Fixed-Parameter Tractable Submodular Maximization over a Matroid

Shamisa Nematollahi
CNRS, IRIF, Université Paris Cité, Paris, France

Adrian Vladu
CNRS, IRIF, Université Paris Cité, Paris, France

Junyao Zhao
CNRS, IRIF, Université Paris Cité, Paris, France

Abstract

In this paper, we design fixed-parameter tractable (FPT) algorithms for (non-monotone) submodular maximization subject to a matroid constraint, where the matroid rank r is treated as a fixed parameter that is independent of the total number of elements n. We provide two FPT algorithms: one for the offline setting and another for the random-order streaming setting. Our streaming algorithm achieves a $1/2 - \varepsilon$ approximation using $\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)$ memory, while our offline algorithm obtains a $1 - 1/e - \varepsilon$ approximation with $n \cdot 2^{\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)}$ runtime and $\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)$ memory. Both approximation factors are near-optimal in their respective settings, given existing hardness results. In particular, our offline algorithm demonstrates that – unlike in the polynomial-time regime – there is essentially no separation between monotone and non-monotone submodular maximization under a matroid constraint in the FPT framework.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Submodular maximization, matroids, parameterized complexity, streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.105

Related Version *Full paper*: `https://arxiv.org/abs/2509.01591`

Funding *Shamisa Nematollahi*: Partially supported by the French Agence Nationale de la Recherche (ANR) under grant ANR-21-CE48-0016 (project COMCOPT).
Adrian Vladu: Partially supported by the French Agence Nationale de la Recherche (ANR) under grant ANR-21-CE48-0016 (project COMCOPT).
Junyao Zhao: Supported by a postdoctoral fellowship from the Fondation Sciences Mathématiques de Paris (FSMP).

1 Introduction

Submodular maximization under a matroid constraint is a fundamental problem in combinatorial optimization, where we are given[1] a submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ and a matroid $\mathcal{M} \subseteq 2^{[n]}$ with rank $r \in \mathbb{N}^+$, and our task is to find a set of elements $X \in \mathcal{M}$ that maximizes the function value $f(X)$. For this problem, there is a well-known gap between monotone (i.e., non-decreasing) and non-monotone submodular functions: For monotone submodular functions, the optimal polynomial-time algorithm achieves a $1 - 1/e$ approximation to the optimal solution [8], whereas for non-monotone submodular functions, no polynomial-time

[1] We assume that the submodular function f is given by a value oracle that outputs $f(X)$ for input $X \subseteq [n]$, and the matroid $\mathcal{M}$ is given by a membership oracle that answers whether $X \in \mathcal{M}$ for input $X \subseteq [n]$.

algorithm can obtain a 0.478 approximation [17, 12], even for the special case of the cardinality constraint [25], where the matroid $\mathcal{M}$ simply consists of all subsets of $[n]$ with size at most r.

One of the most important paradigms in theoretical computer science for overcoming the limitations of polynomial-time algorithms is parameterized complexity, which treats a specific component of the problem as a parameter that is fixed at a small value. In the context of matroid-constrained submodular maximization, it is natural to assume that the matroid rank r is a small parameter[2] and design *fixed-parameter tractable* (FPT) algorithms, i.e., algorithms with runtime $h(r) \cdot \mathsf{poly}(n)$, where h can be any finite function.

For the special case of the cardinality constraint, [26] designed an FPT algorithm that guarantees a 0.539 approximation for non-monotone submodular functions, showing a substantial advantage of FPT algorithms over polynomial-time algorithms. On the hardness side, essentially the only known query complexity lower bound that applies to FPT algorithms is from the classic work of [24]. They proved that any algorithm that achieves a better-than-$(1 - 1/e)$ approximation requires $n^{\Omega(r)}$ queries to the value oracle, a result that holds for both monotone and non-monotone submodular maximization with the cardinality constraint. These results seem to suggest that the gap between monotone and non-monotone submodular maximization (at least for the cardinality constraint) could be narrowed further, or perhaps even closed, in the FPT regime. This brings us to the question:

> For matroid/cardinality-constrained non-monotone submodular maximization, what is the best possible approximation ratio that can be achieved by FPT algorithms[3]? Is there a separation between monotone and non-monotone submodular maximization under a matroid/cardinality constraint in the FPT framework?

In this paper, we reveal a surprising answer to this question: There is essentially no separation between monotone and non-monotone submodular maximization under a matroid constraint in the FPT regime. Specifically, we design a nearly $(1 - 1/e)$-approximation FPT algorithm (Algorithm 3) for general (non-monotone) submodular maximization subject to a matroid constraint.

▶ **Theorem 1** (Restatement of Theorem 28). *There is an FPT algorithm that achieves a $1 - 1/e - \varepsilon$ approximation for maximizing any non-negative submodular function under a matroid constraint, with $n \cdot 2^{\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)}$ runtime and $\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)$ memory for any $\varepsilon > 0$.*

The bedrock of our $(1-1/e-\varepsilon)$-approximation FPT algorithm is a subroutine (Subroutine 2) that can be simplified into a random-order semi-streaming algorithm (Algorithm 1), and it has significant implications for the streaming setting as well. Briefly, a random-order streaming algorithm makes a single pass over elements of $[n]$ in a uniformly random order. During the pass, the algorithm maintains a subset of visited elements in a memory buffer of bounded size, and it is allowed to make unlimited queries to the value oracle of the function f and the membership oracle of the matroid $\mathcal{M}$ for any subset of elements stored in its memory. In the literature, there is particular interest in streaming algorithms that use $\widetilde{\mathcal{O}}(r)$ memory, which are known as semi-streaming algorithms.

Our random-order semi-streaming algorithm (Algorithm 1) achieves a nearly $1/2$ approximation. (We note that a nearly $1/2$ approximation is known to be achievable by a streaming algorithm with $2^{\mathcal{O}(r)}$ memory, for the more general setting of matroid intersection constraints

[2] For example, in data summarization [23], given a large image dataset, an algorithm needs to choose a representative subset of images that is small enough to fit the screen of our smartphones.

[3] The special case of this question for the cardinality constraint was posed by [26].

and adversarial-order streams [15].) This matches the lower bound established by [26], which asserts that any random-order streaming algorithm exceeding $1/2$ approximation must use $\Omega\left(\frac{n}{r^2}\right)$ memory, even for the special case of the cardinality constraint.

▶ **Theorem 2** (Restatement of Theorem 16). *There is a random-order semi-streaming algorithm that achieves a $1/2 - \varepsilon$ approximation for maximizing any non-negative submodular function under a matroid constraint, using $\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)$ memory for any $\varepsilon > 0$.*

In comparison, the previous best random-order semi-streaming algorithm for maximizing monotone submodular functions under a matroid constraint achieves a $\left(\frac{1-e^{-2}}{2} \approx 0.432\right)$ approximation [27]. For non-monotone functions, the best known semi-streaming algorithm obtains a 0.1921 approximation [14], though this algorithm applies to the more restrictive adversarial-order streaming setting. We highlight that the algorithms of [27] and [14] run in polynomial time, whereas our algorithm runs in FPT time.

1.1 Overview of our approach

Deferring a more detailed exposition to the main sections, we provide an overview of our algorithms and techniques. As mentioned previously, the core of our $(1-1/e-\varepsilon)$-approximation algorithm (Algorithm 3) is Subroutine 2, which can be simplified into a $(1/2-\varepsilon)$-approximation semi-streaming algorithm (Algorithm 1). We call this subroutine continuous greedy filtering. It is inspired by a filtering technique developed by [26] for cardinality-constrained non-monotone submodular maximization, and a fast continuous greedy algorithm designed by [4] for matroid-constrained monotone submodular maximization.

Continuous greedy filtering

The initial phase of this subroutine maintains a fractional solution over $\frac{1}{\varepsilon}$ epochs. In each epoch $t \in \left[\frac{1}{\varepsilon}\right]$, the subroutine constructs an incremental sequence of integral solutions $S_1^t, \ldots, S_r^t \in \mathcal{M}$ through r selection steps. At each step $i \in [r]$, it samples a small subset of elements. Among the elements that can be added to S_{i-1}^t, it selects the element s_i^t that maximizes the marginal value relative to the current fractional solution x_{i-1}^t (a "weighted average" of the integral solutions $S_r^1, \ldots, S_r^{t-1}, S_{i-1}^t$). The integral solution S_i^t is then updated to be $S_{i-1}^t \cup \{s_i^t\}$.

In the second phase, the subroutine uses the integral solutions constructed in the first phase as a filter to select a subset H of elements from $[n]$. Specifically, for each element $e \in [n]$, e is included in H if it can be added to S_{i-1}^t and its marginal value with respect to x_{i-1}^t exceeds that of element s_i^t, for some $t \in \left[\frac{1}{\varepsilon}\right]$ and $i \in [r]$. A minor detail is that the subroutine rounds the marginal values of e and s_i^t to the closest power of $1+\varepsilon$ before comparing them. This detail will keep the subroutine's memory usage nearly linear in the matroid rank r.

This subroutine achieves a key property: We partition the optimal solution O into two sets $O_L := O \setminus H$ and $O_H := O \cap H$. Then, the subroutine guarantees that there is a feasible subset of $\bigcup_{t=1}^{1/\varepsilon} S_r^t$ that is a $1 - 1/e - \varepsilon$ approximation to the partial optimal solution O_L.

$(1 - 1/e - \varepsilon)$-approximation: a recursive approach

Our $(1 - 1/e - \varepsilon)$-approximation algorithm applies the continuous greedy filtering subroutine recursively on refined instances. Specifically, after the first invocation of the subroutine on the input instance $(f, \mathcal{M})$, if the value of O_H is negligible, then by submodularity, O_L must

account for almost all the optimal value. In this case, because of the aforementioned property, our algorithm can find a $(1 - 1/e - \varepsilon)$-approximate solution through an exhaustive search over all subsets of $\bigcup_{t=1}^{1/\varepsilon} S_r^t$. If otherwise, our algorithm enumerates all subsets of H to identify O_H (the algorithm will encounter O_H in some iteration of the enumeration, even though it does not know what O_H is), and then, it recursively applies the above process to a refined instance $(f', \mathcal{M}')$, where the matroid $\mathcal{M}'$ is obtained by contracting the original matroid $\mathcal{M}$ by the set O_H (see Definition 9), and the function f' is defined by $f'(X) := f(X \cup O_H)$. The high-level intuition is that, after sufficiently many recursions, we will eventually reach the first case, where the value of O'_H (the analogue of O_H in the refined instance) is negligible.

$(1/2 - \varepsilon)$-approximation semi-streaming algorithm: single-epoch filtering

Our random-order semi-streaming algorithm simplifies continuous greedy filtering by performing a single-epoch greedy procedure to construct an integral solution in the first ε fraction of the random stream, followed by a similar filtering phase in the remaining stream. The final solution is obtained by enumerating all subsets of elements selected during these two phases and choosing the best one. In the main body of the paper, we will first present this simpler algorithm and its analysis.

Comparison to [26]

Briefly, [26] first designed a $(1/2 - \varepsilon)$-approximation random-order streaming algorithm with quadratic memory for cardinality-constrained non-monotone submodular maximization, based on the filtering technique. They then identified cases where their streaming algorithm only achieves a $1/2$ approximation, and introduced an offline postprocessing procedure to handle those cases and improve the approximation ratio. Their analysis of this procedure relies on a factor-revealing program to balance different cases. To better leverage this factor-revealing program, they extended their postprocessing procedure to a recursive one, which led to a 0.539-approximation FPT algorithm.

Our semi-streaming algorithm is a generalization of their streaming algorithm to the matroid constraint, with improved memory usage. Compared to their 0.539-approximation FPT algorithm, the two main novelties of our $(1 - 1/e - \varepsilon)$-approximation algorithm, as highlighted above, are (i) the combination of the filtering technique and the continuous greedy algorithm, which enables our subroutine to achieve the aforementioned key property under the matroid constraint, and (ii) a natural recursive procedure that applies the subroutine repeatedly to refine the problem instance. This differs significantly from the recursive procedure of [26], which is used only to postprocess an output produced once by their streaming algorithm.

1.2 Additional related work

FPT submodular maximization

The parameterized complexity of submodular maximization was first studied by [28], who provided FPT approximation schemes for maximizing p-separable monotone submodular functions under the cardinality constraint, assuming that both the cardinality constraint and p are fixed parameters. If the cardinality constraint is the only parameter, without further assumptions, even FPT algorithms cannot obtain a better-than-$(1 - 1/e)$ approximation for monotone submodular functions, both in terms of query complexity [24] and computational complexity

under the Gap-Exponential Time Hypothesis [11, 22]. For non-monotone submodular maximization under the cardinality constraint, [26] gave a 0.539-approximation FPT algorithm, while the current best polynomial-time algorithm achieves a 0.401 approximation [6], which also applies to the matroid constraint.

Streaming submodular maximization

In the random-order streaming setting, [1] designed a nearly $(1 - 1/e)$-approximation semi-streaming algorithm for monotone submodular maximization subject to a cardinality constraint, which was later simplified by [21] to reduce the memory size. Moreover, under the cardinality constraint, [26] designed a quadratic-memory streaming algorithm that achieves a 0.5029 approximation and a $1/2$ approximation for symmetric and asymmetric submodular maximization respectively. Under the matroid constraint, [27] gave a $\frac{1-e^{-2}}{2}$-approximation semi-streaming algorithm for monotone submodular maximization. All these random-order streaming algorithms (and ours) also apply to the (similar but incomparable) secretary with shortlists model [1].

In the adversarial-order streaming setting, [3] designed a nearly $1/2$-approximation semi-streaming algorithm for cardinality-constrained monotone submodular maximization, which was later improved by [20] to achieve a linear memory size. On the hardness side, [19] showed that any streaming algorithm for monotone submodular maximization that achieves an approximation better than $\frac{2}{2+\sqrt{2}}$ under a cardinality constraint must use $\Omega\left(\frac{n}{r^2}\right)$ memory, while attaining an approximation better than $\frac{r}{2r-1}$ under a partition matroid constraint requires $\Omega\left(\frac{n}{r}\right)$ memory. Later, [16] proved that any streaming algorithm achieving a better-than-$1/2$ approximation for cardinality-constrained monotone submodular maximization requires $\Omega\left(\frac{n}{r^3}\right)$ memory. More recently, [2] designed a nearly $1/2$-approximation semi-streaming algorithm for cardinality-constrained non-monotone submodular maximization, which is also an FPT algorithm (although not explicitly stated in their paper). Furthermore, matroid-constrained submodular maximization in the adversarial-order streaming setting was first studied by [9], who provided a $1/4$-approximation semi-streaming algorithm for monotone functions. This was recently improved to a 0.3178 approximation by [14], who also gave a 0.1921 approximation for non-monotone functions. Finally, we mention that [14] developed a multi-pass nearly $(1 - 1/e)$-approximation semi-streaming algorithm for matroid-constrained monotone submodular maximization, which also builds on the continuous greedy algorithm of [4]. However, their approach is very different from ours.

2 Preliminaries

2.1 Submodular functions and matroids

We first introduce submodular functions, matroids, and their associated concepts and properties. Given a ground set $[n]$ and a non-negative discrete function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, for any $X, Y \subseteq [n]$, we let $f(X|Y)$ denote the marginal value of X with respect to Y, i.e., $f(X|Y) := f(X \cup Y) - f(Y)$.

▶ **Definition 3** (submodular function). *A non-negative function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ is submodular, if it satisfies that $f(X|Y) \geq f(X|Z)$ for any $X \subseteq [n]$ and any $Y \subseteq Z \subseteq [n]$.*

For any vector $x \in [0, 1]^n$, we use the notation $\mathcal{R}(x)$ to refer to a random set such that each element of $[n]$ appears in $\mathcal{R}(x)$ independently with probability x_i, and we define the multi-linear extension of a submodular function as follows.

▶ **Definition 4** (multi-linear extension). *The multi-linear extension $F : [0,1]^n \to \mathbb{R}_{\geq 0}$ of a submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ is given by*

$$F(x) := \mathbb{E}[f(\mathcal{R}(x))] = \sum_{S \subseteq [n]} \prod_{i \in S} x_i \cdot \prod_{i \in [n] \setminus S} (1 - x_i) \cdot f(S) \text{ for all } x \in [0,1]^n.$$

Moreover, for any $x, y \in [0,1]^n$ such that $x + y \in [0,1]^n$, we let $F(x|y)$ denote the marginal value of x with respect to y, i.e., $F(x|y) := F(x+y) - F(y)$.

We are interested in the problem of maximizing a non-negative submodular function subject to a matroid constraint. A matroid is a set system that satisfies certain independence structure.

▶ **Definition 5** (matroid). *A set system $\mathcal{M} \subseteq 2^{[n]}$ is a matroid*[4] *if the following conditions hold:*

- **i.** $\emptyset \in \mathcal{M}$.
- **ii.** *If $X \in \mathcal{M}$, then $Y \in \mathcal{M}$ for all $Y \subseteq X$.*
- **iii.** *If $X, Y \in \mathcal{M}$ and $|Y| < |X|$, then there exists $i \in X \setminus Y$ such that $Y \cup \{i\} \in \mathcal{M}$.*

Given a matroid, we define its rank, bases, restriction and contraction as follows.

▶ **Definition 6** (rank). *Given a matroid $\mathcal{M} \subseteq 2^{[n]}$, the rank of $\mathcal{M}$ is $r_{\mathcal{M}} := \max_{X \in \mathcal{M}} |X|$.*

▶ **Definition 7** (base). *Given a matroid $\mathcal{M} \subseteq 2^{[n]}$, we say that a set $B \in \mathcal{M}$ is a base of $\mathcal{M}$, if $B \cup \{i\} \notin \mathcal{M}$ for all $i \in [n]$.*

▶ **Definition 8** (restriction). *Given a matroid $\mathcal{M} \subseteq 2^{[n]}$ and a set $S \subseteq [n]$, we define the restriction of $\mathcal{M}$ to S as $\mathcal{M}_S := \{X \subseteq S \mid X \in \mathcal{M}\}$. It is well-known that the restriction $\mathcal{M}_S$ is a matroid.*

▶ **Definition 9** (contraction). *Given a matroid $\mathcal{M} \subseteq 2^{[n]}$ and a set $S \subseteq [n]$, we let $\mathcal{M}/S$ denote the contraction of $\mathcal{M}$ by S, which is given by $\mathcal{M}/S := \{X \subseteq [n] \setminus S \mid X \cup S' \in \mathcal{M} \text{ for all } S' \subseteq S \text{ such that } S' \in \mathcal{M}\}$. It is well-known that the contraction $\mathcal{M}/S$ is a matroid, and its membership oracle can be efficiently evaluated using the membership oracle of $\mathcal{M}$.*

In matroid-constrained submodular maximization, an algorithm ALG is given a non-negative submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ and a matroid $\mathcal{M} \subseteq 2^{[n]}$, and its task is to find a feasible set $X \in \mathcal{M}$ that maximizes $f(X)$. Throughout the paper, we denote the optimal solution by O, i.e., $O := \arg\max_{X \in \mathcal{M}} f(X)$ (and we reserve $\mathcal{O}$ for the Big-O notation to avoid confusion). We say that ALG achieves an α approximation with $\alpha \in [0,1]$, if its solution set X_{ALG} satisfies that $\mathbb{E}[f(X_{\mathsf{ALG}})] \geq \alpha \cdot f(O)$, where the expectation is taken over the randomness of ALG (and the arrival order of the elements, in case ALG is a random-order streaming algorithm, which we will introduce shortly). Moreover, we assume that the matroid rank $r_{\mathcal{M}}$ is a fixed parameter that does not depend on the total number of elements n (and hence, $r_{\mathcal{M}} = o(n)$[5]), and we are interested in *fixed-parameter tractable* (FPT) algorithms, i.e., algorithms with runtime $h(r_{\mathcal{M}}) \cdot \mathsf{poly}(n)$, where h can be any finite function.

---

[4] In the literature, a matroid is often represented as a pair $([n], \mathcal{M})$, where $[n]$ is the ground set and $\mathcal{M} \subseteq 2^{[n]}$ is the family of independent sets. For simplicity, we refer to the matroid simply as $\mathcal{M}$ throughout this paper.

[5] If $r_{\mathcal{M}} = \Omega(n)$, then an FPT algorithm with $2^{\mathcal{O}(r_{\mathcal{M}})}$ runtime can simply enumerate all subsets of $[n]$, and a streaming algorithm with $\mathcal{O}(r_{\mathcal{M}})$ memory can simply store all elements of $[n]$, rendering the problem trivial.

2.2 Random-order streaming model

Besides the standard offline setting (where all elements in the ground set $[n]$ are presented to the algorithm from the outset), we are also interested in the random-order streaming setting, where elements in $[n]$ arrive in a uniformly random order as a data stream.

In this setting, an algorithm has a memory buffer of a limited size (which is $\widetilde{\mathcal{O}}(r_{\mathcal{M}})$ for semi-streaming algorithms). Each time when a new element arrives, the algorithm decides how to update its memory buffer, i.e., whether to store the new element, remove some previously stored elements, or modify other stored information. At any point, the algorithm can make any number of queries to the value oracle of the submodular function and the membership oracle of the matroid, provided that the input for the query is a subset of elements stored in its memory. At the end of the stream, the algorithm outputs a subset of elements from its memory that satisfies the matroid constraint.

2.3 Other useful notions and lemmata

We use the notations $\mathbf{1}_S \in \{0,1\}^n$ and $\mathbf{1}_e \in \{0,1\}^n$ to represent the indicator vectors for any set $S \subseteq [n]$ and for any element $e \in [n]$ respectively, and let $\mathbf{0}$ denote the all-zero vector. Moreover, we define a rounding operator that will be the key to reducing the memory usage of our algorithms.

▶ **Definition 10** (rounding operator). *Given any finite non-empty set of real numbers I, we use the notation $\lfloor \cdot \rfloor_I$ to denote the rounding operator that maps any real number $a \in \mathbb{R}$ to its greatest lower bound in I, or to the smallest number in I if no lower bound exists, i.e., $\lfloor a \rfloor_I := \max\{i \in I \mid i \leq a\}$ if $a \geq \min I$, and $\lfloor a \rfloor_I := \min I$ otherwise.*

Then, we present two base exchange lemmas for matroids and two sampling lemmas for non-negative submodular functions.

▶ **Lemma 11** ([18, 29]). *Given two bases B_1 and B_2 of a matroid $\mathcal{M} \subseteq 2^{[n]}$, and a partition $B_1 = X_1 \cup Y_1$, there exists a partition $B_2 = X_2 \cup Y_2$ such that $X_1 \cup Y_2$ and $X_2 \cup Y_1$ are both bases of $\mathcal{M}$.*

▶ **Lemma 12** ([5]). *Given two bases B_1 and B_2 of a matroid $\mathcal{M} \subseteq 2^{[n]}$, there exists a bijection $h : B_1 \setminus B_2 \to B_2 \setminus B_1$ such that for all $e \in B_1 \setminus B_2$, $(B_1 \setminus \{e\}) \cup \{h(e)\}$ is a base of $\mathcal{M}$.*

▶ **Lemma 13** ([13, Lemma 2.2]). *Given a non-negative submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ and a set $X \subseteq [n]$, if R is a random subset of X such that every element of X appears in R with probability **exactly** p (not necessarily independently), then we have that $\mathbb{E}[f(R)] \geq (1-p) \cdot f(\emptyset) + p \cdot f(X)$.*

▶ **Lemma 14** ([7, Lemma 2.2]). *Given a non-negative submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ and a set $X \subseteq [n]$, if R is a random subset of X such that every element of X appears in R with probability **at most** p (not necessarily independently), then we have that $\mathbb{E}[f(R)] \geq (1-p) \cdot f(\emptyset)$.*

Finally, we state a lemma that follows from standard rounding schemes [8, 10] for matroid-constrained submodular maximization.

▶ **Lemma 15.** *Given a matroid $\mathcal{M} \in 2^{[n]}$, a submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, and its multi-linear extension $F : [0,1]^n \to \mathbb{R}_{\geq 0}$, for any $x = \sum_{i=1}^{N} \frac{1}{N} \cdot \mathbf{1}_{S_i}$ such that $S_i \in \mathcal{M}$ for all $i \in [N]$, there exists $X \subseteq \bigcup_{i=1}^{N} S_i$ such that $X \in \mathcal{M}$ and $f(X) \geq F(x)$.*

3 A $(1/2 - \varepsilon)$-approximation random-order streaming algorithm

In this section, we present our semi-streaming algorithm, Greedy-Filtering (Algorithm 1), which achieves a $1/2-\varepsilon$ approximation using $\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)$ memory for any submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ and any matroid with rank $r = o(n)$. Algorithm 1 operates in three phases.

Algorithm 1 Greedy-Filtering$(f, \mathcal{M}, r, \pi, \varepsilon)$.

Input : Matroid $\mathcal{M} \subseteq 2^{[n]}$ with rank $r \in \mathbb{N}^+$, submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, uniformly random permutation $\pi : [n] \to [n]$, and parameter $\varepsilon \in \left(0, \frac{1}{2}\right)$.

Throughout the stream π, the algorithm maintains a set T consisting of the top $\left\lceil \frac{\ln(1/\varepsilon)}{\varepsilon} \right\rceil$ elements with the highest singleton values among the elements that have arrived (in case of ties, w.l.o.g., keep the element that arrived earliest);

`// Phase 1`

Denote $V_0 := \{\pi(1), \ldots, \pi(\lceil \varepsilon n \rceil)\}$;

$w \leftarrow \max_{e \in V_0} f(\{e\}) \cdot \frac{r}{\varepsilon}$;

`// Phase 2`

Denote $V_i := \{\pi(j) \mid j \in \{\lceil \varepsilon n \rceil + (i-1) \cdot \left\lceil \frac{\varepsilon n}{r} \right\rceil + 1, \ldots, \lceil \varepsilon n \rceil + i \cdot \left\lceil \frac{\varepsilon n}{r} \right\rceil\}\}$ for all $i \in [r]$;

$s_1 \leftarrow \arg\max_{e \in V_1} f(\{e\})$;

$S_0 \leftarrow \emptyset$ and $S_1 \leftarrow \{s_1\}$;

for $i = 2, \ldots, r$ **do**

  $s_i \leftarrow \arg\max_{e \in V_i \cup \{s_1\} \text{ s.t. } S_{i-1} \cup \{e\} \in \mathcal{M}} f(\{e\} | S_{i-1})$;

  $S_i \leftarrow S_{i-1} \cup \{s_i\}$;

end

`// Phase 3`

$I \leftarrow \{(1+\varepsilon)^i \cdot \frac{\varepsilon^2 w}{r^2} \mid i \in \{0, ..., \lceil 2 \log_{1+\varepsilon}(r/\varepsilon) \rceil\}\}$;

$H \leftarrow \emptyset$;

for $j = \lceil \varepsilon n \rceil + r \cdot \left\lceil \frac{\varepsilon n}{r} \right\rceil + 1, \ldots, n$ **do**

  if $\exists i \in [r]$ *s.t.* $S_{i-1} \cup \{\pi(j)\} \in \mathcal{M}$ *and* $\lfloor f(\{\pi(j)\} | S_{i-1}) \rfloor_I > \lfloor f(\{s_i\} | S_{i-1}) \rfloor_I$ **then**

    $H \leftarrow H \cup \{\pi(j)\}$;

  end

  if $|H| > \frac{r \ln(r/\varepsilon) \cdot |I|}{\varepsilon}$ **then**

    break;

  end

end

$X_{\mathsf{ALG}} \leftarrow \arg\max_{X \subseteq T \cup S_r \cup H \text{ s.t. } X \in \mathcal{M}} f(X)$;

return X_{ALG};

Phase 1: Estimating the highest singleton value

Algorithm 1 first identifies the element with the highest singleton value in the initial ε fraction of the random stream π, and sets w to be $\frac{r}{\varepsilon}$ times the value of this element. It treats w as an upper bound on the value of the most valuable element in $[n]$. If w fails to upper bound this value, we can show that w.h.p., only a few of the most valuable elements contribute non-negligibly to the optimal solution. Hence, to address this, throughout the stream, Algorithm 1 also maintains a set T of the top $\left\lceil \frac{\ln(1/\varepsilon)}{\varepsilon} \right\rceil$ elements with highest singleton values among the elements that have appeared.

Phase 2: Greedily constructing a solution set

Then, Algorithm 1 divides the second ε fraction of the stream into r windows of equal sizes, and greedily constructs a feasible solution set S_r as follows: For each $i \in [r]$, among all elements in the i-th window V_i that can be added to the current solution set S_{i-1} without violating the matroid constraint, Algorithm 1 identifies the element s_i with the highest marginal value with respect to S_{i-1}, and updates $S_i = S_{i-1} \cup \{s_i\}$. If no element in V_i can be added to S_{i-1} without violating the matroid constraint (or all elements in V_i that can be added have negative marginal values with respect to S_{i-1}), the algorithm sets s_i to be s_1, which acts as a dummy element with zero marginal value with respect to S_{i-1}.

Phase 3: Filtering the remaining stream

Algorithm 1 filters elements in the rest of the stream, keeping only a subset H of them: For each remaining element e, Algorithm 1 adds element e to H if, for some $i \in [r]$, element e has a strictly higher marginal value with respect to solution set S_{i-1} than element s_i. Notably, when comparing the marginal values of e and s_i, Algorithm 1 first applies the rounding operator defined in Definition 10 with respect to a set I of geometrically increasing numbers, upper bounded by w. As we will show in the analysis, this coarse comparison of marginal values, based on the rounding operator, is crucial for the algorithm to achieve $1/2 - \varepsilon$ approximation while keeping the memory cost nearly linear in the matroid rank w.h.p. In the algorithm, we also set a hard memory limit at Line 17.

Finally, Algorithm 1 finds the most valuable subset of $T \cup S_r \cup H$ that satisfies the matroid constraint through an exhaustive search and returns it as the final solution. The approximation guarantee of Algorithm 1, along with its memory cost, is formally stated in Theorem 16.

▶ **Theorem 16.** *Given any input submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, matroid $\mathcal{M} \subseteq 2^{[n]}$ with rank $r \in \mathbb{N}^+$, and parameter $\varepsilon \in \left(0, \frac{1}{2}\right)$, Algorithm 1 achieves a $1/2 - \varepsilon'$ approximation using $\mathcal{O}\left(\frac{r \ln(r/\varepsilon)^2}{\varepsilon^2}\right)$ memory, where $\varepsilon' = 8 \cdot \sqrt{2\varepsilon + \frac{2r}{n}}$.*

The memory bound follows from a simple calculation: Throughout the data stream, Algorithm 1 only needs to maintain three sets of elements T, S_r, H (note that it can infer $S_1, \ldots, S_{r-1}$ from S_r). By the construction of T and S_r, we have that $|T| = \left\lceil \frac{\ln(1/\varepsilon)}{\varepsilon} \right\rceil = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ and $|S_r| \leq r$. Moreover, the break condition at Line 17 of Algorithm 1 ensures that $|H| = \mathcal{O}\left(\frac{r \ln(r/\varepsilon) \cdot |I|}{\varepsilon}\right) = \mathcal{O}\left(\frac{r \ln(r/\varepsilon)^2}{\varepsilon^2}\right)$, since $|I| = \left\lceil 2 \log_{1+\varepsilon}\left(\frac{r}{\varepsilon}\right) \right\rceil + 1 = \mathcal{O}\left(\frac{\ln(r/\varepsilon)}{\varepsilon}\right)$. Hence, the total memory cost is $\mathcal{O}\left(\frac{r \ln(r/\varepsilon)^2}{\varepsilon^2}\right)$.

Next, we prove the approximation guarantee in three main steps: First, we show that unless $\max_{e \in [n]} f(\{e\}) \geq \frac{r}{\varepsilon} \cdot \min_{e \in T} f(\{e\})$ (in which case we prove that the most valuable and feasible subset of T is a $1 - \varepsilon$ approximation of the optimal solution), w.h.p., the value w in Algorithm 1 is an upper bound on the value of the most valuable element in $[n]$. Then, we show that not many elements pass the filter in the third phase of Algorithm 1, meaning that w.h.p., the break condition at Line 17 is never met. Finally, in the regular case where w caps the value of the most valuable element and only a few elements pass the filter, we prove that w.h.p., the most valuable and feasible subset of $S_r \cup H$ is almost a $1/2$ approximation of the optimal solution. We implement these three steps in the following three subsections.

3.1 The simple case: $\max_{e\in[n]} f(\{e\}) \geq \frac{r}{\varepsilon} \cdot \min_{e\in T} f(\{e\})$

We first analyze the simple case where $\max_{e\in[n]} f(\{e\}) \geq \frac{r}{\varepsilon} \cdot \min_{e\in T} f(\{e\})$. We note that the set T in Algorithm 1 eventually consists of the top $\left\lceil \frac{\ln(1/\varepsilon)}{\varepsilon} \right\rceil$ elements in $[n]$ with the highest singleton values (and we will only consider this final version of T in the analysis). In Lemma 17, we show that Algorithm 1 achieves a $1-\varepsilon$ approximation in this case.

▶ **Lemma 17.** *If* $\max_{e\in[n]} f(\{e\}) \geq \frac{r}{\varepsilon} \cdot \min_{e\in T} f(\{e\})$*, Algorithm 1 obtains a* $1-\varepsilon$ *approximation.*

The proof is provided in the full version.

Given Lemma 17, we can now focus on the cases where $\max_{e\in[n]} f(\{e\}) < \frac{r}{\varepsilon} \cdot \min_{e\in T} f(\{e\})$. Under this assumption, we show that w.h.p., the value w in Algorithm 1 is an upper bound on $\max_{e\in[n]} f(\{e\})$. Specifically, we define the set $T^+ := \{e' \in [n] \mid f(\{e'\}) \geq \min_{e\in T} f(\{e\})\}$ (note that if there are multiple elements with the same singleton value, T could be a random set because of the random stream, but T^+ is always deterministic). In Lemma 18, we show that w.h.p., at least one element of T^+ appears in the first ε fraction of the random stream π, which implies that $w \geq \max_{e\in[n]} f(\{e\})$, assuming $\max_{e\in[n]} f(\{e\}) < \frac{r}{\varepsilon} \cdot \min_{e\in T} f(\{e\})$.

▶ **Lemma 18.** *Let* E_1 *denote the event that there exists some element of* T^+ *that appears in* $V_0 = \{\pi(1), \ldots, \pi(\lceil \varepsilon n \rceil)\}$*, i.e., the first* ε *fraction of the random stream* π*. Then, we have that* $\Pr[E_1] \geq 1-\varepsilon$*. Moreover, if* $\max_{e\in[n]} f(\{e\}) < \frac{r}{\varepsilon} \cdot \min_{e\in T} f(\{e\})$*, then conditioned on event* E_1*, it holds that* $w \geq \max_{e\in[n]} f(\{e\})$*.*

The proof is provided in the full version.

3.2 The unlikely case: $|H| > \frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}$

Recall that the set H in Algorithm 1 consists of elements that pass the filter in the third phase (throughout the analysis, we will only consider the final version of H). In Lemma 19, we show that w.h.p., $|H| \leq \frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}$, which implies that the break condition at Line 17 of Algorithm 1 is unlikely to be reached.

▶ **Lemma 19.** *Let* E_2 *denote the event that* $|H| \leq \frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}$*. Then,* $\Pr[E_2] \geq 1-\varepsilon$*.*

Now we introduce several key concepts that will play important roles in the proof of Lemma 19. First, we notice that in Algorithm 1, each element s_i chosen in the second phase acts as a *selector* in the third phase: for each element e in the third phase of Algorithm 1, the algorithm will include e in set H if it satisfies the selection condition posed by s_i, namely, if $S_{i-1} \cup \{e\} \in \mathcal{M}$ and $\lfloor f(\{e\}|S_{i-1})\rfloor_I > \lfloor f(\{s_i\}|S_{i-1})\rfloor_I$ (we say that an element $e \in [n]$ is *selected* by the selector s_i if it satisfies this condition). For each $i \in [r]$, we let G_i denote the set of elements selected by s_i among all elements appearing after the i-th window in the second phase of Algorithm 1, i.e.,

$$G_i := \{e \in [n] \setminus (\textstyle\bigcup_{j=0}^{i} V_j) \mid S_{i-1} \cup \{e\} \in \mathcal{M} \text{ and } \lfloor f(\{e\}|S_{i-1})\rfloor_I > \lfloor f(\{s_i\}|S_{i-1})\rfloor_I\} \quad (1)$$

(recall that V_0 is the set of elements in the first ε fraction of the stream, and for each $j \in [r]$, V_j is the set of elements in the j-th window during the second phase of Algorithm 1).

We order the selectors according to their indices: $s_1, \ldots, s_r$, and for each $i \in [r]$, we define

$$H_i := \textstyle\bigcup_{j=1}^{i} G_j \text{ (and we let } H_0 := \emptyset \text{ for completeness).} \quad (2)$$

We say that a selector s_i is *ineffective* if there is an earlier selector s_j for some $j < i$ such that $\lfloor f(\{s_i\}|S_{i-1})\rfloor_I \geq \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$ (otherwise we call s_i an *effective* selector). In Claim 20, we show that, as the name suggests, any element that is selected by an ineffective selector s_i would have already been selected by an earlier selector s_j for some $j < i$, which implies that $H_i = H_{i-1}$.

▷ Claim 20. Suppose that s_i is an ineffective selector, i.e., there exists some $j < i$ such that $\lfloor f(\{s_i\}|S_{i-1})\rfloor_I \geq \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$. Then, any element $e \in [n]$ that satisfies the selection condition of s_i, i.e., $S_{i-1} \cup \{e\} \in \mathcal{M}$ and $\lfloor f(\{e\}|S_{i-1})\rfloor_I > \lfloor f(\{s_i\}|S_{i-1})\rfloor_I$, also satisfies the selection condition of s_j, i.e., $S_{j-1} \cup \{e\} \in \mathcal{M}$ and $\lfloor f(\{e\}|S_{j-1})\rfloor_I > \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$. In particular, this implies that $H_i = H_{i-1}$.

Proof. We establish the first part of the claim by showing that $S_{i-1} \cup \{e\} \in \mathcal{M}$ implies $S_{j-1} \cup \{e\} \in \mathcal{M}$, and $\lfloor f(\{e\}|S_{i-1})\rfloor_I > \lfloor f(\{s_i\}|S_{i-1})\rfloor_I$ implies $\lfloor f(\{e\}|S_{j-1})\rfloor_I > \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$.

First, since $\mathcal{M}$ is a matroid and $S_{j-1} \cup \{e\} \subseteq S_{i-1} \cup \{e\}$, the condition that $S_{i-1} \cup \{e\} \in \mathcal{M}$ implies the condition that $S_{j-1} \cup \{e\} \in \mathcal{M}$.

Moreover, because f is a submodular function and $S_{j-1} \subseteq S_{i-1}$, it holds that $f(\{e\}|S_{j-1}) \geq f(\{e\}|S_{i-1})$, which implies that $\lfloor f(\{e\}|S_{j-1})\rfloor_I \geq \lfloor f(\{e\}|S_{i-1})\rfloor_I$, by monotonicity of the rounding operator $\lfloor\cdot\rfloor_I$ (Definition 10). Hence, the condition that $\lfloor f(\{e\}|S_{i-1})\rfloor_I > \lfloor f(\{s_i\}|S_{i-1})\rfloor_I$ implies that $\lfloor f(\{e\}|S_{j-1})\rfloor_I > \lfloor f(\{s_i\}|S_{i-1})\rfloor_I$. Since we assume that $\lfloor f(\{s_i\}|S_{i-1})\rfloor_I \geq \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$, it follows that $\lfloor f(\{e\}|S_{j-1})\rfloor_I > \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$.

Next, we prove the second part of the claim: $H_i = H_{i-1}$. Recall that every element $e \in G_i$ appears after the i-th window and satisfies the selection condition of s_i, which implies that element e also appears after the j-th window (because $j < i$) and satisfies the selection condition of s_j (because of the first part of the claim). It follows that every element $e \in G_i$ belongs to G_j. Hence, we have that $G_i \subseteq G_j \subseteq H_{i-1}$, since $H_{i-1} = \bigcup_{\ell=1}^{i-1} G_\ell$ and $j \leq i-1$. Finally, $H_i = H_{i-1}$ follows from $H_i = G_i \cup H_{i-1}$ and $G_i \subseteq H_{i-1}$. ◁

Now observe that the set H in Algorithm 1 is a subset of H_r. Hence, to upper bound $|H|$, it suffices to upper bound $|H_r|$. If we could prove that for each $i \in [r]$, $|H_i \setminus H_{i-1}| = \mathcal{O}\left(\frac{r\ln(r/\varepsilon)}{\varepsilon}\right)$ holds w.h.p., conditioned on s_i being an effective selector, then Lemma 19 would follow immediately. Indeed, Claim 20 states that $H_i \setminus H_{i-1} = \emptyset$ for every ineffective selector s_i, and thus, we have that $H_r = \bigcup_{i\in[r] \text{ s.t. } s_i \text{ is an effective selector}} (H_i \setminus H_{i-1})$. Notice that there can be at most $|I|$ effective selectors because of the rounding operator $\lfloor\cdot\rfloor_I$ (specifically, any two distinct effective selectors s_i and s_j must satisfy that $\lfloor f(\{s_i\}|S_{i-1})\rfloor_I \neq \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$, and there are only $|I|$ distinct values in the range of $\lfloor\cdot\rfloor_I$). Therefore, if $|H_i \setminus H_{i-1}| = \mathcal{O}\left(\frac{r\ln(r/\varepsilon)}{\varepsilon}\right)$ holds for every effective selector s_i, then it would follow that $|H_r| = \mathcal{O}\left(\frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}\right)$.

However, there is a caveat to the above argument: $|H_i \setminus H_{i-1}| = \mathcal{O}\left(\frac{r\ln(r/\varepsilon)}{\varepsilon}\right)$ does not always hold w.h.p., conditioned on s_i being an effective selector. It is possible to construct instances where $|H_i \setminus H_{i-1}| = \omega\left(\frac{r\ln(r/\varepsilon)}{\varepsilon}\right)$ always holds, conditioned on s_i being an effective selector. Fortunately, if this is the case, we can show that the probability that s_i is an effective selector is negligible. We formalize this intuition in Lemma 21. The proof of Lemma 21 is provided in the full version.

▶ **Lemma 21.** *For each $i \in [r]$, let k_i denote the number of effective selectors among the first i selectors $s_1, \ldots, s_i$, and let $k_0 = 0$. Then, for all $i \in [r]$, we have that*

$$\Pr\left[|H_i| > k_i \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon} \,\middle|\, \forall\, j \in \{0,\ldots,i-1\},\ |H_j| \leq k_j \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon}\right] \leq \frac{\varepsilon}{r}. \tag{3}$$

We are ready to complete the proof of Lemma 19.

Proof of Lemma 19. First, following the notations in Lemma 21, we upper bound the probability $\Pr\left[|H_r| \leq k_r \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon}\right]$ as follows,

$$\begin{aligned}
&\Pr\left[|H_r| \leq k_r \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon}\right] \\
&\geq \Pr\left[\forall\, i \in [r],\, |H_i| \leq k_i \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon}\right] \\
&= \prod_{i\in[r]} \Pr\left[|H_i| \leq k_i \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon} \,\middle|\, \forall\, j \in \{0,\ldots,i-1\},\, |H_j| \leq k_j \cdot \frac{r\ln(r/\varepsilon)}{\varepsilon}\right] \\
&\geq \left(1-\frac{\varepsilon}{r}\right)^r && \text{(By Lemma 21)} \\
&\geq 1-\varepsilon. && (4)
\end{aligned}$$

Moreover, we notice that the total number of effective selectors k_r is at most $|I|$, because any two distinct effective selectors s_i and s_j must satisfy that $\lfloor f(\{s_i\}|S_{i-1})\rfloor_I \neq \lfloor f(\{s_j\}|S_{j-1})\rfloor_I$ (and there are only $|I|$ distinct values in the range of $\lfloor\cdot\rfloor_I$). Thus, Ineq. ((By Lemma 21)) implies that $\Pr\left[|H_r| \leq \frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}\right] \geq 1-\varepsilon$. The proof finishes by noticing that the set H in Algorithm 1 is a subset of H_r. ◀

3.3 The regular case: $\max_{e\in[n]} f(\{e\}) < \frac{r}{\varepsilon}\cdot\min_{e\in T} f(\{e\})$ and $|H| \leq \frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}$

Now we consider the regular case where $\max_{e\in[n]} f(\{e\}) < \frac{r}{\varepsilon}\cdot\min_{e\in T} f(\{e\})$ and $|H| \leq \frac{r\ln(r/\varepsilon)\cdot|I|}{\varepsilon}$. To analyze this case, we first establish two simple claims. Claim 22 shows that w.h.p., the value of the optimal solution O does not decrease significantly, when combined with any subset of elements in a small fraction of the random stream π.

▷ Claim 22. For any $\eta, \delta \in [0,1]$, let V be a random subset of $[n]$ such that every element of $[n]$ appears in V with probability at most δ (not necessarily independently). Then, with probability at least $1-\frac{\delta}{\eta}$, it holds for all $X \subseteq V$ that $f(X|O) \geq -\eta\cdot f(O)$.

Proof. We define $g : 2^{[n]} \to \mathbb{R}_{\geq 0}$ such that $g(Y) = f(Y\cup O)$ for all $Y \subseteq [n]$. Because f is a non-negative submodular function, it follows that g is also non-negative and submodular. Moreover, we define $X_{\min} := \arg\min_{X\subseteq V} f(X|O)$. Since $X_{\min}$ is a subset of V and each element of $[n]$ appears in V with probability at most δ, it follows that each element of $[n]$ appears in $X_{\min}$ with probability at most δ. Hence, by Lemma 14, we have that $\mathbb{E}[g(X_{\min})] \geq (1-\delta)\cdot g(\emptyset)$, which is equivalent to $\mathbb{E}[f(X_{\min}|O) + f(O)] \geq (1-\delta)\cdot f(O)$. By rearranging, we obtain that $\mathbb{E}[-f(X_{\min}|O)] \leq \delta\cdot f(O)$. Then, by applying Markov's inequality to the random variable $-f(X_{\min}|O)$ (note that this random variable is non-negative because $f(X_{\min}|O) \leq f(\emptyset|O) = 0$), we have that

$$\Pr[-f(X_{\min}|O) \geq \eta\cdot f(O)] \leq \frac{\mathbb{E}[-f(X_{\min}|O)]}{\eta\cdot f(O)} \leq \frac{\delta\cdot f(O)}{\eta\cdot f(O)} = \frac{\delta}{\eta}.$$

Thus, $f(X_{\min}|O) \geq -\eta\cdot f(O)$ holds with probability at least $1-\frac{\delta}{\eta}$, which implies the claim. ◁

Claim 23 bounds the error incurred by the rounding operator $\lfloor\cdot\rfloor_I$. The proof is provided in the full version.

▷ Claim 23. Given $I = \{(1+\varepsilon)^i \cdot \gamma w \mid i \in \{0, ..., \lceil \log_{1+\varepsilon}(1/\gamma) \rceil\}\}$ with $w \geq 0$ and $\varepsilon, \gamma \in (0,1)$, for any two real numbers $a \geq 0$ and $b \leq w$, if $\lfloor a \rfloor_I \geq \lfloor b \rfloor_I$, then we have that $a \geq (1-\varepsilon)b - \gamma w$.

Next, we denote by V_{final} the set of elements that appear in the third phase of Algorithm 1, i.e., $V_{\mathsf{final}} := \{\pi(i) \mid i = \lceil \varepsilon n \rceil + r \cdot \lceil \frac{\varepsilon n}{r} \rceil + 1, \ldots, n\}$. Recall that X_{ALG} denotes the solution set returned by Algorithm 1, and O denotes the optimal solution. In Lemma 24, we show that X_{ALG} is almost a 1/2 approximation of $O \cap V_{\mathsf{final}}$, conditioned on three high-probability events in the regular case.

▶ **Lemma 24.** *Let E_1 and E_2 be the events defined in Lemma 18 and Lemma 19 respectively. Moreover, let E_3 denote the event that $f(X|O) \geq -\eta \cdot f(O)$ for all $X \subseteq \bigcup_{i=1}^{r} V_i$, where $\eta := \sqrt{\frac{2\varepsilon n + 2r}{n}}$. Suppose that the input parameter ε for Algorithm 1 is in the range $\left(0, \frac{1}{2}\right)$, and that $\max_{e \in [n]} f(\{e\}) < \frac{r}{\varepsilon} \cdot \min_{e \in T} f(\{e\})$. Then, conditioned on events E_1, E_2, E_3, Algorithm 1 guarantees that $f(X_{\mathsf{ALG}}) \geq \left(\frac{1}{2} - \varepsilon\right) \cdot (f(O \cap V_{\mathsf{final}}) - 5\eta \cdot f(O))$.*

We provide the proofs of Lemma 24 and Theorem 16 in the full version. Here, we give a simplified proof sketch to illustrate the main idea.

Proof sketch. To highlight the main idea, we make the simplifying assumption that both the set S_r constructed in the second phase of Algorithm 1 and the optimal solution O are bases of matroid $\mathcal{M}$. We further assume that all elements in the optimal solution O arrive in the third phase of Algorithm 1 (in reality, we expect a random $1 - 2\varepsilon$ fraction of them to appear in the third phase, which, by Lemma 13, accounts for $1 - 2\varepsilon$ fraction of the optimal value in expectation).

First, we partition the optimal solution into two sets: $O = O_L \cup O_H$, where $O_L := O \setminus H$ and $O_H := O \cap H$. That is, O_H contains the optimal elements that are included in H by Algorithm 1, and O_L consists of those that are not included in H (note that conditioned on the high-probability event E_2 defined in Lemma 19, these elements were not added to H because they did not satisfy the filter condition at Line 14 of Algorithm 1, not because of the break condition at Line 17). By Lemma 11, we can find a partition $S_r = S_L \cup S_H$ such that $S_L \cup O_H$ and $O_L \cup S_H$ are also bases. Moreover, because both $S_r = S_L \cup S_H$ and $O_L \cup S_H$ are bases, by Lemma 12, there exists a bijection $h : S_L \to O_L$ such that for all $s_i \in S_L$, substituting element s_i in S_r with element $h(s_i)$ yields a new base. In particular, this implies that element $h(s_i)$ could be added to set S_{i-1} without violating the matroid constraint. Despite this, element $h(s_i) \in O_L$ was not selected by the selector s_i (since otherwise it would have been included in H). Thus, it must hold that $f(\{h(s_i)\}|S_{i-1}) \leq f(\{s_i\}|S_{i-1})$ (barring the rounding error incurred by the rounding operator $\lfloor \cdot \rfloor_I$, which can be handled by Claim 23). Hence, we can derive that

$$
\begin{aligned}
f(S_L) &= \sum_{i \in [r] \text{ s.t. } s_i \in S_L} f(\{s_i\}|S_{i-1} \cap S_L) && \text{(By telescoping sum)} \\
&\geq \sum_{i \in [r] \text{ s.t. } s_i \in S_L} f(\{s_i\}|S_{i-1}) && \text{(By submodularity)} \\
&\geq \sum_{i \in [r] \text{ s.t. } s_i \in S_L} f(\{h(s_i)\}|S_{i-1}) \\
&= \sum_{o \in O_L} f(\{o\}|S_{i-1}) && \text{(Since } h : S_L \to O_L \text{ is a bijection)} \\
&\geq f(O_L|S_{i-1}) \geq f(O_L|S_r) && \text{(By submodularity).} \qquad (5)
\end{aligned}
$$

Then, we consider the following two solution sets: S_r and $S_L \cup O_H$. Recall that both S_r and $S_L \cup O_H$ are bases, and thus, they are feasible solutions. Moreover, both S_r and $S_L \cup O_H$ are subsets of $S_r \cup H$, and hence, they are candidate solutions in the final exhaustive search of Algorithm 1. Now we lower bound the sum of the solution values of S_r and $S_L \cup O_H$, as follows,

$$\begin{aligned} f(S_r) + f(S_L \cup O_H) &= f(S_r) + f(S_L) + f(O_H|S_L) \\ &\geq f(S_r) + f(O_L|S_r) + f(O_H|S_L) && \text{(By Ineq. (5))} \\ &\geq f(S_r) + f(O_L|S_r) + f(O_H|O_L \cup S_r) && \text{(By submodularity and } S_L \subseteq S_r\text{)} \\ &= f(O \cup S_r) && \text{(Since } O_L \cup O_H = O\text{)} \\ &\geq (1-\eta) \cdot f(O), \end{aligned}$$

where the last inequality holds because of the event E_3 in the statement of Lemma 24 (which is a high-probability event by Claim 22) and the fact that $S_r \subseteq \bigcup_{i=1}^{r} V_i$. It follows that one of S_r and $S_L \cup O_H$ must be a nearly $1/2$ approximation of the optimal solution O. This establishes Lemma 24.

To complete the proof of Theorem 16, we can assume w.l.o.g. that $\max_{e \in [n]} f(\{e\}) < \frac{r}{\varepsilon} \cdot \min_{e \in T} f(\{e\})$ (because otherwise Theorem 16 follows immediately from Lemma 17), and then combine Lemma 24 with the high-probability bounds we have established for events E_1, E_2, E_3.

4 Interlude: continuous greedy filtering

In this section, we present Continuous-Greedy-Filtering (Subroutine 2), a subroutine that will serve as a building block of our final $(1 - 1/e - \varepsilon)$-approximation offline algorithm. Subroutine 2 combines the techniques used in our streaming algorithm (Algorithm 1) with a fast continuous greedy algorithm introduced by [4]. It operates in three phases.

Phase 1: Computing the highest singleton value

Subroutine 2 first identifies the element in $[n]$ with the highest singleton value and assigns this value to v. Since Subroutine 2 is designed for the offline setting, v can be computed exactly by iterating through all elements in $[n]$.

Phase 2: Constructing a fractional solution using the continuous greedy algorithm

Then, Subroutine 2 uses the continuous greedy algorithm to construct a fractional solution from $\frac{r}{\varepsilon}$ subsampled sets: It runs for $\frac{1}{\varepsilon}$ epochs. In each epoch $t \in \left[\frac{1}{\varepsilon}\right]$, it builds a new integral solution S_r^t from scratch in r steps. In each step $i \in [r]$, it samples a small subset of elements V_i^t. Among the elements in V_i^t that can be added to the current integral solution without violating the matroid constraint, Subroutine 2 identifies the element with the highest "marginal value" (as specified at Line 11) with respect to the current fractional solution. If such an element exists and its marginal value is non-negative, Subroutine 2 includes it in the current integral solution and adds an ε fraction of it to the current fractional solution.

Phase 3: Filtering all elements in $[n]$

Finally, Subroutine 2 filters all elements in $[n]$, selecting a subset H from them: For each element $e \in [n]$, it adds element e to H if, for some step $i \in [r]$ of some epoch $t \in \left[\frac{1}{\varepsilon}\right]$ in the second phase, (i) element e can be added to the integral solution at that step without violating the matroid constraint, and (ii) it has a strictly higher marginal value than element

Subroutine 2 CONTINUOUS-GREEDY-FILTERING$(f, \mathcal{M}, r, \varepsilon)$.

Input: Matroid $\mathcal{M} \subseteq 2^{[n]}$ with rank $r \in \mathbb{Z}_{\geq 0}$, submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ (and its multi-linear extension[a] $F : [0,1]^n \to \mathbb{R}_{\geq 0}$), and parameter ε such that $\frac{1}{\varepsilon} \in \mathbb{N}^+$.

if $r = 0$ **then**
  // In case Subroutine 2 is invoked on an input matroid with rank zero
  $S \leftarrow \emptyset$ and $H \leftarrow \emptyset$;
  return S and H;
// Phase 1
$v \leftarrow \max_{e \in [n]} f(\{e\})$;
// Phase 2
$x^0 \leftarrow \mathbf{0}$;
for $t = 1, \ldots, \frac{1}{\varepsilon}$ **do**
  $x^t \leftarrow x^{t-1}$ and $S_0^t \leftarrow \emptyset$;
  for $i = 1, \ldots, r$ **do**
    Sample a set[b] V_i^t by including each element of $[n]$ independently with probability $\frac{\varepsilon^3}{r}$;
    if $\exists e \in V_i^t$ *s.t.* $S_{i-1}^t \cup \{e\} \in \mathcal{M}$ *and* $F(\varepsilon \cdot \mathbf{1}_e | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) \geq 0$ **then**
      $s_i^t \leftarrow \arg\max_{e \in V_i^t \text{ s.t. } S_{i-1}^t \cup \{e\} \in \mathcal{M}} F(\varepsilon \cdot \mathbf{1}_e | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t})$;
      $S_i^t \leftarrow S_{i-1}^t \cup \{s_i^t\}$;
    else
      $S_i^t \leftarrow S_{i-1}^t$;
    // If s_i^t exists, $\Delta_i^t = \mathbf{1}_{s_i^t}$; otherwise, $\Delta_i^t = \mathbf{0}$ acts as a dummy element
    $\Delta_i^t \leftarrow \mathbf{1}_{S_i^t} - \mathbf{1}_{S_{i-1}^t}$;
  $x^t \leftarrow x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_r^t}$;
// Phase 3
$I \leftarrow \{(1+\varepsilon)^i \cdot \frac{\varepsilon^2 v}{r} \mid i \in \{0, ..., \lceil \log_{1+\varepsilon}(r/\varepsilon) \rceil\}\}$;
$H \leftarrow \emptyset$;
for $j = 1, \ldots, n$ **do**
  if $\exists t \in \left[\frac{1}{\varepsilon}\right], i \in [r]$ *s.t.* $S_{i-1}^t \cup \{j\} \in \mathcal{M}$ *and* $\lfloor F(\varepsilon \cdot \mathbf{1}_j | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) \rfloor_I > \lfloor F(\varepsilon \cdot \Delta_i^t | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) \rfloor_I$ **then**
    $H \leftarrow H \cup \{j\}$;
  if $|H| > \frac{r \ln(r/\varepsilon^2) \cdot |I|}{\varepsilon^4}$ **then**
    break;
$S \leftarrow \bigcup_{t=1}^{1/\varepsilon} S_r^t$;
return S and H;

[a] We note that throughout the subroutine, the multi-linear extension F is evaluated only on fractional solutions with a support of size at most $\frac{r}{\varepsilon}$, and hence can be computed exactly in $2^{\mathcal{O}(r/\varepsilon)}$ time.

[b] We define V_i^t explicitly only for the purpose of the analysis. Instead of explicitly sampling and storing V_i^t, the algorithm can iterate through all elements in $[n]$ and skip each element with probability $1 - \frac{\varepsilon^3}{r}$.

s_i^t with respect to the fractional solution at that step (or has a strictly positive marginal value in case s_i^t does not exist). Here, the marginal values are also compared after applying the rounding operator $\lfloor\cdot\rfloor_I$ with some well-chosen discretization I, in order to keep the memory usage nearly linear in the matroid rank w.h.p. In the subroutine, we also impose a hard memory limit at Line 22. At the end of Subroutine 2, it returns the set H, along with the union $\bigcup_{t=1}^{1/\varepsilon} S_r^t$ of all the integral solutions constructed during the second phase.

Now we let $O \in \mathcal{M}$ denote *any* optimal solution, and define $O_H := O \cap H$ and $O_L := O \setminus H$ (i.e., O_H contains the optimal elements that are selected in the third phase of Subroutine 2, and O_L consists of those that are not selected). In Theorem 25, we show that Subroutine 2 satisfies a property that is crucial for proving the $1 - 1/e - \varepsilon$ approximation guarantee of our final algorithm.

▶ **Theorem 25.** *Given any input submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, matroid $\mathcal{M} \subseteq 2^{[n]}$ with rank $r \in \mathbb{Z}_{\geq 0}$, and parameter $\varepsilon < \frac{1}{4}$ such that $\frac{1}{\varepsilon} \in \mathbb{N}^+$, Subroutine 2 guarantees that with probability at least $1 - 2\varepsilon$, there exists a solution set $X_{\mathsf{ALG}} \subseteq S$ such that $X_{\mathsf{ALG}} \in \mathcal{M}$ and*

$$f(X_{\mathsf{ALG}}) \geq \left(1 - \frac{1}{e}\right) \cdot (f(O_L) - 8\varepsilon \cdot f(O)),$$

where the set S is the first output of Subroutine 2.

To prove Theorem 25, we assume w.l.o.g. that the matroid rank r is non-zero, as the statement holds trivially if $r = 0$ (we include this edge case in Theorem 25 because our final algorithm might invoke Subroutine 2 on a matroid with rank zero). We first establish Lemma 26, which shows that w.h.p., the break condition at Line 22 of Subroutine 2 is never reached. The proof of Lemma 26 relies on techniques similar to those developed in the proof of Lemma 19, which we defer to the full version.

▶ **Lemma 26.** *Let E_1 denote the event that $|H| \leq \frac{r \ln(r/\varepsilon^2) \cdot |I|}{\varepsilon^4}$. Then, we have that $\Pr[E_1] \geq 1 - \varepsilon$.*

Then, in Lemma 27, we show that conditioned on event E_1 and another high-probability event E_2, the fractional solution $x^{1/\varepsilon}$ constructed in Subroutine 2 is a nearly $1 - \frac{1}{e}$ approximation of O_L.

▶ **Lemma 27.** *Suppose that the input parameter ε to Subroutine 2 is in the range $\left(0, \frac{1}{4}\right)$. Let E_1 be the event defined in Lemma 26, and let E_2 denote the event that $f(X|O) \geq -\varepsilon \cdot f(O)$ for all $X \subseteq \bigcup_{t \in \left[\frac{1}{\varepsilon}\right], i \in [r]} V_i^t$, where the sets V_i^t are the subsampled sets in Subroutine 2. Then, conditioned on events E_1 and E_2, Subroutine 2 guarantees that $F(x^{1/\varepsilon}) \geq \left(1 - \frac{1}{e}\right) \cdot (f(O_L) - 8\varepsilon \cdot f(O))$.*

We defer the proof of Lemma 27 to the full version and provide a simplified proof sketch.

Proof sketch. To illustrate the main idea, we make the simplifying assumption that the integral solution sets S_r^t constructed in Subroutine 2 for all $t \in \left[\frac{1}{\varepsilon}\right]$, and the optimal solution O are all bases of matroid $\mathcal{M}$.

Recall that the optimal solution O is split into O_L and O_H, where O_H contains the optimal elements that are included in H by Subroutine 2, and O_L consists of those that are not included in H (note that conditioned on the high-probability event E_1 defined in Lemma 26, these elements were not added to H because they did not satisfy the filter condition at Line 20 of Subroutine 2, not because of the break condition at Line 22). Given the partition $O = O_L \cup O_H$, for each $t \in \left[\frac{1}{\varepsilon}\right]$, we can find a partition $S_r^t = S_L^t \cup S_H^t$ such that

$S_L^t \cup O_H$ and $O_L \cup S_H^t$ are also bases. Moreover, because both $S_r^t = S_L^t \cup S_H^t$ and $O_L \cup S_H^t$ are bases, by Lemma 12, there exists a bijection $h_t : S_L^t \to O_L$ such that for all $s_i^t \in S_L^t$, replacing element s_i^t in S_r^t with element $h_t(s_i^t)$ results in a new base. In particular, this implies that element $h_t(s_i^t)$ could be added to set S_{i-1}^t without violating the matroid constraint. Despite this, element $h_t(s_i^t) \in O_L$ did not satisfy the selection condition at Line 20 corresponding to Δ_i^t (since otherwise it would have been included in H). This implies that

$$F(\varepsilon \cdot \mathbf{1}_{h_t(s_i^t)} | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) \leq F(\varepsilon \cdot \Delta_i^t | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) = F(\varepsilon \cdot \mathbf{1}_{s_i^t} | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) \quad (6)$$

(ignoring the error incurred by the rounding operator $\lfloor \cdot \rfloor_I$, which can be addressed by Claim 23). Now we derive that

$$\begin{aligned}
& F(x^t) - F(x^{t-1}) && (7)\\
&= \sum_{i \in [r]} F(\varepsilon \cdot \mathbf{1}_{s_i^t} | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) && \text{(By telescoping sum)}\\
&\geq \sum_{i \in [r] \text{ s.t. } s_i^t \in S_L^t} F(\varepsilon \cdot \mathbf{1}_{s_i^t} | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) && \text{(By the if condition at Line 10)}\\
&\geq \sum_{i \in [r] \text{ s.t. } s_i^t \in S_L^t} F(\varepsilon \cdot \mathbf{1}_{h_t(s_i^t)} | x^{t-1} + \varepsilon \cdot \mathbf{1}_{S_{i-1}^t}) && \text{(By Ineq. (6))}\\
&\geq \sum_{i \in [r] \text{ s.t. } s_i^t \in S_L^t} F(\varepsilon \cdot \mathbf{1}_{h_t(s_i^t)} | x^t) && \text{(By submodularity)}\\
&= \sum_{o \in O_L} F(\varepsilon \cdot \mathbf{1}_o | x^t) && \text{(Since } h_t : S_L^t \to O_L \text{ is a bijection).} \quad (8)
\end{aligned}$$

By Definition 4, $F(\varepsilon \cdot \mathbf{1}_o | x^t)$ represents the additional value gained by increasing the probability of element o appearing in the random set $\mathcal{R}(x^t)$ by ε, and thus, $F(\varepsilon \cdot \mathbf{1}_o | x^t) = \varepsilon \cdot \mathbb{E}[f(\{o\} | \mathcal{R}(x^t) \setminus \{o\})]$. Combining this with Ineq. (7), we obtain that

$$\begin{aligned}
F(x^t) - F(x^{t-1}) &\geq \sum_{o \in O_L} \varepsilon \cdot \mathbb{E}[f(\{o\} | \mathcal{R}(x^t) \setminus \{o\})]\\
&\geq \sum_{o \in O_L} \varepsilon \cdot \mathbb{E}[f(\{o\} | \mathcal{R}(x^t))] && \text{(By submodularity)}\\
&\geq \varepsilon \cdot \mathbb{E}[f(O_L | \mathcal{R}(x^t))] && \text{(By submodularity)}\\
&= \varepsilon \cdot (\mathbb{E}[f(O_L \cup \mathcal{R}(x^t))] - F(x^t)) && \text{(By Definition 4)}\\
&\geq \varepsilon \cdot (f(O_L) - \varepsilon \cdot f(O) - F(x^t)), && (9)
\end{aligned}$$

where the last inequality follows because conditioned on event E_2 in the statement of Lemma 27, the random set $\mathcal{R}(x^t)$ (which is always a subset of $\bigcup_{t \in [\frac{1}{\varepsilon}], i \in [r]} V_i^t$ in Subroutine 2) satisfies that $f(\mathcal{R}(x^t) | O) \geq -\varepsilon \cdot f(O)$.

Then, by iteratively applying Ineq. (9) from $t = 1$ to $t = \frac{1}{\varepsilon}$ (as in the analysis of the continuous greedy algorithm, with the difference being that the approximation guarantee here is with respect to $f(O_L) - \varepsilon \cdot f(O)$ rather than $f(O)$), we can show that $F(x^{1/\varepsilon}) \geq \left(1 - \frac{1}{e}\right) \cdot (f(O_L) - c \cdot \varepsilon \cdot f(O))$ for some constant c, which establishes Lemma 27.

Finally, Theorem 25 follows by combining Lemma 27 with Lemma 26 and Claim 22.

Proof of Theorem 25. We assume w.l.o.g. that the matroid rank r is non-zero. First, we show that the event E_2 defined in Lemma 27 occurs w.h.p. Notice that in Subroutine 2, for any $t \in \left[\frac{1}{\varepsilon}\right]$ and $i \in [r]$, the probability that an element $e \in [n]$ appears in the subsampled set

V_i^t is $\frac{\varepsilon^3}{r}$. By a union bound, the probability that element e appears in $\bigcup_{t\in\left[\frac{1}{\varepsilon}\right],\, i\in[r]} V_i^t$ is at most $\frac{r}{\varepsilon}\cdot\frac{\varepsilon^3}{r} = \varepsilon^2$. Therefore, it follows from Claim 22 that event E_2 occurs with probability at least $1-\varepsilon$. Then, combining this with Lemma 26 through a union bound, the probability that both events E_1 and E_2 occur is at least $1-2\varepsilon$. Thus, by Lemma 27, it holds with probability at least $1-2\varepsilon$ that $F(x^{1/\varepsilon}) \geq \left(1-\frac{1}{e}\right)\cdot(f(O_L) - 8\varepsilon\cdot f(O))$. This implies Theorem 25 by Lemma 15, because $x^{1/\varepsilon} = \sum_{t=1}^{1/\varepsilon} \varepsilon\cdot\mathbf{1}_{S_r^t}$ and $S = \bigcup_{t=1}^{1/\varepsilon} S_r^t$. ◀

On a separate note, Subroutine 2 itself does not guarantee a better-than-$1/2$ approximation. This is because Subroutine 2 can be interpreted as an $\widetilde{\mathcal{O}}\left(\frac{r}{\mathsf{poly}(\varepsilon)}\right)$-memory random-order streaming algorithm (by estimating the highest singleton value as in Algorithm 1 and replacing the subsampled sets with small segments of the random stream), and the hardness result by [26] asserts that such random-order streaming algorithms cannot exceed $1/2$ approximation.

5 A $(1 - 1/e - \varepsilon)$-approximation offline algorithm

In this section, we present our final algorithm (Algorithm 3) that achieves a nearly $1-1/e$ approximation by recursively applying Continuous-Greedy-Filtering (Subroutine 2) to progressively refined instances. Specifically, Algorithm 3 performs a recursive process consisting of $\frac{1}{\alpha}$ levels.

Algorithm 3 Recursive-Continuous-Greedy-Filtering$(f, \mathcal{M}, r, S^{\mathsf{acc}}, O_H^{\mathsf{acc}}, \alpha, k)$.

Input: Matroid $\mathcal{M}\subseteq 2^{[n]}$ with rank $r\in\mathbb{N}^+$, submodular function $f: 2^{[n]}\to\mathbb{R}_{\geq 0}$, sets $S^{\mathsf{acc}}, O_H^{\mathsf{acc}}\subseteq[n]$, parameter α such that $\frac{1}{\alpha}\in\mathbb{N}^+$, and depth counter $k\in\left[\frac{1}{\alpha}\right]$.

Define $f^{(k)}: 2^{[n]\setminus O_H^{\mathsf{acc}}}\to\mathbb{R}_{\geq 0}$ such that $f^{(k)}(X) := f(X\cup O_H^{\mathsf{acc}})$ for all $X\subseteq[n]\setminus O_H^{\mathsf{acc}}$;
Define $\mathcal{M}^{(k)}\subseteq 2^{[n]\setminus O_H^{\mathsf{acc}}}$ as $\mathcal{M}^{(k)} := \mathcal{M}/O_H^{\mathsf{acc}}$;
$S^{(k)}, H^{(k)} \leftarrow$ Continuous-Greedy-Filtering$(f^{(k)}, \mathcal{M}^{(k)}, r_{\mathcal{M}^{(k)}}, \alpha^2)$;
  `// Subroutine 2`
$S^{\mathsf{acc}} \leftarrow S^{\mathsf{acc}}\cup S^{(k)}$;
if $k = \frac{1}{\alpha}$ **then**
  $X_{\mathsf{ALG}} \leftarrow \arg\max_{X\subseteq S^{\mathsf{acc}}\cup O_H^{\mathsf{acc}} \text{ s.t. } X\in\mathcal{M}} f(X)$;
  return X_{ALG};
$X_{\mathsf{ALG}} \leftarrow \emptyset$;
for $O_H^{(k)}\subseteq H^{(k)}$ **do**
  $O_H^{\mathsf{acc}'} \leftarrow O_H^{\mathsf{acc}}\cup O_H^{(k)}$;
  $X'_{\mathsf{ALG}} \leftarrow$ Recursive-Continuous-Greedy-Filtering$(f, \mathcal{M}, r, S^{\mathsf{acc}}, O_H^{\mathsf{acc}'}, \alpha, k+1)$;
  if $f(X'_{\mathsf{ALG}}) > f(X_{\mathsf{ALG}})$ **then**
    $X_{\mathsf{ALG}} \leftarrow X'_{\mathsf{ALG}}$;
return X_{ALG};

Initial phase

We initiate the first call to Algorithm 3 by setting $k=1$, $S^{\mathsf{acc}}=\emptyset$, and $O_H^{\mathsf{acc}}=\emptyset$. We denote the optimal solution to the input instance $(f,\mathcal{M})$ by $O^{(1)}$. At the root level $k=1$, Algorithm 3 first invokes Subroutine 2 with parameter $\varepsilon=\alpha^2$ on the input instance $(f,\mathcal{M})$,

which returns two sets $S^{(1)}$ and $H^{(1)}$. Then, Algorithm 3 enumerate all subsets of $H^{(1)}$ in the for loop at Line 9. Crucially, there will be an iteration of this loop where $O_H^{(1)} = O^{(1)} \cap H^{(1)}$ (all other iterations will be irrelevant to the analysis). Within this particular iteration (as well as in all other iterations), Algorithm 3 makes a recursive call to itself by setting $S^{\mathsf{acc}} = S^{(1)}$ and $O_H^{\mathsf{acc}} = O_H^{(1)}$. These sets will be accumulated throughout the recursion.

Recursion phase

At each internal recursion level $k \in \{2, \ldots, \frac{1}{\alpha} - 1\}$, Algorithm 3 receives the accumulated sets $S^{\mathsf{acc}} = \bigcup_{j=1}^{k-1} S^{(j)}$ and $O_H^{\mathsf{acc}} = \bigcup_{j=1}^{k-1} O_H^{(j)}$ from the previous $k-1$ levels. It first constructs a refined instance $(f^{(k)}, \mathcal{M}^{(k)})$, where the matroid $\mathcal{M}^{(k)}$ is the contraction of the original matroid $\mathcal{M}$ by the accumulated set $\bigcup_{j=1}^{k-1} O_H^{(j)}$, and the non-negative submodular function $f^{(k)}$ is constructed by evaluating the original function f on the union of the accumulated set $\bigcup_{j=1}^{k-1} O_H^{(j)}$ and the input set to $f^{(k)}$. We define $O^{(k)} := O^{(1)} \setminus \left(\bigcup_{j=1}^{k-1} O_H^{(j)}\right)$. Next, Algorithm 3 calls Subroutine 2 with parameter $\varepsilon = \alpha^2$ on the instance $(f^{(k)}, \mathcal{M}^{(k)})$, which returns two sets $S^{(k)}$ and $H^{(k)}$. Then, Algorithm 3 enumerates all subsets of $H^{(k)}$. Crucially, there will be an iteration of this loop where $O_H^{(k)} = O^{(k)} \cap H^{(k)}$ (all other iterations will be irrelevant to the analysis). In this particular iteration (as well as in all other iterations), Algorithm 3 makes a recursive call to itself, passing the updated accumulated sets $S^{\mathsf{acc}} = \bigcup_{j=1}^{k} S^{(j)}$ and $O_H^{\mathsf{acc}} = \bigcup_{j=1}^{k} O_H^{(j)}$ as arguments.

It is important to keep in mind that, in the recursion tree of Algorithm 3 (illustrated in Figure 1), there is a unique root-to-leaf path such that for each $k \in \left[\frac{1}{\alpha} - 1\right]$, the $(k+1)$-th call to Algorithm 3 along the path (corresponding to the $(k+1)$-th node in the path) is invoked by the iterate $O_H^{(k)} = O^{(k)} \cap H^{(k)}$ (from the for loop at Line 9) during the k-th call, where $O^{(k)} = O^{(1)} \setminus \left(\bigcup_{j=1}^{k-1} O_H^{(j)}\right)$. We will focus on this particular path when we prove the approximation guarantee of Algorithm 3.

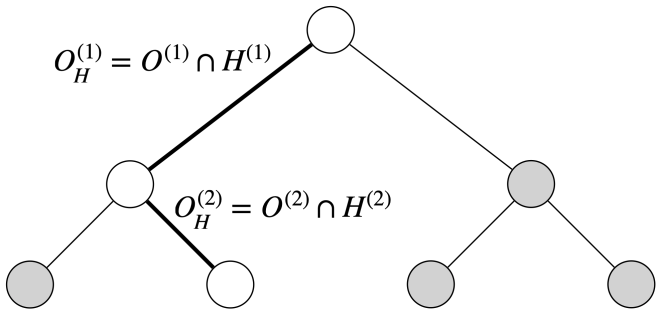

Figure 1 This figure shows an example depth-2 recursion tree of Algorithm 3. The empty nodes form a unique path (made bold in the figure) in which, for each $k \in [2]$, the $(k+1)$-th call is invoked by the iterate $O_H^{(k)} = O^{(k)} \cap H^{(k)}$ during the k-th call, where $O^{(k)} = O^{(1)} \setminus \left(\bigcup_{j=1}^{k-1} O_H^{(j)}\right)$.

Return phase

At the leaf level $k = \frac{1}{\alpha}$, Algorithm 3 first calls Subroutine 2 on a refined instance $(f^{(1/\alpha)}, \mathcal{M}^{(1/\alpha)})$ (constructed similarly to the instances in the previous levels), which returns two sets $S^{(1/\alpha)}$ and $H^{(1/\alpha)}$. Then, it conducts an exhaustive search to find the most valuable subset X_{ALG} of $S^{\mathsf{acc}} \cup O_H^{\mathsf{acc}} = \left(\bigcup_{j=1}^{1/\alpha} S^{(j)}\right) \cup \left(\bigcup_{j=1}^{1/\alpha - 1} O_H^{(j)}\right)$ that belongs to matroid $\mathcal{M}$, and

returns X_{ALG} to the previous level. Then, at each level $k \in \left[\frac{1}{\alpha} - 1\right]$, among all solution sets returned by the recursive calls, Algorithm 3 identifies and returns the most valuable one to the previous level.

Time and space complexity

Now we analyze the time and space complexity of Algorithm 3 (assuming that we set $k = 1$, $S^{\mathsf{acc}} = \emptyset$, and $O_H^{\mathsf{acc}} = \emptyset$ in the initial call). We first observe that at each level $k \in \left[\frac{1}{\alpha}\right]$, the sets $S^{(k)}$ and $H^{(k)}$, which are outputs of Subroutine 2 with parameter α^2, have sizes $|S^{(k)}| \leq \frac{r}{\alpha^2}$ (because $S^{(k)}$ is either the union of $\frac{1}{\alpha^2}$ solution sets constructed in the second phase of Subroutine 2 or an empty set) and $|H^{(k)}| = \mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$ (because of the break condition at Line 22 of Subroutine 2). From this, we observe the following:

- **(i)** At each leaf node of the recursion tree of Algorithm 3, the accumulated set $S^{\mathsf{acc}} = \bigcup_{k=1}^{1/\alpha} S^{(k)}$ has size $|S^{\mathsf{acc}}| \leq \frac{r}{\alpha^3}$ (because each set $S^{(k)}$ has size at most $\frac{r}{\alpha^2}$), and the accumulated set $O_H^{\mathsf{acc}} = \bigcup_{k=1}^{1/\alpha} O_H^{(k)}$ has size $|O_H^{\mathsf{acc}}| = \mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$ (because $|O_H^{(k)}| \leq |H^{(k)}| = \mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$).
- **(ii)** Each non-leaf node of the recursion tree has at most $2^{\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)}$ child nodes (because each set $H^{(k)}$ has size $\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$), which implies that the total number of leaves is $2^{\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)}$, since the recursion tree has $\frac{1}{\alpha}$ levels in total.

Then, we notice that the runtime of each call to Subroutine 2 at each node is $n \cdot 2^{\mathcal{O}(r/\alpha^2)}$, where the $2^{\mathcal{O}(r/\alpha^2)}$ factor accounts for the exact computation of the multi-linear extension (which, if desired, can be replaced by polynomial-time estimation using standard Monte-Carlo sampling). Moreover, the exhaustive search step at each leaf node takes $2^{\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)}$ time, since $|S^{\mathsf{acc}}| + |O_H^{\mathsf{acc}}| = \mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$. Hence, summing the runtime over all nodes, we obtain that the total runtime is $n \cdot 2^{\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)}$. For the space complexity, we assume that Algorithm 3 traverses the recursion tree in depth-first order. We notice that at any node in any level $\ell \in \left[\frac{1}{\alpha}\right]$ of the recursion tree, Algorithm 3 only needs to store the sets $S^{(k)}, H^{(k)}, O_H^{(k)}$ for all $k \in [\ell]$ from its ancestor nodes and itself, and among $S^{(k)}, H^{(k)}, O_H^{(k)}$, the set $H^{(k)}$ has the largest size, which is $\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$. Therefore, the space complexity is $\mathcal{O}\left(\frac{r\ln(r)^2}{\mathsf{poly}(\alpha)}\right)$.

Next, in Theorem 28, we show that Algorithm 3 achieves a nearly $1 - 1/e$ approximation. At a high level, the argument is as follows: Consider the specific path in the recursion tree of Algorithm 3, as illustrated in Figure 1, but with depth $\frac{1}{\alpha}$ for arbitrarily small α. Suppose that for all $k \in \left[\frac{1}{\alpha}\right]$, the marginal value of the set $O_H^{(k)}$ with respect to $\bigcup_{j=1}^{k-1} O_H^{(j)}$ is at least α fraction of the optimal value, i.e., at least $\alpha \cdot f(O^{(1)})$. Then, the union $\bigcup_{j=1}^{1/\alpha} O_H^{(j)}$, which is a candidate solution in the exhaustive search step at the leaf node, must capture the entire optimal value. On the other hand, if for any $k \in \left[\frac{1}{\alpha}\right]$, the marginal value of the set $O_H^{(k)}$ with respect to $\bigcup_{j=1}^{k-1} O_H^{(j)}$ is less than α fraction of the optimal value, then by submodularity, the set $O^{(1)} \setminus O_H^{(k)}$ should account for the majority of the optimal value. We can show that (using Theorem 28) w.h.p., the union of the set $\bigcup_{j=1}^{k-1} O_H^{(j)}$ and a subset of the first output of Subroutine 2 during the k-th call along the path, which is a candidate solution in the exhaustive search step, is a nearly $1 - 1/e$ approximation to $O^{(1)} \setminus O_H^{(k)}$. The complete proof is provided in the full version.

▶ **Theorem 28.** *Given any input submodular function* $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$*, matroid* $\mathcal{M} \subseteq 2^{[n]}$ *with rank* $r \in \mathbb{N}^+$*, and parameter* $\alpha \in \left(0, \frac{1}{2}\right)$ *such that* $\frac{1}{\alpha} \in \mathbb{N}^+$*, by setting* $S^{\mathsf{acc}} = \emptyset$*,* $O_H^{\mathsf{acc}} = \emptyset$*, and* $k = 1$*, Algorithm 3 obtains a* $1 - \frac{1}{e} - 7\alpha$ *approximation with* $n \cdot 2^{\mathcal{O}\left(\frac{r \ln(r)^2}{\mathsf{poly}(\alpha)}\right)}$ *runtime and* $\mathcal{O}\left(\frac{r \ln(r)^2}{\mathsf{poly}(\alpha)}\right)$ *memory.*

References

1. Shipra Agrawal, Mohammad Shadravan, and Cliff Stein. Submodular secretary problem with shortlists. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.1.
2. Naor Alaluf, Alina Ene, Moran Feldman, Huy L Nguyen, and Andrew Suh. An optimal streaming algorithm for submodular maximization with a cardinality constraint. *Mathematics of Operations Research*, 47(4), 2022. doi:10.1287/moor.2021.1224.
3. Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014. doi:10.1145/2623330.2623637.
4. Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014. doi:10.1137/1.9781611973402.110.
5. Richard A Brualdi. Comments on bases in dependence structures. *Bulletin of the Australian Mathematical Society*, 1(2), 1969. doi:10.1017/S000497270004140X.
6. Niv Buchbinder and Moran Feldman. Constrained submodular maximization via new bounds for dr-submodular functions. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, 2024. doi:10.1145/3618260.3649630.
7. Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014. doi:10.1137/1.9781611973402.106.
8. Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6), 2011. doi:10.1137/080733991.
9. Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154, 2015. doi:10.1007/s10107-015-0900-7.
10. Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010. doi:10.1109/FOCS.2010.60.
11. Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight fpt approximations for k-median and k-means. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.42.
12. Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, 2012. doi:10.1145/2213977.2214076.
13. Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4), 2011. doi:10.1137/090779346.
14. Moran Feldman, Paul Liu, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. Streaming submodular maximization under matroid constraints. *Mathematics of Operations Research*, 2025. doi:10.1287/moor.2023.0276.

15 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. Submodular Maximization Subject to Matroid Intersection on the Fly. In *30th Annual European Symposium on Algorithms (ESA 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.52.

16 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. *Journal of the ACM*, 70(4), 2023. doi:10.1145/3588564.

17 Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2011. doi:10.1137/1.9781611973082.83.

18 Curtis Greene. A multiple exchange property for bases. *Proceedings of the American Mathematical Society*, 39(1), 1973. doi:10.1090/S0002-9939-1973-0311494-8.

19 Chien Chung Huang, Naonori Kakimura, Simon Mauras, and Yuichi Yoshida. Approximability of monotone submodular function maximization under cardinality and matroid constraints in the streaming model. *SIAM Journal on Discrete Mathematics*, 2022. doi:10.1137/20M1357317.

20 Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning*. PMLR, 2019. URL: https://proceedings.mlr.press/v97/kazemi19a.html.

21 Paul Liu, Aviad Rubinstein, Jan Vondrák, and Junyao Zhao. Cardinality constrained submodular maximization for random streams. *Advances in Neural Information Processing Systems*, 34, 2021. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/333222170ab9edca4785c39f55221fe7-Paper.pdf.

22 Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020. doi:10.1137/1.9781611975994.5.

23 Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. doi:10.1609/aaai.v32i1.11529.

24 George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3), 1978. doi:10.1287/moor.3.3.177.

25 Benjamin Qi. On maximizing sums of non-monotone submodular and linear functions. *Algorithmica*, 86(4), 2024. doi:10.1007/s00453-023-01183-3.

26 Aviad Rubinstein and Junyao Zhao. Maximizing non-monotone submodular functions over small subsets: Beyond 1/2-approximation. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.106.

27 Mohammad Shadravan. Improved submodular secretary problem with shortlists. *arXiv preprint arXiv:2010.01901*, 2020. arXiv:2010.01901.

28 Piotr Skowron. FPT approximation schemes for maximizing submodular functions. *Information and Computation*, 257, 2017. doi:10.1016/j.ic.2017.10.002.

29 Douglas R Woodall. An exchange theorem for bases of matroids. *Journal of Combinatorial Theory, Series B*, 16(3), 1974. doi:10.1016/0095-8956(74)90067-7.