guarantees. Hence, one of the variants of our algorithm might end up having a better approximation guarantee if future research yields offline algorithms with relevant properties.

Results and techniques that are used throughout this thesis were published in our joint paper with Alina Ene, Huy L. Nguyen and Andrew Suh [1]. In particular, the algorithm in Section 5 is one of the main results in [1]. However, in Sections 3 and 4 we present an improved algorithm that was only submitted in a later journal version of the paper (available in [2]), Furthermore, Section 6 contains a new variant of this algorithm for maximizing DR-submodular functions, which does not appear so far in any paper that was published or submitted for publication. In contrast, the published paper [1] contains an additional algorithm (which is not a part of this thesis) that involves true randomization and gets the same approximation ratio guarantee and update time as our main algorithm, but has a slightly worse space complexity of $\tilde{O}(k/\varepsilon^3)$.

1.1 Additional Related Work

The problem of maximizing a non-negative monotone submodular function subject to a cardinality or a matroid constraint was studied (in the offline model) already in the 1970's. In 1978, Nemhauser et al. [38] and Fisher et al. [28] showed that a natural greedy algorithm achieves an approximation ratio of $1 - 1/e \approx 0.632$ for this problem when the constraint is a cardinality constraint and an approximation ratio of $1/2$ for matroid constraints. The $1 - 1/e$ approximation ratio for cardinality constraints was shown to be optimal already on the same year by Nemhauser and Wolsey [37], but the best possible approximation ratio for matroid constraints was open for a long time. Only a decade ago, Calinescu et al. [12] managed to show that a more involved algorithm, known as "continuous greedy", can achieve $(1 - 1/e)$-approximation for this type of constraint, which is tight since matroid constraints generalize cardinality constraints.

Unlike the natural greedy algorithm, continuous greedy is a randomized algorithm, which raised an interesting question regarding the best possible approximation ratio for matroid constraints that can be achieved by a deterministic algorithm. Very recently, Buchbinder et al. [9] made a slight step towards answering this question. Specifically, they described a deterministic algorithm for maximizing a monotone submodular function subject to a matroid constraint whose approximation ratio is 0.5008. This algorithm shows that the $1/2$-approximation of the greedy algorithm is not the right answer for the above mentioned question.

Many works have studied also the offline problem of maximizing a non-negative (not necessarily monotone) submodular function subject to a cardinality or matroid constraint [7, 10, 19, 22, 23, 26]. The most recent of these works achieves an approximation ratio of 0.385 for both cardinality and matroid constraints [7]. In contrast, it is known that no polynomial time algorithm can achieve an approximation ratio of 0.497 for cardinality constraints or 0.478 for matroid constraints [29].

The study of streaming algorithms for submodular maximization problems is related to the study of online algorithms for such problems. A partial list of works on algorithms of the last kind includes [4, 8, 11, 14, 32, 35].

2 Preliminaries

Basic notation. Let V denote a finite ground set of size $n := |V|$. For two vectors $x, y \in \mathbb{R}^V$, we let $x \vee y$ and $x \wedge y$ be the vectors such that $(x \vee y)_e = \max\{x_e, y_e\}$ and $(x \wedge y)_e = \min\{x_e, y_e\}$ for all $e \in V$. For a set $S \subseteq V$, we let $\mathbf{1}_S$ denote the indicator vector of S, i.e., the vector that has 1 in every coordinate $e \in S$ and 0 in every coordinate $e \in V \setminus S$. Given an element $e \in V$, we use $\mathbf{1}_e$ as a shorthand for $\mathbf{1}_{\{e\}}$.

Submodular functions. In this thesis, we consider the problem of maximizing a non-negative submodular function subject to a cardinality constraint. A set function $f\colon 2^V \to \mathbb{R}$ is submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all subsets $A, B \subseteq V$. Additionally, given a set $S \subseteq V$ and an element $e \in V$, we use $f(e \mid S)$ to denote the marginal contribution of e to S with respect to the set function f, i.e., $f(e \mid S) = f(S \cup \{e\}) - f(S)$.

Multilinear extension. In Section 5, we present an algorithm that makes use of a standard continuous extension of submodular functions, known as the *multilinear extension*. To define this extension, we first need to define the random set $\mathtt{R}(x)$. For every vector $x \in [0,1]^V$, $\mathtt{R}(x)$ is defined as a random subset of V that includes every element $e \in V$ with probability x_e, independently. The multilinear extension F of f is now defined for every $x \in [0,1]^V$ by

$$F(x) = \mathbb{E}\Big[f\Big(\mathtt{R}(x)\Big)\Big] = \sum_{A \subseteq V} f(A) \cdot \Pr[\mathtt{R}(x) = A] = \sum_{A \subseteq V} \left(f(A) \cdot \prod_{e \in A} x_e \cdot \prod_{e \notin A} (1 - x_e)\right) .$$

One can observe from the definition that F is indeed a multilinear function of the coordinates of x, as suggested by its name. Thus, if we use the shorthand $\partial_e F(x)$ for the first partial derivative $\frac{\partial F(x)}{\partial x_e}$ of the multilinear extension F, then $\partial_e F(x) = F(x \vee \mathbf{1}_e) - F\Big(x \wedge \mathbf{1}_{V \setminus \{e\}}\Big)$.

3 Simplified Algorithm

The properties of our main algorithm (StreamProcess—the first variant discussed above) are summarized by the following theorem.

Theorem 1. *Assume there exists an α-approximation offline algorithm* OfflineAlg *for maximizing a non-negative submodular function subject to a cardinality constraint whose space complexity is nearly linear in the size of the ground set. Then, for every constant $\varepsilon \in (0, 1]$, there exists an $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation semi-streaming algorithm for maximizing a non-negative submodular function subject to a cardinality constraint. The algorithm stores $O(k\varepsilon^{-2})$ elements and makes $O(\varepsilon^{-2} \log k)$ marginal value computations while processing each arriving element.*[5] *Furthermore, if* OfflineAlg *is deterministic, then so is the algorithm that we get.*

In this section, we introduce a simplified version of the algorithm used to prove Theorem 1. This simplified version (given as Algorithm 1) captures our main new ideas, but makes the simplifying assumption that it has access to an estimate τ of $f(\text{OPT})$ obeying $(1 - \varepsilon/2) \cdot f(\text{OPT}) \leq \tau \leq f(\text{OPT})$. Such an estimate can be produced using well-known techniques, at the cost of a slight increase in the space complexity and update time of the algorithm. More specifically, in Section 4 we formally show that one such technique due to [33] can be used for that purpose, and that it increases the space complexity and update of the algorithm only by factors of $O(\varepsilon^{-1} \log \alpha^{-1})$ and $O(\varepsilon^{-1} \log(k/\alpha))$, respectively.

Algorithm 1 gets two parameters: the approximation ratio α of OfflineAlg and an integer $p \geq 1$. The algorithm maintains p solutions $S_1, S_2, \ldots, S_p$. All these solutions start empty, and the algorithm may add each arriving element to at most one of them. Specifically, when an element e arrives, the algorithm checks whether there exists a solution S_i such that (1) S_i does not already contain k elements, and (2) the marginal contribution of e with respect to S_i exceeds the threshold of $c\tau/k$ for $c = \alpha/(1 + \alpha)$. If there is such a solution, the algorithm adds e to it (if there are multiple such solutions, the algorithm adds e to an arbitrary one of them); otherwise, the algorithm discards e. After viewing all of the elements, Algorithm 3 generates one more solution S_o by executing OfflineAlg on the union of the p solutions $S_1, S_2, \ldots, S_p$. The output of the algorithm is then the best solution among the $p + 1$ generated solutions.

Algorithm 1: StreamProcess (simplified) (p, α)

```
1 Let $c \leftarrow \frac{\alpha}{1+\alpha}$.
2 for $i = 1$ to $p$ do Let $S_i \leftarrow \varnothing$.
3 for each arriving element $e$ do
4     if there exists an integer $1 \leq i \leq p$ such that $|S_i| < k$ and $f(e \mid S_i) \geq \frac{c\tau}{k}$ then
5         Update $S_i \leftarrow S_i \cup \{e\}$ (if there are multiple options for $i$, pick an arbitrary
          one).
6 Find another feasible solution $S_o \subseteq \bigcup_{i=1}^{p} S_i$ by running OfflineAlg with $\bigcup_{i=1}^{p} S_i$ as
   the ground set.
7 return the solution maximizing $f$ among $S_o$ and the $p$ solutions $S_1, S_2, \ldots, S_p$.
```

Since Algorithm 1 stores elements only in the $p + 1$ solutions it maintains, and all these

[5] Formally, the number of elements stored by the algorithm and the number of marginal value computations also depend on $\log \alpha^{-1}$. Since α is typically a positive constant, or at least lower bounded by a positive constant, we omit this dependence from the statement of the theorem.

solutions are feasible (and thus, contain only k elements), we immediately get the following observation. Note that this observation implies (in particular) that Algorithm 1 is a semi-streaming algorithm for a constant p when the space complexity of OfflineAlg is nearly linear.

Observation 2. *Algorithm 1 stores at most $O(pk)$ elements, and makes at most p marginal value calculations while processing each arriving element.*

We now divert our attention to analyzing the approximation ratio of Algorithm 1. Let us denote by $\hat{S}_i$ the final set S_i (i.e., the content of this set when the stream ends), and consider two cases. The first (easy) case is when at least one of the solutions $S_1, S_2, \ldots, S_p$ reaches a size of k. The next lemma analyzes the approximation guarantee of Algorithm 1 in this case.

Lemma 3. *If there is an integer $1 \leq i \leq p$ such that $|\hat{S}_i| = k$, then the output of Algorithm 1 has value of at least $\frac{\alpha\tau}{1+\alpha}$.*

Proof. Denote by $e_1, e_2, \ldots, e_k$ the elements of $\hat{S}_i$ in the order of their arrival. Using this notation, the value of $f(\hat{S}_i)$ can be written as follows.

$$f(\hat{S}_i) = f(\varnothing) + \sum_{j=1}^{k} f\Big(e_j \mid \{e_1, e_2, \ldots, e_{j-1}\}\Big) \geq \sum_{i=1}^{k} \frac{\alpha\tau}{(1+\alpha)k} = \frac{\alpha\tau}{1+\alpha} \enspace ,$$

where the inequality holds since the non-negativity of f implies $f(\varnothing) \geq 0$ and Algorithm 1 adds an element e_j to S_i only when $f\Big(e_j \mid \{e_1, e_2, \ldots, e_{j-1}\}\Big) \geq \frac{c\tau}{k} = \frac{\alpha\tau}{(1+\alpha)k}$. The lemma now follows since the solution outputted by Algorithm 1 is at least as good as S_i. □

Consider now the case in which no set S_i reaches the size of k. In this case our objective is to show that at least one of the solutions computed by Algorithm 1 has a large value. Lemmata 5 and 6 lower bound the value of the average solution among $S_1, S_2, \ldots, S_p$ and the solution S_o, respectively. The proof of Lemma 5 uses the following known lemma.

Lemma 4 (Lemma 2.2 from [10])**.** *Let $f\colon 2^V \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Denote by $A(p)$ a random subset of A where each element appears with probability at most p (not necessarily independently). Then, $\mathbb{E}[f(A(p))] \geq (1-p) \cdot f(\varnothing)$.*

Let $O = \mathrm{OPT} \setminus \bigcup_{i=1}^{p} \hat{S}_i$, and let $b = |O|/k$.

Lemma 5. *If $|\hat{S}_i| < k$ for every integer $1 \leq i \leq p$, then, for every fixed set $A \subseteq V$, $p^{-1} \cdot \sum_{i=1}^{p} f(\hat{S}_i \cup A) \geq (1 - p^{-1}) \cdot f(O \cup A) - \alpha b\tau/(1+\alpha)$.*

Proof. The elements in O were rejected by Algorithm 1. Since no set S_i reaches a size of k, this means that the the marginal contribution of the elements of O with respect to every set S_i at the time of their arrival was smaller than $c\tau/k$. Moreover, since Algorithm 1 only adds elements to its solutions during its execution, the submodularity of f guarantees that the marginals of the elements of O are below this threshold also with respect to $\hat{S}_1 \cup A, \hat{S}_2 \cup A, \ldots, \hat{S}_p \cup A$. More formally, we get

$$f(e \mid \hat{S}_i \cup A) < \frac{c\tau}{k} = \frac{\alpha\tau}{k(1+\alpha)} \quad \forall\, e \in O \text{ and integer } 1 \leq i \leq p \enspace .$$

Using the submodularity of f again, this implies that for every integer $1 \leq i \leq p$

$$f\Big(\hat{S}_i \cup O \cup A\Big) \leq f(\hat{S}_i \cup A) + \sum_{e \in O} f(e \mid \hat{S}_i \cup A) \leq f(\hat{S}_i \cup A) + |O| \cdot \frac{\alpha\tau}{k(1+\alpha)} = f(\hat{S}_i \cup A) + \frac{\alpha b \tau}{1+\alpha} \enspace .$$

Adding up the above inequalities (and dividing by p), we get

$$p^{-1} \cdot \sum_{i=1}^{p} f(\hat{S}_i \cup A) \geq p^{-1} \cdot \sum_{i=1}^{p} f\Big(\hat{S}_i \cup O \cup A\Big) - \frac{\alpha b \tau}{1+\alpha} \enspace .$$

We now note that $p^{-1} \cdot \sum_{i=1}^{p} f\Big(\hat{S}_i \cup O \cup A\Big)$ can be viewed as the expected value of a non-negative submodular function $g(S) = f(S \cup O \cup A)$ over a random set S that is equal to every one of the sets $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$ with probability $1/p$. Since the sets $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$ are disjoint, S contains every element with probability at most $1/p$, and thus, by Lemma 4,

$$p^{-1} \cdot \sum_{i=1}^{p} f\Big(\hat{S}_i \cup O \cup A\Big) = \mathbb{E}[g(S)] \geq (1 - p^{-1}) \cdot g(\varnothing) = (1 - p^{-1}) \cdot f(O \cup A) \enspace .$$

The lemma now follows by combining the last two inequalities. $\square$

As mentioned above, our next step is to get a lower bound on the value of $f(S_o)$. One easy way to get such a lower bound is to observe that $\mathrm{OPT} \setminus O$ is a subset of $\bigcup_{i=1}^{p} \hat{S}_i$ of size at most k, and thus, is a feasible solution for the instance faced by the algorithm OfflineAlg used to find S_o; which implies $\mathbb{E}[f(S_o)] \geq \alpha \cdot f(\mathrm{OPT} \setminus O)$ since OfflineAlg is an α-approximation algorithm. The following lemma proves a more involved lower bound by considering the vectors $(b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O}$ as feasible fractional solutions for the same instance (using rounding methods such as Pipage Rounding or Swap Rounding [12, 18], such feasible factional solutions can be converted into integral feasible solutions of at least the same value).

Lemma 6. *If $|\hat{S}_i| < k$ for every integer $1 \leq i \leq p$, then $\mathbb{E}[f(S_o)] \geq \alpha b \tau (1 - p^{-1} - \alpha b/(1 + \alpha)) + \alpha(1 - b) \cdot f(\mathrm{OPT} \setminus O)$.*

Proof. Fix some integer $1 \leq i \leq p$, and consider the vector $(b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O}$. Clearly,

$$\Big\|(b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O}\Big\|_1 \leq b \cdot |\hat{S}_i| + |\mathrm{OPT} \setminus O| \leq |O| + |\mathrm{OPT} \setminus O| = |\mathrm{OPT}| \leq k \enspace ,$$

where the second inequality holds by the definition of b since $|\hat{S}_i| < k$. Given the last property of the vector $(b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O}$, standard rounding techniques such as Pipage Rounding [12] and Swap Rounding [18] can be used to produce from this vector a set $A_i \subseteq \hat{S}_i \cup (\mathrm{OPT} \setminus O) \subseteq \bigcup_{i=1}^{p} \hat{S}_i$ of size at most k such that $f(A_i) \geq F((b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O})$. Since S_o is produced by the algorithm OfflineAlg whose approximation ratio is α and A_i is one possible output for this algorithm, this implies $\mathbb{E}[f(S_o)] \geq \alpha \cdot f(A_i) \geq \alpha \cdot F((b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O})$. Furthermore, by averaging this equation over all the possible values of i, we get

$$\begin{aligned}
\mathbb{E}[f(S_o)] &\geq \alpha p^{-1} \cdot \sum_{i=1}^{p} F((b\mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O}) \\
&\geq \alpha p^{-1} \cdot \left[(1-b) \cdot \sum_{i=1}^{p} f(\mathrm{OPT} \setminus O) + b \cdot \sum_{i=1}^{p} f(\hat{S}_i \cup (\mathrm{OPT} \setminus O))\right] \\
&\geq \alpha(1-b) \cdot f(\mathrm{OPT} \setminus O) + \alpha b[(1 - p^{-1}) \cdot f(\mathrm{OPT}) - \alpha b \tau/(1+\alpha)] \enspace ,
\end{aligned}$$

where the second inequality holds since the submodularity of f implies that $F((t \cdot \mathbf{1}_{\hat{S}_i}) \vee \mathbf{1}_{\mathrm{OPT} \setminus O})$ is a concave function of t within the range $[0, 1]$ (and b falls is inside this range), and the last inequality holds by Lemma 5 (for $A = \mathrm{OPT} \setminus O$). The lemma now follows by recalling that $f(OPT) \geq \tau$. □

Combining the guarantees of the last two lemmata we can now obtain a lower bound on the value of the solution of Algorithm 1 in the case that $|S_i| < k$ for every integer $1 \leq i \leq p$ which is independent of b.

Corollary 7. *If $|\hat{S}_i| < k$ for every integer $1 \leq i \leq p$, then the output of Algorithm 1 has value of at least $\left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau$.*

Proof. The corollary follows immediately from the non-negativity of f when $p = 1$. Thus, we may assume $p \geq 2$ in the rest of the proof.

Since Algorithm 1 outputs the best solution among the $p + 1$ solutions it creates, we get by Lemma 5 (for $A = \varnothing$) and Lemma 6 that the value of the solution it outputs is at least

$$\begin{aligned}
\max\left\{p^{-1}\sum_{i=1}^{p} f(\hat{S}_i), f(S_o)\right\} \geq{}& \max\{(1 - p^{-1}) \cdot f(O) - \alpha b\tau/(1 + \alpha), \\
&\alpha b\tau(1 - p^{-1} - \alpha b/(1 + \alpha)) + \alpha(1 - b) \cdot f(\mathrm{OPT} \setminus O)\} \\
\geq{}& \frac{\alpha(1 - b)}{\alpha(1 - b) + 1 - p^{-1}} \cdot \left[(1 - p^{-1}) \cdot f(O) - \alpha b\tau/(1 + \alpha)\right] \\
&+ \frac{1 - p^{-1}}{\alpha(1 - b) + 1 - p^{-1}} \cdot \left[\alpha b\tau(1 - p^{-1} - \alpha b/(1 + \alpha)) + \alpha(1 - b) \cdot f(\mathrm{OPT} \setminus O)\right] .
\end{aligned}$$

Note now that the submodularity and non-negativity of f guarantee together $f(O) + f(\mathrm{OPT} \setminus O) \geq f(\mathrm{OPT}) \geq \tau$. Using this fact and the non-negativity of f, the previous inequality yields

$$\begin{aligned}
&\frac{\max\left\{p^{-1}\sum_{i=1}^{p} f(\hat{S}_i), f(S_o)\right\}}{\alpha\tau} \geq \\
&\qquad \max\left\{0, \frac{(1 - b)(1 - p^{-1}) - \alpha b(1 - b)/(1 + \alpha) + b(1 - p^{-1})(1 - p^{-1} - \alpha b/(1 + \alpha))}{\alpha(1 - b) + 1 - p^{-1}}\right\} \\
&\quad \geq \max\left\{0, \frac{(1 - b)(1 - p^{-1}) - \alpha b(1 - b)/(1 + \alpha) + b(1 - p^{-1})(1 - p^{-1} - \alpha b/(1 + \alpha))}{\alpha(1 - b) + 1}\right\} \\
&\quad \geq \frac{(1 - b) - \alpha b(1 - b)/(1 + \alpha) + b(1 - \alpha b/(1 + \alpha))}{\alpha(1 - b) + 1} - 2p^{-1} ,
\end{aligned}$$

where the last two inequalities hold since $\alpha \in (0, 1]$, $b \in [0, 1]$ and $p \geq 2$. Simplifying the last inequality, we get

$$\begin{aligned}
\frac{\max\left\{p^{-1}\sum_{i=1}^{p} f(\hat{S}_i), f(S_o)\right\}}{\alpha\tau} &\geq \frac{(1 - b)(1 + \alpha) - \alpha b(1 - b) + b(1 + \alpha - \alpha b)}{(1 + \alpha)[\alpha(1 - b) + 1]} - 2p^{-1} \\
&= \frac{1 + \alpha(1 - b)}{(1 + \alpha)[\alpha(1 - b) + 1]} - 2p^{-1} = \frac{1}{1 + \alpha} - 2p^{-1} .
\end{aligned}$$

The corollary now follows by rearranging this inequality (and recalling that $\alpha \in (0, 1]$). □

Note that the guarantee of Corollary 7 (for the case it considers) is always weaker than the guarantee of Lemma 3. Thus, we can summarize the results we have proved so far using the following proposition.

Proposition 8. *Algorithm 1 stores $O(pk)$ elements, makes at most p marginal value calculations while processing each arriving element and its output set has an expected value of at least $\left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau$.*

Using the last proposition, we can now prove the following theorem. As discussed at the beginning of the section, in Section 4 we explain how the assumption that τ is known can be dropped at the cost of a slight increase in the number of of elements stored by the algorithm and its update time, which yields Theorem 1.

Theorem 9. *Assume there exists an α-approximation offline algorithm* OfflineAlg *for maximizing a non-negative submodular function subject to cardinality constraint whose space complexity is nearly linear in the size of the ground set. Then, for every constant $\varepsilon \in (0, 1]$, there exists a semi-streaming algorithm that assumes access to an estimate τ of $f(\text{OPT})$ obeying $(1 - \varepsilon/2) \cdot f(\text{OPT}) \leq \tau \leq f(\text{OPT})$ and provides $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation for the problem of maximizing a non-negative submodular function subject to cardinality constraint. This algorithm stores $O(k\varepsilon^{-1})$ elements and calculates $O(\varepsilon^{-1})$ marginal values while processing each arriving element.*

Proof. Consider the algorithm obtained from Algorithm 1 by setting $p = \lceil 4/\varepsilon \rceil$. By Proposition 8 and the non-negativity of f, this algorithm stores only $O(pk) = O(k\varepsilon^{-1})$ elements, calculates only $p = O(\varepsilon^{-1})$ marginal values while processing each arriving element, and the expected value of its output set is at least

$$\max\left\{0, \left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau\right\} \geq \left(\frac{\alpha}{\alpha+1} - \frac{\varepsilon}{2}\right) \cdot (1 - \varepsilon/2) \cdot f(\text{OPT}) \geq \left(\frac{\alpha}{\alpha+1} - \varepsilon\right) \cdot f(\text{OPT}) ,$$

where the first inequality holds since $p \geq 4/\varepsilon$ and τ obeys, by assumption, $\tau \geq (1 - \varepsilon/2) \cdot f(\text{OPT})$. □

4 Complete Algorithm

In this section, we explain how one can drop the assumption from Section 3 that the algorithm has access to an estimate τ of $f(\text{OPT})$. This completes the proof of Theorem 1.

Technically, we analyze in this section the variant of Algorithm 1 given as Algorithm 2. It gets the same two parameters α and p received by Algorithm 1 plus an additional parameter $\varepsilon' \in (0, 1)$ controlling the guaranteed quality of the output. The algorithm is based on a technique originally due to Kazemi et al. [33]. Throughout its execution, Algorithm 2 tracks in m a lower bound on the value of $f(\text{OPT})$. The algorithm also maintains a set $T = \{(1+\varepsilon')^i \mid m/(1+\varepsilon') \leq (1+\varepsilon')^i \leq mk/c\}$ of values that, given the current value of the lower bound m, are (1) possible estimates for $f(\text{OPT})$ at the current point and (2) are not so large that they dwarf this lower bound (of course, T includes only a subset of the possible estimates obeying these requirements). For every estimate τ in T, the algorithm maintains p sets $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$. We note that the set of solutions maintained is updated every time that T is updated (which happens after every update of m). Specifically, whenever a new value τ is added to T, the algorithm instantiate p new sets $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$, and whenever a value τ is dropped from T, the algorithm deletes $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$.

While a value τ remains in T, Algorithm 2 maintains the sets $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$ in exactly the same way that Algorithm 1 maintains its sets $S_1, S_2, \ldots, S_p$ given this value τ as an estimate for $f(\mathrm{OPT})$. Moreover, we show below that if τ remains in T when the algorithm terminates, then the contents of $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$ when the algorithm terminates are equal to the contents of the sets $S_1, S_2, \ldots, S_p$ when Algorithm 1 terminates after executing with this τ as the estimate for $f(\mathrm{OPT})$. Thus, one can view Algorithm 2 as parallel execution of Algorithm 1 for many estimates of $f(\mathrm{OPT})$ at the same time. After viewing the last element, Algorithm 2 calculates for every $\tau \in T$ an output set $\bar{S}_\tau$ based on the sets $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$ in the same way Algorithm 1 does that, and then outputs the best output set computed for any $\tau \in T$.

Algorithm 2: StreamProcess $(p, \alpha, \varepsilon')$

```
1  Let $c \leftarrow \frac{\alpha}{1+\alpha}$.
2  Let $m \leftarrow f(\varnothing)$ and $T \leftarrow \{(1+\varepsilon')^h \mid m/(1+\varepsilon') \leq (1+\varepsilon')^h \leq mk/c\}$.
3  for each arriving element $e$ do
4      Let $m' \leftarrow \max\{f(\{e\}), \max_{\tau \in T, 1 \leq i \leq p} f(S_i^\tau)\}$.
5      if $m < m'$ then
6          Update $m \leftarrow m'$, and then $T \leftarrow \{(1+\varepsilon')^h \mid m/(1+\varepsilon') \leq (1+\varepsilon')^h \leq mk/c\}$.
7          Delete $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$ for every value $\tau$ removed from $T$ in Line 6.
8          Initialize $S_i^\tau \leftarrow \varnothing$ for every value $\tau$ added to $T$ in Line 6 and integer
            $1 \leq i \leq p$.
9      for every $\tau \in T$ do
10         if there exists an integer $1 \leq i \leq p$ such that $|S_i^\tau| < k$ and $f(e \mid S_i^\tau) \geq \frac{c\tau}{k}$
            then
11             Update $S_i^\tau \leftarrow S_i^\tau \cup \{e\}$ (if there are multiple options for $i$, pick an
                arbitrary one).
12 for every $\tau \in T$ do
13     Find another feasible solution $S_o^\tau \subseteq \bigcup_{i=1}^p S_i^\tau$ by running OfflineAlg with
        $\bigcup_{i=1}^p S_i^\tau$ as the ground set.
14     Let $\bar{S}_\tau$ be the better solution among $S_o^\tau$ and the $p$ solutions $S_1^\tau, S_2^\tau, \ldots, S_p^\tau$.
15 return the best solution among $\{\bar{S}_\tau\}_{\tau \in T}$, or the empty set if $T = \varnothing$.
```

We begin the analysis of Algorithm 2 by providing a basic lower bound on the value of each set S_i^τ. This lower bound can be viewed as a generalized counterpart of Lemma 3.

Lemma 10. *At every point during the execution of the algorithm, for every $\tau \in T$ and $1 \leq i \leq p$ it holds that $f(S_i^\tau) \geq \frac{\alpha\tau \cdot |S_i^\tau|}{(1+\alpha)k}$.*

Proof. Denote by $e_1, e_2, \ldots, e_{|S_i^\tau|}$ the elements of S_i^τ in the order of their arrival. Using this notation, the value of $f(S_i^\tau)$ can be written as follows.

$$f(S_i^\tau) = f(\varnothing) + \sum_{j=1}^{|S_i^\tau|} f\Big(e_j \mid \{e_1, e_2, \dots, e_{j-1}\}\Big) \geq \sum_{i=1}^{|S_i^\tau|} \frac{\alpha\tau}{(1+\alpha)k} = \frac{\alpha\tau \cdot |S_i^\tau|}{(1+\alpha)k} \enspace,$$

where the inequality holds since the non-negativity of f implies $f(\varnothing) \geq 0$ and Algorithm 2 adds an element e_j to S_i^τ only when $f\Big(e_j \mid \{e_1, e_2, \dots, e_{j-1}\}\Big) \geq \frac{c\tau}{k} = \frac{\alpha\tau}{(1+\alpha)k}$. □

Using the last lemma we can now prove the following observation, which upper bounds the size of each set S_i^τ maintained by Algorithm 2, and thus, serves as a first step towards bounding the space complexity of this algorithm.

Observation 11. *At the end of the every iteration of Algorithm 2, the size of the set S_i^τ is at most $mk/(c\tau) + 1$ for every $\tau \in T$ and integer $1 \leq i \leq p$.*

Proof. Let $\bar{S}_i^\tau$ denote the content of the set S_i^τ at the beginning of the iteration. Since at most a single element is added to S_i^τ during the iteration, we get via Lemma 10 that the value of the set $\bar{S}_i^\tau$ is at least

$$\frac{\alpha\tau \cdot |\bar{S}_i^\tau|}{(1+\alpha)k} \geq \frac{\alpha\tau \cdot (|S_i^\tau| - 1)}{(1+\alpha)k} = \frac{c\tau \cdot (|S_i^\tau| - 1)}{k} \enspace.$$

The way in which Algorithm 2 updates m guarantees that immediately after the update of m at the beginning of the iteration, the value of m was at least as large as the value of $\bar{S}_i^\tau$, and thus, we get

$$m \geq \frac{c\tau \cdot (|S_i^\tau| - 1)}{k} \enspace,$$

which implies the observation. □

We are now ready to bound the space complexity and update time of Algorithm 2.

Lemma 12. *Algorithm 2 can be implemented so that it stores $O(pk(\varepsilon')^{-1} \log \alpha^{-1})$ elements and makes only $O(p(\varepsilon')^{-1} \log(k/\alpha))$ marginal value calculations when processing each arriving element.*

Proof. We begin the proof with two technical calculations. First, note that the number of estimates in T is upper bounded at all times by

$$1 + \log_{1+\varepsilon'}\left(\frac{km/c}{m/(1+\varepsilon')}\right) = 2 + \frac{\ln k - \ln c}{\ln(1+\varepsilon')} \leq 2 + \frac{\ln k - \ln c}{2\varepsilon'/3} = O((\varepsilon')^{-1}(\log k - \log c)) \enspace.$$

Second, note that

$$\begin{aligned}\sum_{\tau \in T}\sum_{i=1}^{p}(|S_i^\tau| - 1) &\leq \sum_{\tau \in T} \min\left\{pk, \frac{pmk}{c\tau}\right\} \leq \sum_{\substack{h \in \mathbb{Z} \\ (1+\varepsilon')^{h+1} \geq m}} \min\left\{pk, \frac{pmk}{c(1+\varepsilon')^h}\right\} \\ &\leq pk \cdot \left|\left\{h \in \mathbb{Z} \mid \frac{m}{1+\varepsilon'} \leq (1+\varepsilon')^h < m/c\right\}\right| + \sum_{h=0}^{\infty} \frac{pk}{(1+\varepsilon')^h} \\ &\leq pk \cdot \left(\log_{1+\varepsilon'}\left(\frac{m/c}{m}\right) + 2\right) + \frac{pk}{1 - 1/(1+\varepsilon')} \leq \frac{pk \cdot (\ln c^{-1} + 2)}{\ln(1+\varepsilon')} + \frac{pk(1+\varepsilon')}{\varepsilon'} \\ &= O(pk(\varepsilon')^{-1} \log c^{-1}) + O(pk(\varepsilon')^{-1}) = O(pk(\varepsilon')^{-1} \log c^{-1}) \enspace,\end{aligned}$$

where the first inequality follows from Observation 11 and the fact that all the sets maintained by Algorithm 2 are of size at most k; and the second inequality follows from the definition of T.

We now observe that, while processing each arriving element, Algorithm 2 makes p marginal value calculations in association with every value $\tau \in T$ and another set of $O(p)$ such calculations that are not associated with any value of T. Thus, the total number of marginal value calculations made by the algorithm during the processing of an arriving element is

$$\begin{aligned} p \cdot |T| + O(p) &= p \cdot O((\varepsilon')^{-1}(\log k - \log c)) + O(p) \\ &= O(p(\varepsilon')^{-1}(\log k - \log c)) = O(p(\varepsilon')^{-1}\log(k/c)) \enspace . \end{aligned}$$

We also note that Algorithm 2 has to store only the elements belonging to S_i^τ for some $\tau \in T$ and integer $1 \leq i \leq p$. Thus, in all these sets together the algorithm stores $O(pk(\varepsilon')^{-1}\log c^{-1})$ elements since

$$\begin{aligned} \sum_{\tau \in T}\sum_{i=1}^{p} |S_i^\tau| = {} & \sum_{\tau \in T}\sum_{i=1}^{p}(|\operatorname{supp}(S_i^\tau)| - 1) + p|T| \\ &= O(pk(\varepsilon')^{-1}\log c^{-1}) + p \cdot O((\varepsilon')^{-1}(\log k - \log c)) = O(pk(\varepsilon')^{-1}\log c^{-1}) \enspace . \end{aligned}$$

To complete the proof of the lemma, it remains to note that $c = \alpha/(1+\alpha) \geq \alpha/2$. □

A this point we divert our attention to analyzing the approximation gaurantee of Algorithm 2. Let us denote by $\hat{m}$ and $\hat{T}$ the final values of m and T, respectively.

Observation 13. *$c \in (0, 1/2]$, and thus, $\hat{T}$ is not empty unless $\hat{m} = 0$.*

Proof. Since $\alpha \in (0, 1]$ and $\alpha/(\alpha + 1)$ is an increasing function of α,

$$c = \frac{\alpha}{\alpha+1} \in \left(\frac{0}{0+1}, \frac{1}{1+1}\right] = (0, 1/2] \enspace . \qquad \square$$

The last observation immediately implies that when $\hat{T}$ is empty, all the singletons have zero values, and the same must be true for every other non-empty set by the submodularity and non-negativity of f. Thus, $\mathrm{OPT} = \varnothing$, which makes the output of Algorithm 2 optimal in the rare case in which $\hat{T}$ is empty. Hence, we can assume from now on that $\hat{T} \neq \varnothing$. The following lemma shows that $\hat{T}$ contains a good estimate for $f(\mathrm{OPT})$ in this case.

Lemma 14. *The set $\hat{T}$ contains a value $\hat{\tau}$ such that $(1 - \varepsilon') \cdot f(\mathrm{OPT}) \leq \hat{\tau} \leq f(\mathrm{OPT})$.*

Proof. Observe that $\hat{m} \geq \max\Big\{f(\varnothing), \max_{e \in V}\Big\{f\Big(\{e\}\Big)\Big\}\Big\}$. Thus, by the submodularity of f,

$$f(\mathrm{OPT}) \leq f(\varnothing) + \sum_{e \in \mathrm{OPT}}[f(\{e\}) - f(\varnothing)] \leq \max\left\{f(\varnothing), \sum_{e \in \mathrm{OPT}} f\Big(\{e\}\Big)\right\} \leq k\hat{m} \leq \frac{k\hat{m}}{c} \enspace .$$

In contrast, one can note that m is equal to the value of f for some feasible solution, and thus, $f(\mathrm{OPT}) \geq \hat{m}$. Since $\hat{T}$ contains all the values of the form $(1 + \varepsilon')^i$ in the range $[\hat{m}/(1 + \varepsilon'), k\hat{m}/c]$, the above inequalities imply that it contains in particular the largest value of this form that is still not larger than $f(\mathrm{OPT})$. Let us denote this value by $\hat{\tau}$. By definition, $\hat{\tau} \leq f(\mathrm{OPT})$. Additionally,

$$\hat{\tau} \cdot (1 + \varepsilon') \geq f(\mathrm{OPT}) \Rightarrow \hat{\tau} \geq \frac{f(\mathrm{OPT})}{1 + \varepsilon'} \geq (1 - \varepsilon') \cdot f(\mathrm{OPT}) \ . \qquad \square$$

Let us now concentrate on the value $\hat{\tau}$ whose existence is guaranteed by Lemma 14, and let $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$ denote the sets maintained by Algorithm 1 when it gets $\hat{\tau}$ as the estimate for $f(\mathrm{OPT})$. Additionally, let us denote by $e_1, e_2, \ldots, e_n$ the elements of V in the order of their arrival, and let e_j be the element whose arrival made Algorithm 2 add $\hat{\tau}$ to T (i.e., $\hat{\tau}$ was added to T by Algorithm 2 while processing e_j). If $\hat{\tau}$ belonged to T from the very beginning of the execution of Algorithm 2, then we define $j = 1$

Lemma 15. *All the sets $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$ maintained by Algorithm 1 are empty immediately prior to the arrival of e_j.*

Proof. If $j = 1$, then lemma is trivial. Thus, we assume $j > 1$ in the rest of this proof. Prior to the arrival of e_j, $\hat{\tau}$ was not part of T. Nevertheless, since $\hat{\tau} \in \hat{T}$, we must have at all times

$$\hat{\tau} \geq \frac{\hat{m}}{1 + \varepsilon'} \geq \frac{m}{1 + \varepsilon'} \ ,$$

where the second inequality holds since the value m only increases over time. Therefore, the reason that $\hat{\tau}$ did not belong to T prior to the arrival of e_j must have been that m was smaller than $c\hat{\tau}/k$. Since m is at least as large as the value of any singleton containing an element already viewed by the algorithm we get, for every two integers $1 \leq t \leq j - 1$ and $1 \leq i \leq p$,

$$\frac{c\hat{\tau}}{k} > f(\{e_t\}) \geq f(\{e_t\} \mid \varnothing) \geq f\Big(e_t \mid \hat{S}_i \wedge \mathbf{1}_{\{e_1, e_2, \ldots, e_{t-1}\}}\Big) \ ,$$

where the second inequality follows from the non-negativity of f and the last from its submodularity. Thus, e_t is not added by Algorithm 1 to any one of the sets $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$, which implies that all these sets are empty at the moment e_j arrives. $\square$

According to the above discussion, from the moment $\hat{\tau}$ gets into T, Algorithm 2 updates the sets $S_1^{\hat{\tau}}, S_2^{\hat{\tau}}, \ldots, S_p^{\hat{\tau}}$ in the same way that Algorithm 1 updates $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$ (note that, once $\hat{\tau}$ gets into T, it remains there for good since $\hat{\tau} \in \hat{T}$). Together with the previous lemma which shows that $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$ are empty just like $S_1^{\hat{\tau}}, S_2^{\hat{\tau}}, \ldots, S_p^{\hat{\tau}}$ at the moment e_j arrives, this implies that the final contents of $S_1^{\hat{\tau}}, S_2^{\hat{\tau}}, \ldots, S_p^{\hat{\tau}}$ are equal to the final contents of $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$, respectively. Since the set $\bar{S}_{\hat{\tau}}$ is computed based on the final contents of $S_1^{\hat{\tau}}, S_2^{\hat{\tau}}, \ldots, S_p^{\hat{\tau}}$ in the same way that the output of Algorithm 1 is computed based on the final contents of $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p$, we get the following corollary.

Corollary 16. *If it is guaranteed that the approximation ratio of Algorithm 1 is at least β when $(1 - \varepsilon') \cdot f(\mathrm{OPT}) \leq \tau \leq f(\mathrm{OPT})$ for some choice of the parameters α and p, then the approximation ratio of Algorithm 2 is at least β as well for this choice of α and p.*

We are now ready to prove Theorem 1.

Theorem 1. *Assume there exists an α-approximation offline algorithm* OfflineAlg *for maximizing a non-negative submodular function subject to a cardinality constraint whose space complexity is nearly linear in the size of the ground set. Then, for every constant $\varepsilon \in (0, 1]$, there exists an $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation semi-streaming algorithm for maximizing a non-negative submodular function subject to a cardinality constraint. The algorithm stores $O(k\varepsilon^{-2} \log \alpha^{-1})$ elements and makes $O(\varepsilon^{-2} \log(k/\alpha))$ marginal value computations while processing each arriving element. Furthermore, if* OfflineAlg *is deterministic, then so is the algorithm that we get.*

Proof. The proof of Theorem 9 shows that Algorithm 1 achieves an approximation guarantee of $\frac{\alpha}{1+\alpha} - \varepsilon$ when it has access to a value τ obeying $(1 - \varepsilon/2) \cdot f(\text{OPT}) \leq \tau \leq f(\text{OPT})$ and its parameter p is set to $\lceil 4/\varepsilon \rceil$. According to Corollary 16, this implies that by setting the parameter p of Algorithm 2 in the same way and setting ε' to $\varepsilon/2$, we get an algorithm whose approximation ratio is at least $\frac{\alpha}{1+\alpha} - \varepsilon$ and does not assume access to an estimate τ of $f(\text{OPT})$.

It remains to bound the space requirement and update time of the algorithm obtained in this way. Plugging the equalities $p = \lceil 4/\varepsilon \rceil$ and $\varepsilon' = \varepsilon/2$ into the guarantee of Lemma 12, we get that the algorithm we have obtained stores $O(k\varepsilon^{-2} \log \alpha^{-1})$ elements and makes $O(\varepsilon^{-2} \log(k/\alpha))$ marginal value calculations while processing each arriving element. $\square$

5 Multilinear Extension Based Algorithm

The properties of the multlinear extension based variant of our algorithm are summerized by the following theorem.

Theorem 17. *Assume there exists an α-approximation offline algorithm* OfflineAlg *for maximizing a non-negative submodular function subject to cardinality constraint whose space complexity is nearly linear in the size of the ground set. Then, for every constant $\varepsilon \in (0, 1]$, there exists an $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation semi-streaming algorithm for maximizing a non-negative submodular function subject to a cardinality constraint. The algorithm stores at most $O(k\varepsilon^{-2} \log \alpha^{-1})$ elements.*

For ease of the reading, we present below only a simplified version of the multlinear extension based variant of our algorithm. This simplified version (given as Algorithm 3) captures our main new ideas, but makes two simplifying assumptions that can be avoided using standard techniques.

- The first assumption is that Algorithm 3 has access to an estimate τ of $f(\text{OPT})$ obeying $(1-\varepsilon/8) \cdot f(\text{OPT}) \leq \tau \leq f(\text{OPT})$. Such an estimate can be produced using well-known techniques, such as a technique of [33] used in Section 4, at the cost of increasing the space complexity of the algorithm only by a factor of $O(\varepsilon^{-1} \log \alpha^{-1})$.

- The second assumption is that Algorithm 3 has value oracle access to the multilinear extension F. If the time complexity of Algorithm 3 is not important, then this assumption is of no consequence since a value oracle query to F can be emulated using an exponential number of value oracle queries to f. However, the assumption becomes problematic when we would like to keep the time complexity of the algorithm polynomial and we only have value oracle access to f, in which case this assumption can be dropped using standard sampling techniques (such as the one used in [12]). Interestingly, the rounding step of the algorithm and the sampling technique are the only parts of the extension based algorithm that employ randomness. Since the rounding can be made deterministic given either exponential time or value oracle access to F, we get the following observation.

Observation 18. *If* OfflineAlg *is deterministic, then our multinear extension based algorithm is also* deterministic *when it is allowed either exponential computation time or value oracle access to F.*

A more detailed discussion of the techniques for removing the above assumptions can be found in an earlier version of our published paper (available in [3]) that had this variant of our algorithm as one of its main results.

Algorithm 3 has two constant parameters $p \in (0, 1)$ and $c > 0$ and maintains a fractional solution $x \in [0, 1]^V$. This fractional solution starts empty, and the algorithm adds to it fractions of elements as they arrive. Specifically, when an element e arrives, the algorithm considers its marginal contribution with respect to the current fractional solution x. If this marginal contribution exceeds the threshold of $c\tau/k$, then the algorithm tries to add to x a p-fraction of e, but might end up adding a smaller fraction of e if adding a full p-fraction of e to x will make x an infeasible solution, i.e., make $\|x\|_1 > k$ (note that $\|x\|_1$ is the sum of the coordinates of x).

After viewing all of the elements, Algorithm 3 uses the fractional solution x to generate two sets S_1 and S_2 that are feasible (integral) solutions. The set S_1 is generated by rounding the fractional solution x. As mentioned in Section 3, two rounding procedures, named Pipage Rounding and Swap Rounding, were suggested for this task in the literature [12, 18]. Both procedures run in polynomial time and guarantee that the output set S_1 of the rounding is always feasible, and that its expected value with respect to f is at least the value $F(x)$ of the fractional solution x. The set S_2 is generated by applying OfflineAlg to the support of the vector x, which produces a feasible solution that (approximately) maximizes f among all subsets of the support whose size is at most k. After computing the two feasible solutions S_1 and S_2, Algorithm 3 simply returns the better one of them.

Algorithm 3: StreamProcessExtension (simplified) (p, c)

```
1 Let $x \leftarrow \mathbf{1}_\varnothing$.
2 for each arriving element $e$ do
3     if $\partial_e F(x) \geq \frac{c\tau}{k}$ then $x \leftarrow x + \min\{p, k - \|x\|_1\} \cdot \mathbf{1}_e$.
4 Round the vector $x$ to yield a feasible solution $S_1$ such that $\mathbb{E}[f(S_1)] \geq F(x)$.
5 Find another feasible solution $S_2 \subseteq \text{supp}(x)$ by running OfflineAlg with supp($x$)
    as the ground set.
6 return the better solution among $S_1$ and $S_2$.
```

Let us denote by $\hat{x}$ the final value of the fractional solution x (i.e., its value when the stream ends). We begin the analysis of Algorithm 3 with the following useful observation.

In the statement of observation, and in the rest of the section, we denote by $\operatorname{supp}(x)$ the support of vector x, i.e., the set $\{e \in V \mid x_e > 0\}$.

Observation 19. *If $\|\hat{x}\|_1 < k$, then $\hat{x}_e = p$ for every $e \in \operatorname{supp}(\hat{x})$. Otherwise, $\|\hat{x}\|_1 = k$, and the first part of the observation is still true for every element $e \in \operatorname{supp}(\hat{x})$ except for maybe a single element.*

Proof. For every element e added to the support of x by Algorithm 3, the algorithm sets x_e to p unless this will make $\|x\|_1$ exceed k, in which case the algorithm set x_e to be the value that will make $\|x\|_1$ equal to k. Thus, after a single coordinate of x is set to a value other than p (or the initial 0), $\|x\|_1$ becomes k and Algorithm 3 stops changing x. $\square$

Using the last observation we can now bound the space complexity of Algorithm 3, and show (in particular) that it is a semi-streaming algorithm for a constant p when the space complexity of OfflineAlg is nearly linear.

Observation 20. *Algorithm 3 can be implemented so that it stores at most $O(k/p)$ elements.*

Proof. To calculate the sets S_1 and S_2, Algorithm 3 needs access only to the elements of V that appear in the support of x. Thus, the number of elements it needs to store is $O(|\operatorname{supp}(\hat{x})|) = O(k/p)$, where the equality follows from Observation 19. $\square$

We now divert our attention to analyzing the approximation ratio of Algorithm 3. The first step in this analysis is lower bounding the value of $F(\hat{x})$, which we do by considering two cases, one when $\|\hat{x}\|_1 = k$, and the other when $\|\hat{x}\|_1 < k$. The following lemma bounds the value of $F(\hat{x})$ in the first of these cases. Intuitively, this lemma holds since $\operatorname{supp}(\hat{x})$ contains many elements, and each one of these elements must have increased the value of $F(x)$ significantly when added (otherwise, Algorithm 3 would not have added this element to the support of x).

Lemma 21. *If $\|\hat{x}\|_1 = k$, then $F(\hat{x}) \geq c\tau$.*

Proof. Denote by $e_1, e_2, \ldots, e_\ell$ the elements in the support of $\hat{x}$, in the order of their arrival. Using this notation, the value of $F(\hat{x})$ can be written as follows.

$$\begin{aligned} F(\hat{x}) &= F(\mathbf{1}_\varnothing) + \sum_{i=1}^{\ell} \left(F\Big(\hat{x} \wedge \mathbf{1}_{\{e_1,e_2,\ldots,e_i\}}\Big) - F\Big(\hat{x} \wedge \mathbf{1}_{\{e_1,e_2,\ldots,e_{i-1}\}}\Big)\right) \\ &= F(\mathbf{1}_\varnothing) + \sum_{i=1}^{\ell} \left(\hat{x}_{e_i} \cdot \partial_{e_i} F\Big(\hat{x} \wedge \mathbf{1}_{\{e_1,e_2,\ldots,e_{i-1}\}}\Big)\right) \\ &\geq F(\mathbf{1}_\varnothing) + \frac{c\tau}{k} \cdot \sum_{i=1}^{\ell} \hat{x}_{e_i} = F(\mathbf{1}_\varnothing) + \frac{c\tau}{k} \cdot \|\hat{x}\|_1 \geq c\tau \enspace , \end{aligned}$$

where the second equality follows from the multilinearity of F, and the first inequality holds since Algorithm 3 selects an element e_i only when $\partial_{e_i} F\Big(\hat{x} \wedge \mathbf{1}_{\{e_1,e_2,\ldots,e_{i-1}\}}\Big) \geq \frac{c\tau}{k}$. The last inequality holds since f (and thus, also F) is non-negative and $\|\hat{x}\|_1 = k$ by the assumption of the lemma. $\square$

Consider now the case in which $\|\hat{x}\|_1 < k$. Recall that our objective is to lower bound $F(\hat{x})$ in this case as well. To do that, we need to a tool for upper bounding the possible increase in the value of $F(x)$ when some of the indices of x are zeroed. The next lemma provides such an upper bound.

Lemma 22. *Let $f\colon 2^V \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function, let p be a number in the range $[0, 1]$ and let x, y be two vectors in $[0, 1]^V$ such that*

- $\operatorname{supp}(x) \cap \operatorname{supp}(y) = \varnothing$*,*
- *and $y_e \leq p$ for every $e \in V$.*

Then, the multilinear extension F of f obeys $F(x + y) \geq (1 - p) \cdot F(x)$.

Proof. Let us define the function $G_x(S) = \mathbb{E}[f(\mathtt{R}(x) \cup S)]$. It is not difficult to verify that G_x is non-negative and submodular, and that $G_x(\varnothing) = F(x)$. Additionally, since $\operatorname{supp}(x) \cap \operatorname{supp}(y) = \varnothing$, $\mathtt{R}(x + y)$ has the same distribution as $\mathtt{R}(x) \cup \mathtt{R}(y)$, and therefore,

$$\begin{aligned} F(x+y) = \mathbb{E}\Big[f\Big(\mathtt{R}(x+y)\Big)\Big] &= \mathbb{E}\Big[f\Big(\mathtt{R}(x) \cup \mathtt{R}(y)\Big)\Big] \\ &= \mathbb{E}[G_x(\mathtt{R}(y))] \geq (1-p) \cdot G_x(\varnothing) = (1-p) \cdot F(x) \enspace, \end{aligned}$$

where the inequality follows from Lemma 4. □

Using the last lemma, we next prove two lemmata proving upper and lower bounds the expression $F(\hat{x} + \mathbf{1}_{\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})})$. To prove the first of these lemmata, we make use of another standard continuous extension for submodular functions. This extension is known as the *Lovász extension* $\hat{f}\colon [0, 1]^V \to \mathbb{R}$, and it is defined as follows. For every $x \in [0, 1]^V$, $\hat{f}(x) = \mathbb{E}_{\theta \sim [0,1]}[f(\{e \in V : x_e \geq \theta\})]$, where we use the notation $\theta \sim [0, 1]$ to denote a value chosen uniformly at random from the interval $[0, 1]$. An important property of the Lovász extension that we use in the proof of the following lemma, is that it always lower bounds the multilinear extension, i.e., $F(x) \geq \hat{f}(x)$ for every $x \in [0, 1]^V$ [41, Lemma A.4].

Lemma 23. *If $\|\hat{x}\|_1 < k$, then $F\Big(\hat{x} + \mathbf{1}_{\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})}\Big) \geq (1-p) \cdot \Big[p \cdot f(\mathrm{OPT}) + (1-p) \cdot f\Big(\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})\Big)\Big]$.*

Proof. Since $\|\hat{x}\|_1 < k$, Observation 19 guarantees that $\hat{x}_e = p$ for every $e \in \operatorname{supp}(\hat{x})$. Thus $\hat{x} = p \cdot \mathbf{1}_{\mathrm{OPT} \cap \operatorname{supp}(\hat{x})} + p \cdot \mathbf{1}_{\operatorname{supp}(\hat{x}) \setminus \mathrm{OPT}}$, and therefore,

$$\begin{aligned} F\Big(\hat{x} + \mathbf{1}_{\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})}\Big) &= F\Big(p \cdot \mathbf{1}_{\mathrm{OPT} \cap \operatorname{supp}(\hat{x})} + p \cdot \mathbf{1}_{\operatorname{supp}(\hat{x}) \setminus \mathrm{OPT}} + \mathbf{1}_{\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})}\Big) \\ &\geq (1-p) \cdot F\Big(p \cdot \mathbf{1}_{\mathrm{OPT} \cap \operatorname{supp}(\hat{x})} + \mathbf{1}_{\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})}\Big) \\ &\geq (1-p) \cdot \hat{f}\Big(p \cdot \mathbf{1}_{\mathrm{OPT} \cap \operatorname{supp}(\hat{x})} + \mathbf{1}_{\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})}\Big) \\ &= (1-p) \cdot \Big[p \cdot f(\mathrm{OPT}) + (1-p) \cdot f\Big(\mathrm{OPT} \setminus \operatorname{supp}(\hat{x})\Big)\Big] \enspace, \end{aligned}$$

where the first inequality follows from Lemma 22, the second inequality holds since the Lovász extension lower bounds the multilinear extension, and the last equality follows from the definition of the Lovász extension. □

In the following lemma, and the rest of the section, we use the notation $b = k^{-1} \cdot |\text{OPT} \setminus \text{supp}(\hat{x})|$. Intuitively, the lemma holds since the fact that the elements of $\text{OPT} \setminus \text{supp}(\hat{x})$ were not added to the support of x by Algorithm 3 implies that their marginal contribution is small.

Lemma 24. *If $\|\hat{x}\|_1 < k$, then $F\Big(\hat{x} + \mathbf{1}_{\text{OPT}\setminus\text{supp}(\hat{x})}\Big) \leq F(\hat{x}) + bc\tau$.*

Proof. The elements in $\text{OPT} \setminus \text{supp}(\hat{x})$ were rejected by Algorithm 3, which means that their marginal contribution with respect to the fractional solution x at the time of their arrival was smaller than $c\tau/k$. Since the fractional solution x only increases during the execution of the algorithm, the submodularity of f guarantees that the same is true also with respect to $\hat{x}$. More formally, we get

$$\partial_e F(\hat{x}) < \frac{c\tau}{k} \quad \forall\, e \in \text{OPT} \setminus \text{supp}(\hat{x}) \ .$$

Using the submodularity of f again, this implies

$$F\Big(\hat{x} + \mathbf{1}_{\text{OPT}\setminus\text{supp}(\hat{x})}\Big) \leq F(\hat{x}) + \sum_{e\in\text{OPT}\setminus\text{supp}(\hat{x})} \partial_e F(\hat{x}) \leq F(\hat{x}) + |\text{OPT} \setminus \text{supp}(\hat{x})| \cdot \frac{c\tau}{k} = F(\hat{x}) + bc\tau \ . \quad \square$$

Combining the last two lemmata immediately yields the promised lower bound on $F(\hat{x})$. To understand the second inequality in the following corollary, recall that $\tau \leq f(\text{OPT})$.

Corollary 25. *If $\|\hat{x}\|_1 < k$, then $F(\hat{x}) \geq (1-p) \cdot \Big[p \cdot f(\text{OPT}) + (1-p) \cdot f\Big(\text{OPT} \setminus \text{supp}(\hat{x})\Big)\Big] - bc\tau \geq [p(1-p) - bc]\tau + (1-p)^2 \cdot f\Big(\text{OPT} \setminus \text{supp}(\hat{x})\Big)$.*

Our next step is to get a lower bound on the expected value of $f(S_2)$. One easy way to get such a lower bound is to observe that $\text{OPT} \cap \text{supp}(\hat{x})$ is a subset of the support of $\hat{x}$ of size at most k, and thus, is a feasible solution for OfflineAlg to return; which implies $\mathbb{E}[f(S_2)] \geq \alpha \cdot f(\text{OPT} \cap \text{supp}(\hat{x}))$ since the algorithm OfflineAlg used to find S_2 is an α-approximation algorithm. The following lemma proves a more involved lower bound by considering the vector $(b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}$ as a fractional feasible solution (using the rounding methods discussed above it, it can be converted into an integral feasible solution of at least the same value). The proof of the lemma lower bounds the value of the vector $(b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}$ using the concavity of the function $F((t \cdot \hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$ as well as ideas used in the proofs of the previous claims.

Lemma 26. *If $\|\hat{x}\|_1 < k$, then $\mathbb{E}[f(S_2)] \geq \alpha b(1 - p - cb)\tau + \alpha(1 - b) \cdot f(\text{OPT} \cap \text{supp}(\hat{x}))$.*

Proof. Consider the vector $(b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}$. Clearly,

$$\begin{aligned}\Big\|(b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}\Big\|_1 &\leq b \cdot \|\hat{x}\|_1 + \Big\|\mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}\Big\|_1 \\ &\leq |\text{OPT} \setminus \text{supp}(\hat{x})| + |\text{OPT} \cap \text{supp}(\hat{x})| = |\text{OPT}| \leq k \ ,\end{aligned}$$

where the second inequality holds by the definition of b since $\|\hat{x}\|_1 < k$. Thus, due to the existence of the rounding methods discussed in the beginning of the section, there must exist a set S of size at most k obeying $f(S) \geq F((b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$. Since S_2 is produced by OfflineAlg, whose approximation ratio is α, this implies $\mathbb{E}[f(S_2)] \geq \alpha \cdot F((b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$. Thus, to prove the lemma it suffices to show that $F((b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$ is always at least $b(1-p-cb)\tau + (1-b) \cdot f(\text{OPT} \cap \text{supp}(\hat{x}))$.

The first step towards proving the last inequality is getting a lower bound on $F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$. Recall that we already showed in the proof of Lemma 24 that

$$\partial_e F(\hat{x}) < \frac{c\tau}{k} \quad \forall\, e \in \text{OPT} \setminus \text{supp}(\hat{x}) \ .$$

Thus, the submodularity of f implies

$$\begin{aligned} F(\hat{x} \vee \mathbf{1}_{\text{OPT}}) &\leq F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}) + \sum_{e \in \text{OPT}\setminus\text{supp}(\hat{x})} \partial_e F(\hat{x}) \\ &\leq F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}) + \frac{c\tau \cdot |\text{OPT} \setminus \text{supp}(\hat{x})|}{k} = F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}) + cb\tau \ . \end{aligned}$$

Rearranging this inequality yields

$$F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}) \geq F(\hat{x} \vee \mathbf{1}_{\text{OPT}}) - cb\tau \geq (1-p) \cdot f(\text{OPT}) - cb\tau \geq (1-p-cb)\tau \ ,$$

where the second inequality holds by Lemma 22 since Observation 19 guarantees that every coordinate of $\hat{x}$ is either 0 or p. This gives us the promised lower bound on $F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$.

We now note that the submodularity of f implies that $F((t \cdot \hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})})$ is a concave function of t within the range $[0, 1]$. Since b is inside this range,

$$\begin{aligned} F((b\hat{x}) \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}) &\geq b \cdot F(\hat{x} \vee \mathbf{1}_{\text{OPT}\cap\text{supp}(\hat{x})}) + (1-b) \cdot f(\text{OPT} \cap \text{supp}(\hat{x})) \\ &\geq b(1-p-cb)\tau + (1-b) \cdot f(\text{OPT} \cap \text{supp}(\hat{x})) \ , \end{aligned}$$

which completes the proof of the lemma. □

Using the last two claims we can now obtain a lower bound on the value of the solution of Algorithm 3 in the case of $\|\hat{x}\|_1 < k$ which is a function of α, τ and p alone. We note that both the guarantees of Corollary 25 and Lemma 26 are lower bounds on the expected value of the output of the algorithm in this case since $\mathbb{E}[f(S_1)] \geq F(\hat{x})$. Thus, any convex combination of these guarantees is also such a lower bound, and the proof of the following corollary basically proves a lower bound for one such convex combination—for the specific value of c stated in the corollary.

Corollary 27. *If $\|\hat{x}\|_1 < k$ and c is set to $\frac{\alpha(1-p)}{\alpha+1}$, then $\mathbb{E}[\max\{f(S_1), f(S_2)\}] \geq \frac{(1-p)\alpha\tau}{\alpha+1}$.*

Proof. The corollary follows immediately from the non-negativity of f when $p = 1$. Thus, we may assume $p < 1$ in the rest of the proof.

By the definition of S_1, $\mathbb{E}[f(S_1)] \geq F(\hat{x})$. Thus, by Corollary 25 and Lemma 26,

$$\begin{aligned}
\mathbb{E}[\max\{f(S_1), f(S_2)\}] &\geq \max\{\mathbb{E}[f(S_1)], \mathbb{E}[f(S_2)]\} \\
&\geq \max\{[p(1-p) - bc]\tau + (1-p)^2 \cdot f\Big(\mathrm{OPT} \setminus \mathrm{supp}(\hat{x})\Big), \\
&\quad \alpha b(1-p-cb)\tau + \alpha(1-b) \cdot f(\mathrm{OPT} \cap \mathrm{supp}(\hat{x}))\} \\
&\geq \frac{\alpha(1-b)}{\alpha(1-b) + (1-p)^2} \cdot \Big[[p(1-p) - bc]\tau + (1-p)^2 \cdot f\Big(\mathrm{OPT} \setminus \mathrm{supp}(\hat{x})\Big)\Big] \\
&\quad + \frac{(1-p)^2}{\alpha(1-b) + (1-p)^2} \cdot [\alpha b(1-p-cb)\tau + \alpha(1-b) \cdot f(\mathrm{OPT} \cap \mathrm{supp}(\hat{x}))] \enspace .
\end{aligned}$$

To keep the following calculations short, it will be useful to define $q = 1 - p$ and $d = 1 - b$. Using this notation and the fact that the submodularity and non-negativity of f guarantee together $f\Big(\mathrm{OPT} \setminus \mathrm{supp}(\hat{x})\Big) + f\Big(\mathrm{OPT} \cap \mathrm{supp}(\hat{x})\Big) \geq f(\mathrm{OPT}) \geq \tau$, the previous inequality implies

$$\begin{aligned}
\frac{\mathbb{E}[\max\{f(S_1), f(S_2)\}]}{\alpha\tau} &\geq \frac{(1-b)[p(1-p) - bc] + b(1-p)^2(1-p-bc) + (1-b)(1-p)^2}{\alpha(1-b) + (1-p)^2} \\
&= \frac{d[q(1-q) - (1-d)c] + q^2(1-d)[q - (1-d)c] + dq^2}{\alpha d + q^2} \\
&= \frac{d[q - (1-d)c] + q^2(1-d)[q - (1-d)c]}{\alpha d + q^2} = \frac{[d + q^2(1-d)][q - (1-d)c]}{\alpha d + q^2} \\
&= \frac{q[d + q^2(1-d)](d\alpha + 1)}{(\alpha+1)(d\alpha + q^2)} = \frac{d^2\alpha + d\alpha q^2 - d^2\alpha q^2 + d + q^2 - dq^2}{d\alpha + q^2} \cdot \frac{q}{\alpha + 1} \enspace , \qquad (1)
\end{aligned}$$

where the fourth equality holds by plugging in the value we assume for c.

The second fraction in the last expression is independent of the value of d, and the derivative of the first fraction in this expression as a function of d is

$$\begin{aligned}
&\frac{(2d\alpha + \alpha q^2 - 2d\alpha q^2 + 1 - q^2)[d\alpha + q^2] - \alpha(d^2\alpha + d\alpha q^2 - d^2\alpha q^2 + d + q^2 - dq^2)}{[d\alpha + q^2]^2} \\
&\qquad = \frac{1 - q^2}{[d\alpha + q^2]^2} \cdot [q^2(1-\alpha) + d\alpha(d\alpha + 2q^2)] \enspace ,
\end{aligned}$$

which is always non-negative since both q and α are numbers between 0 and 1. Thus, we get that the minimal value of the expression (1) is obtained for $d = 0$ for any choice of q and α. Plugging this value into d yields

$$\mathbb{E}[\max\{f(S_1), f(S_2)\}] \geq \frac{q\alpha\tau}{\alpha + 1} = \frac{(1-p)\alpha\tau}{\alpha + 1} \enspace . \qquad \square$$

Note that Lemma 21 and Corollary 27 both prove the same lower bound on the expectation $\mathbb{E}[\max\{f(S_1), f(S_2)\}]$ when c is set to the value it is set to in Corollary 27 (because $\mathbb{E}[\max\{f(S_1), f(S_2)\}] \geq \mathbb{E}[f(S_1)] \geq F(\hat{x})$). Thus, we can summarize the results we have proved so far using the following proposition.

Proposition 28. *Algorithm 3 is a semi-streaming algorithm storing $O(k/p)$ elements. Moreover, for the value of the parameter c given in Corollary 27, the output set produced by this algorithm has an expected value of at least $\frac{\alpha\tau(1-p)}{\alpha+1}$.*

Using the last proposition, we can now prove the following theorem. As discussed at the beginning of the section, the assumption that τ is known can be dropped at the cost of a slight increase in the number of of elements stored by the algorithm, which yields Theorem 17.

Theorem 29. *For every constant $\varepsilon \in (0, 1]$, there exists a semi-streaming algorithm that assumes access to an estimate τ of $f(\mathrm{OPT})$ obeying $(1 - \varepsilon/8) \cdot f(\mathrm{OPT}) \leq \tau \leq f(\mathrm{OPT})$ and provides $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation for the problem of maximizing a non-negative submodular function subject to cardinality constraint. This algorithm stores at most $O(k\varepsilon^{-1})$ elements.*

Proof. Consider the algorithm obtained from Algorithm 3 by setting $p = \varepsilon/2$ and c as is set in Corollary 27. By Proposition 28, this algorithm stores only $O(k/p) = O(k\varepsilon^{-1})$ elements, and the expected value of its output set is at least

$$\frac{\alpha\tau(1-p)}{\alpha+1} \geq \frac{\alpha(1-\varepsilon/8)(1-\varepsilon/2)}{\alpha+1} \cdot f(\mathrm{OPT}) \geq \frac{\alpha(1-\varepsilon)}{\alpha+1} \cdot f(\mathrm{OPT}) \geq \left(\frac{\alpha}{\alpha+1} - \varepsilon\right) \cdot f(\mathrm{OPT}) \ ,$$

where the first inequality holds since τ obeys, by assumption, $\tau \geq (1 - \varepsilon/8) \cdot f(\mathrm{OPT})$. □

6 Algorithm for DR-SM Functions

In section 5 we saw that submodular function can be optimized via their continuous multilinear extensions. In this case, the extension to the continuous domain was a mean towards optimizing a discrete function. However, in other settings it is interested to study submodular-like continuous functions for their own sake. In this section we consider a family of such functions known as DR-submodular functions, which includes the multilinear extensions of submodular functions as a special case. Let a, b be two vectors in $[0,1]^V$ such that $a \leq b$.[6] Formally, a function $F\colon [0,1]^V \to \mathbb{R}_{\geq 0}$ is DR-submodular if for every $e \in V$ and $0 < k < 1 - b_e$,

$$F(a + k \cdot \mathbf{1}_e) - F(a) \geq F(b + k \cdot \mathbf{1}_e) - F(b) \ .$$

Given an integer k, our goal is to construct a semi-streaming algorithm for maximizing a non-negative DR-submodular function $F\colon [0,1]^V \to \mathbb{R}_{\geq 0}$ subject to a support cardinality constraint, meaning that a solution $x \in [0,1]^V$ is feasible only if $|\operatorname{supp}(x)| \leq k$. We propose for this problem the algorithm given as Algorithm 4. This algorithm makes two simplifying assumptions. The first assumption is that it has access to an estimate τ of $F(x^*)$, where x^* is an optimal solution for our problem, such that $(1 - \varepsilon/2) \cdot F(x^*) \leq \tau \leq F(x^*)$. Recall that in Section 4, we explained how one can drop the assumption that τ is known at the cost of an increase in the space complexity of Algorithm 1. One can easily verify that the same technique can also be used to omit this assumption from Algorithm 4. The second simplifying assumption made by Algorithm 4 is that it can find the exact maximum of F when the domain is restricted to a single dimension. We explain how to drop this assumption in Section 7 at the cost of making slightly more value oracle queries.

[6]We say that $a \leq b$, if $a_e \leq b_e$ for every element $e \in V$.

Algorithm 4: Deterministic (DR-SM) (p, α)

```
1 Let $c \leftarrow \frac{\alpha}{1+\alpha}$.
2 for $i = 1$ to $p$ do Let $x_i \leftarrow \mathbf{1}_\varnothing$.
3 for each arriving element $e$ do
4     for $i = 1$ to $p$ do
5         Let $q_i \leftarrow \arg\max_{q \in [0,1]} F(x_i + q \cdot \mathbf{1}_e)$.
6     if there exists an integer $1 \leq i \leq p$ such that $|\operatorname{supp}(x_i)| < k$ and
      $F(x_i + q_i \cdot \mathbf{1}_e) \geq F(x_i) + \frac{c\tau}{k}$ then
7         Update $x_i \leftarrow x_i + q_i \cdot \mathbf{1}_e$ (if there are multiple options for $i$, pick an arbitrary
          one).
8 Find another feasible solution $x_o \leq \bigvee_{i=1}^{p} \mathbf{1}_{\operatorname{supp}(x_i)}$ by running OfflineAlg with
  $\bigcup_{i=1}^{p} \operatorname{supp}(x_i)$ as the ground set.
9 return the solution maximizing $F$ among $x_o$ and the $p$ solutions $x_1, x_2, \ldots, x_p$.
```

Since Algorithm 4 stores elements only in the $p+1$ solutions it maintains, and all these solutions are feasible (and thus, their support contains at most k elements), we immediately get the following observation. Note that this observation implies (in particular) that Algorithm 4 is a semi-streaming algorithm for a constant p when the space complexity of OfflineAlg is nearly linear.

Observation 30. *Algorithm 4 stores at most $O(pk)$ elements.*

We now divert our attention to analyzing the approximation ratio of Algorithm 4. Let us denote by $\hat{x}_i$ the final state of x_i (i.e., the content of this vector when the stream ends), and consider two cases. The first (easy) case is when the support of at least one of the solutions $x_1, x_2, \ldots, x_p$ reaches a size of k. The next lemma analyzes the approximation guarantee of Algorithm 4 in this case.

Lemma 31. *If there is an integer $1 \leq i \leq p$ such that $|\operatorname{supp}(\hat{x}_i)| = k$, then the output of Algorithm 4 has value of at least $\frac{\alpha\tau}{1+\alpha}$.*

Proof. Denote by $e_1, e_2, \ldots, e_k$ the elements of $\operatorname{supp}(\hat{x}_i)$ in the order of their arrival, and let us define $\hat{x}_i^j = \hat{x}_i \wedge \mathbf{1}_{\{e_1, e_2, \ldots, e_j\}}$, and $q_i^j = \left(\hat{x}_i\right)_j$. Using these notations, the value of $F(\hat{x}_i)$ can be written as follows.

$$\begin{aligned} F(\hat{x}_i) = F(\mathbf{1}_\varnothing) + \sum_{j=1}^{k} \left(F(\hat{x}_i^j) - F(\hat{x}_i^{j-1})\right) &= F(\mathbf{1}_\varnothing) + \sum_{j=1}^{k} \left(F(\hat{x}_i^{j-1} + q_i^j \cdot \mathbf{1}_{e_j}) - F(\hat{x}_i^{j-1})\right) \\ &\geq \sum_{j=1}^{k} \frac{\alpha\tau}{(1+\alpha)k} = \frac{\alpha\tau}{1+\alpha} \enspace, \end{aligned}$$

where the inequality holds since the non-negativity of F implies $F(\mathbf{1}_\varnothing) \geq 0$ and Algorithm 4 adds a fraction $q_i^j \cdot \mathbf{1}_{e_j}$ to x_i only when $F(\hat{x}_i^{j-1} + q_i^j \cdot \mathbf{1}_{e_j}) - F(\hat{x}_i^{j-1}) \geq \frac{c\tau}{k} = \frac{\alpha\tau}{(1+\alpha)k}$. The lemma now follows since the solution outputted by Algorithm 4 is at least as good as $\hat{x}_i$. □

Consider now the case in which no vector x_i reaches a support of size k. In this case our objective is to show that at least one of the solutions computed by Algorithm 4 has a large value. Lemmata 35 and 36 lower bound the value of the average solution among $x_1, x_2, \dots, x_p$ and the solution x_o, respectively. The proof of Lemma 35 uses Corollary 33, which is derived from the following known lemma.

Lemma 32 (Lemma 2.2 from [10])**.** *Let $f\colon 2^V \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Denote by $A(p)$ a random subset of A where each element appears with probability at most p (not necessarily independently). Then, $\mathbb{E}[f(A(p))] \geq (1-p) \cdot f(\varnothing)$.*

Corollary 33. *Let $F\colon [0,1]^V \to \mathbb{R}_{\geq 0}$ be a non-negative DR-submodular function. Denote by $x(p)$ a random vector generated from x where $\big(x(p)\big)_i$ is equal to the i-th element of x with probability at most p (not necessarily independently), and is equal to 0 otherwise. Then, $\mathbb{E}[F(x(p))] \geq (1-p) \cdot F(\mathbf{1}_\varnothing)$.*

Proof. Let $A = \operatorname{supp}(x)$, and define the function $f(B) = F(x \wedge \mathbf{1}_B)$ for each set $B \subseteq V$. Clearly, f is submodular, and the random set $A(p) = \operatorname{supp}(x(p))$ is a subset of A obeying the conditions stated in Lemma 32. Therefore, Lemma 32 yields

$$\mathbb{E}[F(x(p))] = \mathbb{E}[f(A(p))] \geq (1-p) f(\varnothing) = (1-p) F(\mathbf{1}_\varnothing) \ . \qquad \square$$

Before we proceed with the analysis of the case of $|\operatorname{supp}(\hat{x}_i)| < k$ for every $1 \leq i \leq p$, we need to introduce some additional notation. For two vectors $x, y \in [0,1]^V$, we let $x * y$ be the vector such that $(x*y)_e = 1 - (1-x_e)(1-y_e) = x_e + y_e - x_e y_e$ for every element $e \in V$. Additionally, we define the continuous function $G_x(y) = F(x * y)$. Intuitively, G_x is obtained from F using the following two steps: (i) restrict F to the box lower bounded by x and upper bounded by $\mathbf{1}_V$, and (ii) apply a linear transformation to map this box back to $[0,1]^V$. Given this intuition, it is not difficult to see that G_x should be DR-submodular. The following observation shows this formally.

Observation 34. *For every vector $x \in [0,1]^V$, G_x is DR-submodular.*

Proof. Fix two vectors $y, z \in [0,1]^V$ such that $y \leq z$, and consider an element $e \in V$ and value $\delta \geq 0$ such that $z_e + \delta \leq 1$. To prove the observation, we need to show that

$$G_x(y + \delta \cdot \mathbf{1}_e) - G_x(y) \geq G_x(z + \delta \cdot \mathbf{1}_e) - G_x(z) \ .$$

To see that this is indeed the case, we note that $x * w$ is a monotone function of w, and therefore, $x * y \leq x * z$. By the DR-submodularity of F, this implies

$$\begin{aligned} G_x(y + \delta \cdot \mathbf{1}_e) - G_x(y) &= F(x * (y + \delta \cdot \mathbf{1}_e)) - F(x*y) \\ &= F(x * y + \delta(1-x_e) \cdot \mathbf{1}_e) - F(x*y) \\ &\geq F(x * z + \delta(1-x_e) \cdot \mathbf{1}_e) - F(x*z) \\ &= F(x * (z + \delta \cdot \mathbf{1}_e)) - F(x*z) \\ &= G_x(z + \delta \cdot \mathbf{1}_e) - G_x(z) \ . \end{aligned} \qquad \square$$

We also denote by o the vector whose e-th coordinate equals $(x^*)_e$ if no fraction of e was added during the execution of Algorithm 4 to any of the vectors $x_1, x_2, \ldots, x_p$, and 0 otherwise. More formally, $o = \mathrm{OPT} \wedge \mathbf{1}_{V \setminus \bigcup_{i=1}^p \mathrm{supp}(\hat{x}_i)}$. It is also useful to define the shorthand $b = |\mathrm{supp}(o)|/k$. Using this notation, we can now prove the promised lower bounds on the values of the solutions that Algorithm 4 constructs when none of the vectors $x_1, x_2, \ldots, x_p$ reach a support of size k.

Lemma 35. *If $|\mathrm{supp}(\hat{x}_i)| < k$ for every integer $1 \leq i \leq p$, then, for every fixed vector $a \in [0,1]^V$, $p^{-1} \cdot \sum_{i=1}^p G_a(\hat{x}_i) \geq (1 - p^{-1}) \cdot G_a(o) - \alpha b \tau/(1+\alpha)$.*

Proof. The elements in $\mathrm{supp}(o)$ were rejected by Algorithm 4. Since no solution x_i reaches a support of size k, this means that the marginal contribution of every fraction of an element in $\mathrm{supp}(o)$ with respect to every solution x_i at the time of their arrival was smaller than $c\tau/k$. Moreover, since Algorithm 4 only increases its solutions during its execution, the DR-submodularity of F guarantees that for every $e \in \mathrm{supp}(o)$ the contribution $\max_{q \in [0,1]} \Big(G_a(\hat{x}_i + q \cdot \mathbf{1}_e) - G_a(\hat{x}_i)\Big) = \max_{q \in [0,1]} \Big(F((\hat{x}_i * a) + q \cdot \mathbf{1}_e) - F(\hat{x}_i * a)\Big)$ is also below this threshold for every $1 \leq i \leq p$ since $\hat{x}_i * a \geq \hat{x}_i \geq x_i$. More formally, we get

$$\max_{q \in [0,1]} \Big(G_a(\hat{x}_i + q \cdot \mathbf{1}_e) - G_a(\hat{x}_i)\Big) < \frac{c\tau}{k} = \frac{\alpha\tau}{k(1+\alpha)} \quad \forall\, e \in \mathrm{supp}(o) \text{ and integer } 1 \leq i \leq p \enspace .$$

Using the DR-submodularity of G_a, this implies that for every integer $1 \leq i \leq p$

$$\begin{aligned} G_a(o \vee \hat{x}_i) &\leq G_a(\hat{x}_i) + \sum_{e \in \mathrm{supp}(o)} \max_{q \in [0,1]} \Big(G_a(\hat{x}_i + q \cdot \mathbf{1}_e) - G_a(\hat{x}_i)\Big) \\ &\leq G_a(\hat{x}_i) + |\mathrm{supp}(o)| \cdot \frac{\alpha\tau}{k(1+\alpha)} = G_a(\hat{x}_i) + \frac{\alpha b \tau}{1+\alpha} \enspace . \end{aligned}$$

Adding up the above inequalities for all i (and dividing by p), we get

$$p^{-1} \cdot \sum_{i=1}^p G_a(\hat{x}_i) \geq p^{-1} \cdot \sum_{i=1}^p G_a(o \vee \hat{x}_i) - \frac{\alpha b \tau}{1+\alpha} \enspace .$$

We now note that $p^{-1} \cdot \sum_{i=1}^p G_a(o \vee \hat{x}_i)$ can be viewed as the expected value of the non-negative DR-submodular function $H(x) = G_a(o \vee x)$ over a random vector x that is equal to every one of the solutions $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_p$ with probability $1/p$. Since $\mathrm{supp}(\hat{x}_i) \cap \mathrm{supp}(\hat{x}_j) = \varnothing$ for every $1 \leq i \neq j \leq p$, the support of x contains every element with probability at most $1/p$, and thus, by Corollary 33,

$$p^{-1} \cdot \sum_{i=1}^p G_a(o \vee \hat{x}_i) = \mathbb{E}[H(x)] \geq (1 - p^{-1}) \cdot H(\mathbf{1}_\varnothing) = (1 - p^{-1}) \cdot G_a(o) \enspace .$$

The lemma now follows by combining the last two inequalities. □

As mentioned above, our next step is to get a lower bound on the value of $F(x_o)$. One easy way to get such a lower bound is to observe that $\mathrm{supp}(\mathrm{OPT} - o)$ is a subset of $\bigcup_{i=1}^p \mathrm{supp}(\hat{x}_i)$ of size at most k, and thus, $x^* - o$ is a candidate to be x_o; which implies $\mathbb{E}[F(x_o)] \geq \alpha \cdot f(\mathrm{OPT} - o)$

since the algorithm OfflineAlg used to find x_o is an α-approximation algorithm. The following lemma proves a more involved lower bound by considering vectors whose values are at least $G_{(\text{OPT}-o)}(b \cdot \hat{x}_i)$, and whose support is of size at most k, as candidates to be x_o. The proof of the lemma obtains such vectors using rounding methods such as Pipage Rounding and Swap Rounding [12, 18].

Lemma 36. *If $|\operatorname{supp}(\hat{x}_i)| < k$ for every integer $1 \leq i \leq p$, then $\mathbb{E}[F(x_o)] \geq \alpha b\tau(1 - p^{-1} - \alpha b/(1+\alpha)) + \alpha(1-b) \cdot F(\text{OPT} - o)$.*

Proof. Fix some integer $1 \leq i \leq p$, and consider the set function $g(S) = G_{(\text{OPT}-o)}(\hat{x}_i \wedge \mathbf{1}_S)$. Clearly, g is submodular. Furthermore, since $|\operatorname{supp}(\hat{x}_i)| < k$, the definition of b implies that $\|(b \cdot \hat{x}_i)\|_1 \leq b \cdot |\operatorname{supp}(\hat{x}_i)| \leq |\operatorname{supp}(o)|$. Thus, using either Pipage Rounding [12] or Swap Rounding [18] one can obtain a set $S \subseteq |\operatorname{supp}(\hat{x}_i)|$ of size at most $|\operatorname{supp}(o)|$ such that $g(S) \geq G_{(\text{OPT}-o)}(b \cdot \hat{x}_i)$. Notice now that the support of $x * y$ is exactly the union of $\operatorname{supp}(x)$ and $\operatorname{supp}(y)$, and therefore,

$$|\operatorname{supp}((\hat{x}_i \wedge \mathbf{1}_S) * (\text{OPT}-o))| \leq |S| + |\operatorname{supp}(\text{OPT})| - |\operatorname{supp}(o)| \leq |\operatorname{supp}(o)| + k - |\operatorname{supp}(o)| = k \ .$$

Thus, each vector $(\hat{x}_i \wedge \mathbf{1}_S) * (\text{OPT} - o)$ is a candidate to be x_o, whose value is

$$F((\hat{x}_i \wedge \mathbf{1}_S) * (\text{OPT} - o)) = G_{(\text{OPT}-o)}(\hat{x}_i \wedge \mathbf{1}_S) = g(S) \geq G_{(\text{OPT}-o)}(b \cdot \hat{x}_i) \ .$$

Since the vector x_o is produced by OfflineAlg, whose approximation ratio is α, this implies $\mathbb{E}[F(x_o)] \geq \alpha \cdot G_{(\text{OPT}-o)}(b \cdot \hat{x}_i)$. Furthermore, by averaging this equation over the possible values of i, we get

$$\begin{aligned}\mathbb{E}[F(x_o)] &\geq \alpha p^{-1} \cdot \sum_{i=1}^{p} G_{(\text{OPT}-o)}(b \cdot \hat{x}_i) \\ &\geq \alpha p^{-1} \cdot \left[(1-b) \cdot \sum_{i=1}^{p} G_{(\text{OPT}-o)}(\mathbf{1}_\varnothing) + b \cdot \sum_{i=1}^{p} G_{(\text{OPT}-o)}(\hat{x}_i)\right] \ ,\end{aligned}$$

where the second inequality holds since the DR-submodularity of $G_{(\text{OPT}-o)}$ implies that $G_{(\text{OPT}-o)}(t \cdot \hat{x}_i)$ is a concave function of t within the range $[0, 1]$ (and b falls is inside this range). Since $G_{(\text{OPT}-o)}(\mathbf{1}_\varnothing) = F(\text{OPT} - o)$, using Lemma 35 with $a = \text{OPT} - o$ yields

$$\begin{aligned}\mathbb{E}[F(x_o)] &\geq \alpha(1-b) \cdot F(\text{OPT} - o) + \alpha b \left[(1 - p^{-1}) \cdot G_{(\text{OPT}-o)}(o) - \alpha b\tau/(1+\alpha)\right] \\ &\geq \alpha(1-b) \cdot F(\text{OPT} - o) + \alpha b\tau[(1 - p^{-1}) - \alpha b/(1+\alpha)] \ ,\end{aligned}$$

where the last inequality holds since $G_{(\text{OPT}-o)}(o) = F(\text{OPT}) \geq \tau$. $\square$

Combining the guarantees of the last two lemmata we can now obtain a lower bound which is independent of b on the value of the solution of Algorithm 4 in the case in which $|\operatorname{supp}(\hat{x}_i)| < k$ for every integer $1 \leq i \leq p$.

Corollary 37. *If $|\operatorname{supp}(\hat{x}_i)| < k$ for every integer $1 \leq i \leq p$, then the output of Algorithm 4 has value of at least $\left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau$.*

Proof. The corollary follows immediately from the non-negativity of F when $p = 1$. Thus, we may assume $p \geq 2$ in the rest of the proof.

One can easily verify that $G_{\mathbf{1}_\varnothing}(x) = F(x * \mathbf{1}_\varnothing) = F(x)$. Since Algorithm 4 outputs the best solution among the $p + 1$ solutions it creates, we get by Lemma 35 (for $a = \mathbf{1}_\varnothing$) and Lemma 36 that the value of the solution it outputs is at least

$$\begin{aligned}
\max\left\{p^{-1}\sum_{i=1}^{p} F(\hat{x}_i), F(x_o)\right\} \geq{}& \max\{(1-p^{-1})\cdot F(o) - \alpha b\tau/(1+\alpha), \\
&\alpha b\tau(1-p^{-1}-\alpha b/(1+\alpha)) + \alpha(1-b)\cdot F(\mathrm{OPT}-o)\} \\
\geq{}& \frac{\alpha(1-b)}{\alpha(1-b)+1-p^{-1}}\cdot\left[(1-p^{-1})\cdot F(o) - \alpha b\tau/(1+\alpha)\right] \\
&+ \frac{1-p^{-1}}{\alpha(1-b)+1-p^{-1}}\cdot\left[\alpha b\tau(1-p^{-1}-\alpha b/(1+\alpha)) + \alpha(1-b)\cdot F(\mathrm{OPT}-o)\right] .
\end{aligned}$$

Note now that the DR-submodularity and non-negativity of F guarantee together $F(o) + F(\mathrm{OPT} - o) \geq F(\mathrm{OPT}) \geq \tau$. Using this fact and the non-negativity of F, the previous inequality yields

$$\begin{aligned}
\frac{\max\{p^{-1}\sum_{i=1}^{p}F(\hat{x}_i), F(x_o)\}}{\alpha\tau} \geq{}& \\
&\max\left\{0, \frac{(1-b)(1-p^{-1}) - \alpha b(1-b)/(1+\alpha) + b(1-p^{-1})(1-p^{-1}-\alpha b/(1+\alpha))}{\alpha(1-b)+1-p^{-1}}\right\} \\
\geq{}& \max\left\{0, \frac{(1-b)(1-p^{-1}) - \alpha b(1-b)/(1+\alpha) + b(1-p^{-1})(1-p^{-1}-\alpha b/(1+\alpha))}{\alpha(1-b)+1}\right\} \\
\geq{}& \frac{(1-b) - \alpha b(1-b)/(1+\alpha) + b(1-\alpha b/(1+\alpha))}{\alpha(1-b)+1} - 2p^{-1} ,
\end{aligned}$$

where the last two inequalities hold since $\alpha \in (0, 1]$, $b \in [0, 1]$ and $p \geq 2$. Simplifying the last inequality, we get

$$\begin{aligned}
\frac{\max\{p^{-1}\sum_{i=1}^{p}F(\hat{x}_i), F(x_o)\}}{\alpha\tau} &\geq \frac{(1-b)(1+\alpha) - \alpha b(1-b) + b(1+\alpha-\alpha b)}{(1+\alpha)[\alpha(1-b)+1]} - 2p^{-1} \\
&= \frac{1+\alpha(1-b)}{(1+\alpha)[\alpha(1-b)+1]} - 2p^{-1} = \frac{1}{1+\alpha} - 2p^{-1} .
\end{aligned}$$

The corollary now follows by rearranging this inequality (and recalling that $\alpha \in (0, 1]$). □

Note that the guarantee of Corollary 37 (for the case it considers) is always weaker than the guarantee of Lemma 31. Thus, we can summarize the results we have proved so far using the following proposition.

Proposition 38. *Algorithm 4 stores $O(pk)$ elements, makes at most $O(p)$ value oracle queries while processing each arriving element and its output has an expected value of at least $\left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau$.*

Using the last proposition, we can now prove the following theorem.

Theorem 39. *Assume there exists an α-approximation offline algorithm* OfflineAlg *for maximizing a non-negative DR-submodular function F subject to a support cardinality constraint whose space complexity is nearly linear in the size of the ground set. Then, for every constant $\varepsilon \in (0, 1]$, there exists a semi-streaming algorithm that assumes access to an estimate τ of $F(x^*)$ obeying $(1-\varepsilon/2)\cdot F(x^*) \leq \tau \leq F(x^*)$ and provides $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation for the same problem. This algorithm stores $O(k\varepsilon^{-1})$ elements and uses $O(\varepsilon^{-1})$ value oracle queries while processing each arriving element.*

Proof. Consider the algorithm obtained from Algorithm 4 by setting $p = \lceil 4/\varepsilon \rceil$. By Proposition 38 and the non-negativity of F, this algorithm stores only $O(pk) = O(k\varepsilon^{-1})$ elements, uses only $p = O(\varepsilon^{-1})$ value oracle queries while processing each arriving element, and the expected value of its output vector is at least

$$\max\left\{0, \left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau\right\} \geq \left(\frac{\alpha}{\alpha+1} - \frac{\varepsilon}{2}\right) \cdot (1-\varepsilon/2) \cdot F(x^*) \geq \left(\frac{\alpha}{\alpha+1} - \varepsilon\right) \cdot F(x^*) \enspace ,$$

where the first inequality holds since $p \geq 4/\varepsilon$ and τ obeys, by assumption, $\tau \geq (1-\varepsilon/2) \cdot F(x^*)$. $\square$

7 Avoiding One Dimensional Optimization Problems

Line 5 of Algorithm 4 requires us to calculate $\arg\max_{q\in[0,1]} F(x_i + q\cdot \mathbf{1}_e)$ for an arriving element e and a maintained solution x_i. If we are able to exactly solve 1-dimensional optimization problems involving F, then this line can be implemented as is. Otherwise, we need to approximate it. Section 7.1 explains how to get such an approximation efficiently. Then, Section 7.2 analyzes the performance of Algorithm 4 when Line 5 is implemented in this approximate manner.

7.1 Approximate Implementation of Line 5

When viewed as a function of q, $F(x_i + q \cdot \mathbf{1}_e)$ is a concave non-negative function. Thus, the problem of approximating $\arg\max_{q\in[0,1]} F(x_i + q \cdot \mathbf{1}_e)$ is a special case of the following more general problem. Given a 1-dimensional concave function $H\colon [0, 1] \to \mathbb{R}_{\geq 0}$, find a value $\tilde{q} \in [0, 1]$ that approximately maximizes $H(\tilde{q})$. More formally, given a value $\delta \in [0, 1/4]$, we want an algorithm that produces $\tilde{q} \in [0, 1]$ such that $H(\tilde{q}) \geq (1 - 4\delta)H(q^*)$, where $q^* = \max_{q\in[0,1]} H(q)$. For the sake of simplicity, we assume below that δ^{-1} is an integer. If this is not the case, then δ can be replaced with some value from the range $[\delta/2, \delta]$ that has this property.

The algorithm we suggest for the above problem appears below as Algorithm 5. This algorithm discretizes the domain of H by considering only points that are integer multiples of δ. Formally, the discretized version of H is denoted by the function $H_\delta\colon \{0, 1, \ldots, 1/\delta\} \to \mathbb{R}_{\geq 0}$ defined as $H_\delta(t) = H(t\delta)$. Algorithm 5 then uses binary search to find a pair of values t_1 and t_2 such that $t_1 + 1 = t_2$ and $q^* \in [\delta t_1, \delta(t_2 + 1)]$. For the purpose of the binary search, it is useful to define $\Delta H_\delta(t) \triangleq H_\delta(t + 1) - H_\delta(t)$. Intuitively, $\Delta H_\delta(t)$ is a discrete derivative of H_δ at t.

Algorithm 5: 1-Dimensional Optimization (δ)

```
1 Let $t_1 \leftarrow 0$ and let $t_2 \leftarrow 1/\delta - 1$.
2 while $t_2 - t_1 > 1$ do
3     Let $t' \leftarrow t_1 + \lceil (t_2 - t_1)/2 \rceil$.
4     if $\Delta H_\delta(t') > 0$ then
5         Update $t_1 \leftarrow t'$.
6     else
7         Update $t_2 \leftarrow t'$.
8 return $\tilde{q} \leftarrow \arg\max_{t \in t_1, t_2+1} H(t\delta)$.
```

The concavity of the H implies the following observation, which shows that we indeed have $q^* \in [\delta t_1, \delta(t_2 + 1)]$ when Algorithm 5 terminates.

Observation 40. *For every $t \in [0, 1/\delta - 1]$, if $\Delta H_\delta(t) = H(t\delta + \delta) - H(t\delta) > 0$, then $q^* \geq t\delta$. Otherwise, $q^* \leq t\delta + \delta$.*

Using the last observation we can now analyze the approximation ratio of Algorithm 5.

Proposition 41. *Given a non-negative 1-dimensional concave function $H\colon [0,1] \to \mathbb{R}_{\geq 0}$ with a maximum value at q^* and an error parameter $0 \leq \delta \leq 1/4$, Algorithm 5 outputs an estimate $\tilde{q}$ such that $H(\tilde{q}) \geq (1 - 4\delta)H(q^*)$.*

Proof. Assume first that $q^* \leq 1/2$, and let $a = [1-(t_2+1)\delta]/(1-q^*)$. Since $1 \geq (t_2+1)\delta \geq q^*$, $a \in [0,1]$ and $(t_2 + 1)\delta = aq^* + (1-a) \cdot 1$, and thus, the concavity of H implies

$$\begin{aligned} H((t_2+1)\delta) &\geq a \cdot H(q^*) + (1-a) \cdot H(1) \geq a \cdot H(q^*) \\ &= \frac{1-(t_2+1)\delta}{1-q^*} \cdot H(q^*) \geq \frac{1-q^*-2\delta}{1-q^*} \cdot H(q^*) \geq (1-4\delta) \cdot H(q^*) \enspace , \end{aligned}$$

where the second inequality follows from the non-negativity of H. Similarly, in the case of $q^* \geq 1/2$, we observe that $t_1\delta = aq^* + (1-a) \cdot 0$ for $a = t_1\delta/q^* \in [0,1]$. Thus, the concavity of H yields in this case

$$H(t_1\delta) \geq a \cdot H(q^*) + (1-a) \cdot H(0) \geq \frac{t_1\delta}{q^*} \cdot H(q^*) \geq \frac{q^*-2\delta}{q^*} \cdot H(q^*) \geq (1-4\delta) \cdot H(q^*) \enspace ,$$

where the second inequality holds again by the non-negativity of H. □

To complete the analysis of Algorithm 5, we bound its query complexity in the next observation.

Observation 42. *Algorithm 5 makes at most $O(\log \delta^{-1})$ value oracle queries.*

Proof. The binary search that Algorithm 5 performs starts in a range of size $1/\delta - 1$ and ends when $t_2 - t_1 = 1$, and thus, it runs for $O(\log \delta^{-1})$ iterations. In each of these iterations, the evaluation of ΔH_δ involves two queries to H. The observation follows. □

7.2 Algorithm without One Dimensional Optimization Problems

We proceed by analyzing the performance of a variant of Algorithm 4 that uses Algorithm 5 to maximize one dimensional concave functions, and is given below as Algorithm 6. In the analysis of this algorithm we assume $p \geq 5$.

Algorithm 6: DETERMINISTIC (DR-SM) (p, α)

```
1 Let $c \leftarrow \frac{\alpha}{1+\alpha}$.
2 for $i = 1$ to $p$ do Let $x_i \leftarrow \mathbf{1}_\varnothing$.
3 for each arriving element $e$ do
4     for $i = 1$ to $p$ do
5         Use Algorithm 5, with accuracy parameter $p^{-1}$, to get an approximation $\tilde{q}_i$ of
          $\arg\max_{q \in [0,1]} \Big(F(x_i + q \cdot \mathbf{1}_e) - F(x_i)\Big)$.
6     if there exists an integer $1 \leq i \leq p$ such that $|\operatorname{supp}(x_i)| < k$ and
      $F(x_i + \tilde{q}_i \cdot \mathbf{1}_e) \geq F(x_i) + \frac{(1-4p^{-1})c\tau}{k}$ then
7         Update $x_i \leftarrow x_i + \tilde{q}_i \cdot \mathbf{1}_e$ (if there are multiple options for $i$, pick an arbitrary
          one).
8 Find another feasible solution $x_o \leq \bigvee_{i=1}^{p} \mathbf{1}_{\operatorname{supp}(x_i)}$ by running OFFLINEALG with
  $\bigcup_{i=1}^{p} \operatorname{supp}(x_i)$ as the ground set.
9 return the solution maximizing $F$ among $x_o$ and the $p$ solutions $x_1, x_2, \ldots, x_p$.
```

Algorithm 6 differ from Algorithm 4 only by the implementation of Line 5, and the threshold in Line 6. Notice that for every arriving element Algorithm 6 uses Algorithm 5 p times with $\delta = p^{-1}$. Together with Observations 30 and 42 this implies the following observation.

Observation 43. *Algorithm 6 stores at most $O(pk)$ elements, and makes at most $O(p \log p)$ value oracle queries while processing each arriving element.*

The next step is to bound the approximation ratio of the algorithm. Similarly to what we have done in Section 6, we consider two cases, where in the first case there exists $1 \leq i \leq p$ such that the support of x_i reaches a size of k, and in the second case there is no such i. The following lemma bounds the approximation ratio in the first case, and it corresponds to Lemma 31.

Lemma 44. *If there is an integer $1 \leq i \leq p$ such that $|\operatorname{supp}(\hat{x}_i)| = k$, then the output of Algorithm 6 has value of at least $\left(\frac{\alpha}{1+\alpha} - 2p^{-1}\right)\tau$.*

Proof. The algorithm outputs a solution that is at least as good as $\hat{x}_i$, and thus, it is enough to prove that $F(\hat{x}_i)/\tau \geq \alpha/(1+\alpha) - 2p^{-1}$. As in the proof of Lemma 31, we use $e_1, e_2, \ldots, e_k$ to denote the elements of $\operatorname{supp}(\hat{x}_i)$ in the order of their arrival, and define $\hat{x}_i^j = \hat{x}_i \wedge \mathbf{1}_{\{e_1, e_2, \ldots, e_j\}}$, and $\tilde{q}_i^j = (\hat{x}_i)_j$. This allows us to bound the value of $F(\hat{x}_i)$ as follows.

$$F(\hat{x}_i) = F(\mathbf{1}_\varnothing) + \sum_{j=1}^{k} \left(F(\hat{x}_i^j) - F(\hat{x}_i^{j-1})\right) = F(\mathbf{1}_\varnothing) + \sum_{j=1}^{k} \left(F(\hat{x}_i^{j-1} + \tilde{q}_i^j \cdot \mathbf{1}_{e_j}) - F(\hat{x}_i^{j-1})\right)$$
$$\geq \sum_{j=1}^{k} \frac{(1-4p^{-1})\alpha\tau}{(1+\alpha)k} \geq \left(\frac{\alpha}{1+\alpha} - 2p^{-1}\right)\tau \enspace,$$

where the first inequality follows from the non-negativity of F since the elements in $\operatorname{supp}(\hat{x}_i)$ passed the threshold in Line 6. □

Next, recall that, in Section 6, Corollary 37 gave a bound on the approximation ratio of Algorithm 4 in the case where none of the maintained solutions fills up during the stream. One can verify that if we prove that the guarantees of Lemma 35 also hold for Algorithm 6, then the guarantees of Lemma 36 and Corollary 37 hold for Algorithm 6 as well. We give such a proof in the following lemma, using the notations from Section 6.

Lemma 45. *The guarantees of Lemma 35 apply to Algorithm 6 as well.*

Proof. The only part in the proof of Lemma 35 that relies on properties that Algorithms 4 and 6 do not share is the inequality $\max_{q\in[0,1]}\left(G_a(\hat{x}_i + q\cdot\mathbf{1}_e) - G_a(\hat{x}_i)\right) < c\tau/k$ for every $e \in \operatorname{supp}(o)$ and integer $1 \leq i \leq p$. Thus, it is enough to show that this inequality holds also when considering Algorithm 6. Since the algorithm uses Algorithm 5 to get an estimate $\tilde{q}$ of the argument $q \in [0,1]$ that maximizes $F(x_i + q\cdot\mathbf{1}_e) - F(x_i)$, the rejection of the elements in $\operatorname{supp}(o)$ implies that for every $e \in \operatorname{supp}(o)$ and $1 \leq i \leq p$,

$$\max_{q\in[0,1]}\left(G_a(\hat{x}_i + q\cdot\mathbf{1}_e) - G_a(\hat{x}_i)\right) \leq \max_{q\in[0,1]}\left(F(\hat{x}_i + q\cdot\mathbf{1}_e) - F(\hat{x}_i)\right)$$
$$\leq \frac{F(\hat{x}_i + \tilde{q}\cdot\mathbf{1}_e) - F(\hat{x}_i)}{1-4p^{-1}} < \frac{c\tau}{k} \enspace,$$

where the first inequality follows from the DR-submodularity of F and the inequality $\hat{x}_i \leq \hat{x}_i * a$, which holds for every fixed vector $a \in [0,1]^V$. The second inequality in the display math follows from Proposition 41 and our assumption that $p \geq 5$. □

The discussion above Lemma 45 immediately implies the following corollary, which corresponds to Corollary 37.

Corollary 46. *If $|\operatorname{supp}(\hat{x}_i)| < k$ for every integer $1 \leq i \leq p$, then the output of Algorithm 6 has value of at least $\left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau$.*

Notice that Lemma 44 and Corollary 46 guarantee the same lower bound on the output of the algorithm, and we can summerize these results in the following proposition, which corresponds to Proposition 38 from Section 6.

Proposition 47. *The output of Algorithm 6 has an expected value of at least $\left(\frac{\alpha}{\alpha+1} - 2p^{-1}\right)\tau$.*

Using the guarantees of Observation 43 and Proposition 47 we can now prove the following theorem.

Theorem 48. *Assume there exists an α-approximation offline algorithm* OfflineAlg *for maximizing a non-negative DR-submodular function F subject to a support cardinality constraint whose space complexity is nearly linear in the size of the ground set. Then, for every constant $\varepsilon \in (0, 1]$, there exists a semi-streaming algorithm that assumes access to an estimate τ of $F(x^*)$ obeying $(1-\varepsilon/2) \cdot F(x^*) \leq \tau \leq F(x^*)$ and provides $(\frac{\alpha}{1+\alpha} - \varepsilon)$-approximation for the same problem. This algorithm stores $O(k\varepsilon^{-1})$ elements and uses $O(\varepsilon^{-1} \log \varepsilon^{-1})$ value oracle queries while processing each arriving element.*

Proof. Note that Propositions 38 and 47 guarantee the same approximation ratio as a function of p. Therefore, the proof of Theorem 39 can be modified to prove the current theorem by simply using Proposition 47 instead of Proposition 38. □

References

[1] Naor Alaluf, Alina Ene, Moran Feldman, Huy L. Nguyen, and Andrew Suh. Optimal streaming algorithms for submodular maximization with cardinality constraints. In *ICALP*, pages 6:1–6:19, 2020. doi:10.4230/LIPIcs.ICALP.2020.6.

[2] Naor Alaluf, Alina Ene, Moran Feldman, Huy L. Nguyen, and Andrew Suh. Optimal streaming algorithms for submodular maximization with cardinality constraints. *CoRR*, abs/1911.12959v3, 2020. URL: https://arxiv.org/pdf/1911.12959v3.pdf.

[3] Naor Alaluf, Alina Ene, Moran Feldman, Huy L. Nguyen, and Andrew Suh. Optimal streaming algorithms for submodular maximization with cardinality constraints. *CoRR*, abs/1911.12959v2, 2020. URL: https://arxiv.org/pdf/1911.12959v2.pdf.

[4] Yossi Azar, Iftah Gamzu, and Ran Roth. Submodular max-sat. In *ESA*, pages 323–334, 2011. doi:10.1007/978-3-642-23719-5_28.

[5] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *KDD*, pages 671–680, 2014. URL: http://doi.acm.org/10.1145/2623330.2623637, doi:10.1145/2623330.2623637.

[6] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Trans. Algorithms*, 14(3):32:1–32:20, 2018. doi:10.1145/3184990.

[7] Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a non-symmetric technique. *Mathematics of Operations Research*, 44(3):988–1005, 2019.

[8] Niv Buchbinder, Moran Feldman, Yuval Filmus, and Mohit Garg. Online submodular maximization: Beating 1/2 made simple. In *Integer Programming and Combinatorial Optimization - 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings*, pages 101–114, 2019. doi:10.1007/978-3-030-17953-3_8.

[9] Niv Buchbinder, Moran Feldman, and Mohit Garg. Deterministic ($1/2+\varepsilon$)-approximation for submodular maximization over a matroid. In *SODA*, pages 241–254, 2019. doi:10.1137/1.9781611975482.16.

[10] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, pages 1433–1452, 2014. doi:10.1137/1.9781611973402.106.

[11] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. *ACM Trans. Algorithms*, 15(3):30:1–30:31, 2019. doi:10.1145/3309764.

[12] 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. doi:10.1137/080733991.

[13] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1-2):225–247, 2015. doi:10.1007/s10107-015-0900-7.

[14] T.-H. Hubert Chan, Zhiyi Huang, Shaofeng H.-C. Jiang, Ning Kang, and Zhihao Gavin Tang. Online submodular maximization with free disposal. *ACM Trans. Algorithms*, 14(4):56:1–56:29, 2018. doi:10.1145/3242770.

[15] Chandra Chekuri. Personal communication, 2018.

[16] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *ICALP*, pages 318–330, 2015. doi:10.1007/978-3-662-47672-7_26.

[17] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 318–330. Springer, 2015.

[18] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, pages 575–584, 2010. doi:10.1109/FOCS.2010.60.

[19] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.*, 43(6):1831–1879, 2014. doi:10.1137/110839655.

[20] Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, pages 1057–1064, 2011.

[21] Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue submodularity. *Theory of Computing*, 8(1):95–119, 2012. doi:10.4086/toc.2012.v008a005.

[22] Alina Ene and Huy L Nguyen. Constrained submodular maximization: Beyond 1/e. In *IEEE Foundations of Computer Science (FOCS)*, pages 248–257. IEEE, 2016.

[23] Moran Feldman. Maximizing symmetric submodular functions. *ACM Trans. Algorithms*, 13(3):39:1–39:36, 2017. doi:10.1145/3070685.

[24] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 732–742, 2018.

[25] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *NeurIPS*, pages 730–740, 2018. URL: http://papers.nips.cc/paper/7353-do-less-get-more-streaming-submodular-maximization-with-subsampling.

[26] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Proceedings of the 52nd IEEE Foundations of Computer Science (FOCS)*, pages 570–579. IEEE Computer Society, 2011.

[27] Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *STOC*, pages 1363–1374, 2020.

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

[29] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[30] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 42:427–486, 2011. `doi:10.1613/jair.3278`.

[31] Jason D. Hartline, Vahab S. Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *WWW*, pages 189–198, 2008. `doi:10.1145/1367497.1367524`.

[32] Michael Kapralov, Ian Post, and Jan Vondrák. Online submodular welfare maximization: Greedy is optimal. In *SODA*, pages 1216–1225, 2013. `doi:10.1137/1.9781611973105.88`.

[33] 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 *ICML*, pages 3311–3320, 2019. URL: `http://proceedings.mlr.press/v97/kazemi19a.html`.

[34] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11:105–147, 2015. `doi:10.4086/toc.2015.v011a004`.

[35] 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. `doi:10.1137/15M1051142`.

[36] Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming nonmonotone submodular maximization: Personalized video summarization on the fly. In *AAAI*, pages 1379–1386, 2018. URL: `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17014`.

[37] George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978. `doi:10.1287/moor.3.3.177`.

[38] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Math. Program.*, 14(1):265–294, 1978. doi:10.1007/BF01588971.

[39] Mehraveh Salehi, Amin Karbasi, Dustin Scheinost, and R. Todd Constable. A submodular approach to create individualized parcellations of the human brain. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2017 - 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part I*, pages 478–485, 2017. doi:10.1007/978-3-319-66182-7_55.

[40] Andreas S. Schulz and Nelson A. Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optimization*, 10(2):163–180, 2013. doi:10.1016/j.disopt.2013.02.002.

[41] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013. doi:10.1137/110832318.

A Details about the Error in a Previous Work

As mentioned above, Chekuri et al. [17] described a semi-streaming algorithm for the problem of maximizing a non-negative (not necessarily monotone) submodular function subject to a cardinality constraint, and claimed an approximation ratio of roguhly 0.212 for this algorithm. However, an error was later found in the proof of this result [15] (the error does not affect the other results of [17]). For completeness, we briefly describe in this appendix the error found.

In the proof presented by Chekuri et al. [17], the output set of their algorithm is denoted by $\tilde{S}$. As is standard in the analysis of algorithms based on single-threshold Greedy, the analysis distinguishes between two cases: one case in which $\tilde{S} = k$, and a second case in which $\tilde{S} < k$. To argue about the second case, the analysis then implicitly uses the inequality

$$\mathbb{E}\Big[f(\tilde{S} \cup \mathrm{OPT}) \mid |S| < k\Big] \geq (1 - \max_{e \in V} \Pr[e \in \tilde{S}]) \cdot f(\mathrm{OPT}) \enspace . \tag{2}$$

It is claimed by [17] that this inequality follows from Lemma 4 (originally due to [10]). However, this lemma can yield only the inequalities

$$\mathbb{E}\Big[f(\tilde{S} \cup \mathrm{OPT})\Big] \geq (1 - \max_{e \in V} \Pr[e \in \tilde{S}]) \cdot f(\mathrm{OPT})$$

and

$$\mathbb{E}\Big[f(\tilde{S} \cup \mathrm{OPT}) \mid |S| < k\Big] \geq (1 - \max_{e \in V} \Pr[e \in \tilde{S} \mid |S| < k]) \cdot f(\mathrm{OPT}) \enspace ,$$

which are similar to (2), but do not imply it.

B Inapproximability

In this appendix, we prove an inapproximability result for the problem of maximizing a non-negative submodular function subject to cardinality constraint in the data stream model. This result is given by the next theorem. The proof of the theorem is an adaptation of a proof given by Buchbinder et al. [11] for a similar result applying to an online variant of the same problem.

Theorem 49. *For every constant $\varepsilon > 0$, no data stream algorithm for maximizing a non-negative submodular function subject to cardinality constraint is $(1/2+\varepsilon)$-competitive, unless it uses $\Omega(|V|)$ memory.*

Proof. Let $k \geq 1$ and $h \geq 1$ be two integers to be chosen later, and consider the non-negative submodular function $f\colon 2^V \to \mathbb{R}^+$, where $V = \{u_i\}_{i=1}^{k-1} \cup \{v_i\}_{i=1}^{h} \cup \{w\}$, defined as follows.

$$f(S) = \begin{cases} |S| & \text{if } w \notin S \enspace , \\ k + \left|S \cap \{u_i\}_{i=1}^{k-1}\right| & \text{if } w \in S \enspace . \end{cases}$$

It is clear that f is non-negative. One can also verify that the marginal value of each element in V is non-increasing, and hence, f is submodular.

Let ALG be an arbitrary data stream algorithm for the problem of maximizing a non-negative submodular function subject to a cardinality constraint, and let us consider what happens when we give this algorithm the above function f as input, the last element of V to arrive is the element w and we ask the algorithm to pick a set of size at most k. One can observe that, before the arrival of w, ALG has no way to distinguish between the other elements of V. Thus, if we denote by M the set of elements stored by ALG immediately before the arrival of w and assume that the elements of $V \setminus \{w\}$ arrive at a random order, then every element of $V \setminus \{w\}$ belongs to M with the same probability of $\mathbb{E}[|M|]/|V \setminus \{w\}|$. Hence, there must exist some arrival order for the elements of $V \setminus \{w\}$ guaranteeing that

$$\mathbb{E}\Big[|M \cap \{u_i\}_{i=1}^{k-1}|\Big] = \sum_{i=1}^{k-1} \Pr[u_i \in M] \leq \frac{k \cdot \mathbb{E}[|M|]}{|V \setminus \{w\}|} \enspace .$$

Note now that the above implies that the expected value of the output set produced by ALG given the above arrival order is at most

$$k + \frac{k \cdot \mathbb{E}[|M|]}{|V \setminus \{w\}|} \enspace .$$

In contrast, the optimal solution is the set $\{u_i\}_{i=1}^{k-1} \cup \{w\}$, whose value is $2k-1$. Therefore, the competitive ratio of ALG is at most

$$\frac{k + k \cdot \mathbb{E}[|M|]/|V \setminus \{w\}|}{2k-1} = \frac{1 + \mathbb{E}[|M|]/|V \setminus \{w\}|}{2 - 1/k} \leq \frac{1}{2} + \frac{1}{k} + \frac{\mathbb{E}[|M|]}{|V \setminus \{w\}|} \enspace .$$

To prove the theorem we need to show that, when the memory used by ALG is $o(|V|)$, we can choose large enough values for k and h that will guarantee that the rightmost side of the last inequality is at most $1/2 + \varepsilon$. We do so by showing that the two terms $1/k$ and $\mathbb{E}[|M|]/|V \setminus \{w\}|$ can both be upper bounded by $\varepsilon/2$ when the integers k and h are large enough, respectively. For the term $1/k$ this is clearly the case when k is larger than $2/\varepsilon$. For the term $\mathbb{E}[|M|]/|V \setminus \{w\}|$ this is true because increasing h can make V as large as want, and thus, can make the ratio $\mathbb{E}[|M|]/|V \setminus \{w\}|$ as small as necessary due to our assumption that the memory used by ALG (which includes M) is $o(|V|)$. $\square$

תקציר

אנו חוקרים את בעיית המקסימיזציה של פונקציות תת-מודולריות *לא מונוטוניות* תחת אילוץ ספירה במודל ההזרמה. תרומתנו העיקרית היא אלגוריתם הזרמה (למחצה) שמשתמש בזיכרון של כ- $O\left(\frac{k}{\varepsilon^2}\right)$, כאשר k הוא האילוץ על גודל הקבוצה המקסימלי. בסוף הזרמת הנתונים, האלגוריתם שלנו מפעיל אלגוריתם לא-מקוון *לבחירת המשתמש* על הנתונים ששמר, ופולט פתרון המבטיח יחס קירוב של $\frac{\alpha}{1+\alpha} - \varepsilon$ ביחס לפתרון אופטימלי, כאשר α הוא יחס הקירוב של האלגוריתם הלא-מקוון.

אם נשתמש באלגוריתם לא-מקוון מדויק (ובעל זמן ריצה מעריכי בגודל הקלט) בעיבוד הנתונים השמורים, נקבל יחס קירוב של $\frac{1}{2} - \varepsilon$, שהוא כמעט אופטימלי במודל ההזרמה. לעומת זאת, אם נשתמש באלגוריתם של [7], שמשיג את יחס הקירוב הטוב ביותר שידוע כיום ($\alpha = 0.385$), נקבל אלגוריתם הזרמה שרץ בזמן פולינומי ובעל יחס קירוב של 0.2779; מה שמהווה שיפור על-פני התוצאה הקודמת הטובה ביותר של קירוב 0.1715 שהושגה על-ידי [24].

ראוי לציין שהאלגוריתם שלנו הוא קומבינטורי ודטרמיניסטי, תכונות שנדיר למצוא באלגוריתמים עבור מקסימיזציה של פונקציות תת-מודולריות לא-מונוטוניות, והוא אף נהנה מזמן עדכון מהיר של $O\left(\frac{\log k + \log\left(\frac{1}{\alpha}\right)}{\varepsilon^2}\right)$ לכל איבר קלט. בנוסף, אנו מראים כיצד ניתן לשנות ולהתאים את האלגוריתם העיקרי שלנו בצורה טבעית למקסימיזציה של פונקציות תת-מודולריות רציפות בעלות התכונה של תמורה שולית פוחתת תחת אילוץ ספירה על גודל התמיכה של הפתרון.

תוכן עניינים

האוניברסיטה הפתוחה
המחלקה למתמטיקה ומדעי המחשב

מירוב פונקציה תת-מודולרית
תחת אילוץ ספירה במודל ההזרמה

עבודה זו הוגשה כחלק מהדרישות לקבלת תואר
"מוסמך במדעים" M.Sc. במדעי המחשב
באוניברסיטה הפתוחה
החטיבה למדעי המחשב

על-ידי

נאור אלאלוף

העבודה הוכנה בהנחייתו של פרופ' מורן פלדמן

נובמבר 2020