Journal of Machine Learning Research 26 (2025) 1-28 Submitted 9/23; Published 3/25

Deletion Robust Non-Monotone Submodular Maximization over Matroids

Paul Dütting DUETTING@GOOGLE.COM
Google Research, Zürich, Switzerland

Federico Fusco FUSCOF@DIAG.UNIROMA1.IT
Department of Computer, Control and Management Engineering "Antonio Ruberti" Sapienza University of Rome Rome, Italy

Silvio Lattanzi SILVIOL@GOOGLE.COM
Google Research, Barcelona, Spain

Ashkan Norouzi-Fard ASHKANNOROUZI@GOOGLE.COM
Google Research, Zürich, Switzerland

Morteza Zadimoghaddam ZADIM@GOOGLE.COM
Google Research, Zürich, Switzerland

Editor: Pradeep Ravikumar

Abstract

We study the deletion robust version of submodular maximization under matroid constraints. The goal is to extract a small-size summary of the data set that contains a high-value independent set even after an adversary deletes some elements. We present constant-factor approximation algorithms, whose space complexity depends on the rank k of the matroid, the number d of deleted elements, and the input precision ε. In the centralized setting we present a $(4.494 + O(\varepsilon))$-approximation algorithm with summary size $O(\frac{k+d}{\varepsilon^2} \log \frac{k}{\varepsilon})$ that improves to a $(3.582 + O(\varepsilon))$-approximation with $O(k + \frac{d}{\varepsilon^2} \log \frac{k}{\varepsilon})$ summary size when the objective is monotone. In the streaming setting we provide a $(9.294 + O(\varepsilon))$-approximation algorithm with summary size and memory $O(k + \frac{d}{\varepsilon^2} \log \frac{k}{\varepsilon})$; the approximation factor is then improved to $(5.582 + O(\varepsilon))$ in the monotone case.

Keywords: Submodular maximization, streaming algorithms, deletion robust, approximation algorithms, randomized algorithms

1. Introduction

Submodular maximization is a fundamental problem in machine learning that encompasses a broad range of applications, including active learning (Golovin and Krause, 2011) sparse reconstruction (Bach, 2010; Das and Kempe, 2011; Das et al., 2012), video analysis (Zheng et al., 2014), and data summarization (Lin and Bilmes, 2011; Bairi et al., 2015).
Given a submodular function f, a universe of elements V, and a family $\mathcal{F} \subseteq 2^V$ of feasible subsets of V, the optimization problem consists in finding a set $S \in \mathcal{F}$ that maximizes $f(S)$. A natural choice for $\mathcal{F}$ are capacity constraints (a.k.a. k-uniform matroid constraints) where any subset S of V of size at most k is feasible. Another standard restriction, which

generalizes capacity constraints and naturally comes up in various settings, is matroid constraints. For example, consider a movie recommendation application, where, given a large corpus of movies, we want to produce a set of recommended videos that contains at most one movie from each genre. This can be modeled as a partition matroid.

Exact submodular maximization is NP-hard, but efficient constant-factor approximation algorithms exist in both the centralized and in the streaming setting (e.g., Fisher et al., 1978; Călinescu et al., 2011; Chakrabarti and Kale, 2015; Feldman et al., 2018; Buchbinder and Feldman, 2024).

In this work, we design algorithms for submodular maximization over matroids that are robust to deletions. Primary motivations for considering deletions in the applications are privacy and user preferences, as users may exert their "right to be forgotten" or may update their preferences and thus exclude some of the data points. For instance, in the earlier movie recommendation example, a user may mark some of the recommended videos as "seen" or "inappropriate", and we may wish to quickly update the list of recommendations accordingly.

1.1 The Deletion Robust Approach

Following Mitrovic et al. (2017), we model robustness to deletion as a two-phase game against an adversary. In the first phase, the algorithm receives a robustness parameter d and chooses subset $W \subseteq V$ as a (possibly randomized) summary of the whole data set V, which can be given either offline or in streaming. Concurrently, an adversary selects a subset $D \subseteq V$ with $|D| \leq d$. In the second phase, the adversary reveals D, and the algorithm determines a feasible solution in the portion of the summary that has not been deleted. The goal of the algorithm is to extract a feasible subset of $W \setminus D$ that is competitive with the optimal solution in $V \setminus D$. Natural performance metrics in this model are the algorithm's approximation guarantee and its space complexity as measured by the size of the set W. We study an *oblivious* adversary, whose choice of the deleted elements D *does not* depend on the realized W, but possibly on the structure of the algorithm (not on its random bits). This is a natural assumption in the applications. Consider the movie recommendation example: the fact that a movie has already been watched by the user or that it is deemed inappropriate is independent of whether the specific film has been selected or not in the summary.

In this model, to obtain any constant-factor approximation, the summary size has to be $\Omega(k+d)$, even when f is additive and the constraint is a k-uniform matroid. To see this, consider the case where exactly $k+d$ the elements have unitary weight, and the remaining elements have weight zero. The adversary selects d of the valuable elements for deletion, but the algorithm does not know which. To be protected against any possible choice of the adversary, the best strategy of the algorithm is to choose W uniformly at random from the elements that carry weight. This leads to an expected weight of the surviving elements of k multiplied by $|W|/(k+d)$, while the optimum is k.

Prior work gave deletion robust algorithms for the special case of k-uniform matroids and monotone objectives. The state-of-the-art is a $2+O(\varepsilon)$ approximation with $O(k+\frac{d\log k}{\varepsilon^2})$ space in the centralized setting, while in the streaming setting the same approximation is achievable at the cost of an extra multiplicative factor of $O(\frac{\log k}{\varepsilon})$ in space complexity (Kazemi et al., 2018).

Reference	Objective	Model	Approximation	Summary size
Corollary 9 (this work)	Non-Monotone	Centralized	$4.494 + O(\varepsilon)$	$O\big(\frac{k+d}{\varepsilon^2}\log k\big)$
Corollary 10 (this work)	Monotone	Centralized	$3.582 + O(\varepsilon)$	$O\big(k + \frac{d}{\varepsilon^2}\log k\big)$
Zhang et al. (2022)	Monotone	Centralized	$3.582 + O(\varepsilon)$	$O\big(k + \frac{d}{\varepsilon}\log k\big)$
Mirzasoleiman et al. (2017)	Non-Monotone	Streaming	5.205	$\tilde{O}(kd)$
Theorem 12 (this work)	Non-Monotone	Streaming	$9.294 + O(\varepsilon)$	$O\big(k + \frac{d}{\varepsilon^2}\log k\big)$
Mirzasoleiman et al. (2017)	Monotone	Streaming	3.147	$\tilde{O}(kd)$
Corollary 15 (this work)	Monotone	Streaming	$5.582 + O(\varepsilon)$	$O\big(k + \frac{d}{\varepsilon^2}\log k\big)$
Zhang et al. (2022)	Monotone	Streaming	4	$O(k + d)$

Table 1: The table summarizes the results for deletion robust submodular maximization with matroid constraints. Note that the $\tilde{O}$ hides terms poly-logarithmic in the rank k. The approximation guarantees of Mirzasoleiman et al. (2017) are obtained via the state-of-the-art streaming algorithms for submodular maximization with matroid constraints (Buchbinder and Feldman (2019) and Feldman et al. (2022)). Zhang et al. (2022) is a follow-up paper.

For general matroids, Mirzasoleiman et al. (2017) give a black-box reduction that adapts (non-robust) submodular maximization algorithms to the deletion robust setting. Their approach yields a 3.147-approximation algorithm for monotone objectives using Feldman et al. (2022) and a 5.205 approximation for non-monotone objectives via Buchbinder and Feldman (2019). The downside of Mirzasoleiman et al. (2017) is that the space complexity is $\tilde{O}(kd)$.[1] The multiplicative factor d is inherent in the construction, and the k is needed by any streaming subroutine, so this approach necessarily leads to a space complexity of $\Omega(dk)$. The main open problem from their work is to design *space-efficient* algorithms for the centralized and streaming problems. These questions are particularly important in many practical scenarios where data sets are large and space is a crucial resource.

1.2 Our Results

We present the first constant-factor approximation algorithms for deletion robust submodular maximization subject to general matroid constraints with almost optimal space usage, i.e., our algorithms only use $\tilde{O}(k + d)$ space. More formally, in the centralized setting we present a $(4.494 + O(\varepsilon))$-approximation algorithm with summary size $O(\frac{k+d}{\varepsilon^2}\log\frac{k}{\varepsilon})$, that can be improved to a $(3.582+O(\varepsilon))$-approximation with summary size $O(k+\frac{d}{\varepsilon^2}\log\frac{k}{\varepsilon})$ if the objective is monotone. In the streaming setting we provide a $(9.294 + O(\varepsilon))$-approximation algorithm with summary size and memory $O(k+\frac{d}{\varepsilon^2}\log\frac{k}{\varepsilon})$, that becomes a $(5.582+O(\varepsilon))$-approximation if the objective is monotone. We remark that, besides the black box results by Mirzasoleiman et al. (2017) that have a far-from-optimal space usage, we are the first to offer a positive result for non-monotone submodular functions in the deletion robust setting, with *any* type of constraint. All our results are summarized and compared to the existing literature in Table 1.

For monotone objectives, the approximation factors in the two cases are $2 + \beta$ and $4 + \beta$, where β is $e/(e-1) \approx 1.582$, i.e., the best-possible approximation guarantee for the standard

1. Where the $\tilde{O}$ notation hides logarithmic factors.

centralized problem (Călinescu et al., 2011; Feige, 1998). For the non-monotone case, we still have a $2 + \beta$, where $\beta \approx 2.494$ is the state-of-the-art approximation guarantee for non-robust centralized submodular maximization subject to matroid constraint (Buchbinder and Feldman, 2024); our approximation for the streaming setting also depends on the routine used, but in a more intricate way. We mention that the best approximation factor for submodular maximization subject to matroid constraint is still an open problem: the optimal approximation factor lies between 2.494 (provided by the current best algorithm by Buchbinder and Feldman, 2024), and 2.092 (as per the impossibility result in Gharan and Vondrák, 2011). Thus, better algorithms may exist and would automatically improve our approximation guarantees.

We point out that the state-of-the-art approximation guarantees for (non-robust) submodular maximization with matroid constraint in the streaming model are 3.147-approximation in the monotone case and a 5.205-approximation in the general non-negative, i.e. possibly non-monotone, case (Feldman et al., 2022). The "price of robustness" is thus just an extra small additive term in the approximation factor, depending on the setting. At the same time, up to possibly a logarithmic factor, the memory requirements of all our results are tight.

1.3 Our Techniques

Intuitively, the extra difficulty in obtaining space-efficient robust deletion summaries with general matroid constraints—as opposed to cardinality constraints—is that the algorithm can only use elements respecting the matroid constraint to replace a good deleted element from a candidate solution. This phenomenon gets amplified when multiple elements need to be replaced. This contrasts sharply with the special case of k-uniform matroids, where all elements can replace any other element.

The Centralized Setting. Our algorithms start by setting a logarithmic number of value thresholds that span the average contribution of relevant optimum elements, and use these to group together elements with similar marginal values. The candidate solution is constructed using only elements from large enough bundles (at least a factor of $1/\varepsilon$ larger than the number of deletions). Random selection from a large bundle protects the value of the selected solution against adversarial deletions. In the centralized algorithm, it is possible to sweep through the thresholds in decreasing order. This monotonic behavior helps us design a charging mechanism for high-value elements dismissed due to the matroid constraint. We use the matroid structural properties to find an injective mapping from the optimum elements rejected by the matroid property to the set of selected elements. This allows us to claim that the marginal value of missed opportunities cannot dominate the values of added elements. Finally, since each fixed element of the base set has a very low probability of being added to the solution (due to the deletion robust sampling), we can handle *for free* the non-monotonicity of the objective in the analysis via a well-known sampling argument for submodular functions (Feige et al., 2007).

The Streaming Setting. In the streaming setting, elements arrive in an arbitrary order in terms of their membership to various bundles. We can still keep adding elements as long as a large enough bundle exists, but face the problem that elements from lower-value bundles

may prevent us from selecting some high-value elements due to the matroid constraint. So, when considering a new element, we allow for a swap operation with any of the elements in the solution to maintain feasibility with respect to the matroid constraint. We perform the swap if the marginal value of the new element e is substantially (a constant factor) higher than the marginal value that the element e' we are kicking out of the solution had when it was added to the solution. The constant factor gap between marginal values helps us account for not only e' but also the whole potential chain of elements that e' caused directly or indirectly to be removed during the algorithm. As we repeatedly swap elements in and out of the solution, it no longer holds that each fixed element of the base set has a very low probability of being, at some point, added to the solution. To tackle non-monotonicity, we thus embed a uniform sampling step (as in, e.g., Feldman et al., 2018; Amanatidis et al., 2021, 2022) in the procedure outlined above: any new element is discarded with some fixed probability before even being considered for swapping.

The Deletion Parameter. Finally, a consideration on the deletion robust parameter d. Our algorithms (like those from earlier work) do not need exact knowledge of d; an upper bound yields the same approximation at the cost of a larger summary. Also, with some extra work, it is possible to show that our approximation guarantees degrade gracefully when the estimate of d is too conservative and more elements are deleted. Note that some prior knowledge on d is a reasonable assumption in the applications. Without prior information about d, achieving good approximation and memory efficiency is impossible.

1.4 Related Work

Robust submodular optimization has been studied for more than a decade. In Krause et al. (2008), the authors study robustness from the perspective of multiple agents each with its own submodular valuation function; their objective is to select a subset that maximizes the minimum among all the agents' valuation functions. This objective could be seen as a max-min fair subset selection. Another robustness setting that deals with multiple valuation functions is distributionally robust submodular optimization (Staib et al., 2019) in which we can access samples from a distribution of valuation functions. These settings are fundamentally different from the robustness setting we study in our paper.

Robustness against few deletions. Orlin et al. (2018) and Bogunovic et al. (2017) investigate robustness in the presence of a few deletions. In their model, the algorithm needs to finalize the solution before the adversarial deletions are revealed and the adversary sees the choices of the algorithm. Therefore, having at least k deletions reduces the value of any solution to zero. This is the most prohibitive deletion robust setting we are aware of in the literature, and not surprisingly, the positive results of Orlin et al. (2018) and Bogunovic et al. (2017) are mostly useful when we are dealing with a few deletions.

Dynamic setting with d deletions. Mirzasoleiman et al. (2017) study submodular maximization and provide a general framework to empower insertion-only algorithms to process deletions on the fly as well as insertions. Their result works on general constraints, including matroids. As a result, they provide dynamic algorithms that process a stream of deletions and insertions with an extra multiplicative overhead of d (on both the computation time and memory footprint) compared to insertion-only algorithms. Their elegant idea

consists in running $d+1$ concurrent streaming submodular maximization algorithms where d is the maximum number of deletions. Every element is sent to the first algorithm. If it is not selected, it is sent to the second algorithm; if it is not selected again, it is sent to the third algorithm, and so on. With this trick, they maintain the invariant that the solution of one of these algorithms is untouched by the adversarial deletions, and therefore this set of $\tilde{O}(dk)$ elements suffice to achieve robust algorithms for matroid constraints. This approach has the drawback of having per update computation time linearly dependent on d (in the worst case, a new element can be processes by each one of the d algorithms), which can be prohibitive for a large number of deletions. It also has a total memory of $\tilde{O}(dk)$, which could be suboptimal compared to the lower bound of $\Omega(d+k)$ presented in the Introduction. The gap between this lower bound and the $\tilde{O}(dk)$ memory footprint in Mirzasoleiman et al. (2017) motivates the follow-up works on designing even more memory and computationally efficient algorithms.

Fully-dynamic setting. A closely related line of work investigates submodular maximization on stream of arbitrary insertions and deletions, where the goal is to maintain a good approximation to the dynamic optimum with per update computation time that depends poly-logarithmic in the total number of updates (length of the stream of updates). For cardinality constraint, Lattanzi et al. (2020) and Banihashem et al. (2023) provide a fully polynomial 2-approximation algorithm characterized by poly-logarithmic update time in both n and k, while Monemizadeh (2020) independently design an algorithm with similar approximation guarantee and an update time quadratic in the cardinality constraint with a smaller dependence on logarithmic terms. In subsequent work, Dütting et al. (2025) and Banihashem et al. (2024) tackle matroid constraints by designing constant factor approximation algorithms with update time that depends at most logarithmically in n. Note that these algorithms' much faster computation time comes at the cost of a potentially much larger memory footprint (up to the whole ground set), making them impractical in the deletion robust setting we study.

Two-phase model. Mitrovic et al. (2017) and Kazemi et al. (2018) are the first to study the deletion robust setting we consider in our work. They independently design submodular maximization algorithms for k-uniform matroids. Mitrovic et al. (2017) propose a streaming algorithm that achieves constant competitive ratio with memory $O((k+d)\,\mathrm{polylog}(k))$. Their results hold also against the adaptive adversary, that is aware of the summary set W before selecting the set D of deleted elements. On the other hand, Kazemi et al. (2018) design centralized and distributed algorithms with constant factor approximation and $O(k+d\log(k))$ memory, as well as a streaming algorithm with 2-approximation factor and memory footprint of $O(k\log(k)+d\log^2(k))$. We borrow some of their ideas, including bundling elements based on their marginal values and ensuring that elements are added to the solution before deletions only if they are selected uniformly at random from a large enough bundle (pool of candidates). A subsequent paper (Avdiukhin et al., 2019) shows how to obtain algorithms for the case of knapsack constraints with similar memory requirements and constant factor approximation. We aim to the generality of the matroid constraints studied in Mirzasoleiman et al. (2017) while maintaining the almost optimal computation time and space efficiency of Mitrovic et al. (2017) and Kazemi et al. (2018).

Sliding window. Another well-studied robustness model is the sliding window model. In this case, the deletions occur as the algorithm sweeps through the stream of elements; in that sense, they occur regularly rather than in an adversarial manner. Every element is deleted exactly W steps after its arrival. So, at every moment, the most recent W elements are present, and the objective is to select a subset of these present elements. Epasto et al. (2017) design a 3-approximation algorithm for the case of cardinality constraints with memory independent of the window size W. Zhao et al. (2019) provide algorithms that extend the sliding window algorithm to settings where elements have non-uniform lifespans and leave after arbitrary times.

Non-robust literature. Prior to the deletion robust models and motivated by large-scale applications, Mirzasoleiman et al. (2013) design the distributed greedy algorithm and shows it achieves provable guarantees for the cardinality constraint problem under some assumptions. In follow-up work, Mirrokni and Zadimoghaddam (2015) provide core-set frameworks that achieve constant factor approximation in the distributed setting. Barbosa et al. (2016) show how to approach the optimal $e/(e-1)$ approximation guarantee by increasing the round complexity of the distributed algorithm. For the case of matroid constraints, Ene et al. (2019) provide distributed algorithms that achieve the $e/(e-1)$ approximation with a poly-logarithmic number of distributed rounds. For the streaming setting, Feldman et al. (2022) provides a 3.147 competitive ratio algorithm with $\tilde{O}(k)$ memory with a matroid constraint of rank k.

Preliminary version and follow-up. A preliminary version of this work considering only monotone objectives has appeared as Dütting et al. (2022). In follow-up work, Zhang et al. (2022) design a simple algorithm for the streaming version of the deletion-robust submodular maximization problem with monotone objective subject to a p-matroid constraint. Their algorithm has optimal space usage of $O(k+d)$ and achieves a $4p$ approximation in the streaming case. They handle robustness via an elegant non-uniform sampling technique that avoids the extra $O(\log k)$ due to keeping the different buckets. For more details, see Table 1.

2. Problem formulation and preliminary results

We consider a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ on a ground set V of n elements. Given two sets $X, Y \subseteq V$, the *marginal gain* of X with respect to Y quantifies the change in value of adding X to Y and is defined as

$$f(X \mid Y) = f(X \cup Y) - f(Y).$$

When X consists of a singleton x, we use the shorthand $f(x \mid Y)$ instead of $f(\{x\} \mid Y)$. The function f is called *monotone* if $f(e \mid X) \geq 0$ for any set $X \subseteq V$ and element $e \in V$, and *submodular* if for any two sets X and Y such that $X \subseteq Y \subseteq V$ and any element $e \in V \setminus Y$ we have $f(e \mid X) \geq f(e \mid Y)$. Throughout the paper, we assume that f is given in terms of a value oracle that computes $f(S)$ for given $S \subseteq V$, and that f is *normalized*, i.e., $f(\emptyset) = 0$. We do not assume monotonicity of the function f when not specified otherwise. We slightly abuse the notation, and for a set X and an element e, use $X + e$ to denote $X \cup \{e\}$ and $X - e$ for $X \setminus \{e\}$.

Matroids. A non-empty family of sets $\mathcal{M} \subseteq 2^V$ is called a *matroid* if it satisfies the following properties:

- Downward-closure: if $A \subseteq B$ and $B \in \mathcal{M}$, then $A \in \mathcal{M}$;
- Augmentation: if $A, B \in \mathcal{M}$ with $|A| < |B|$, then $\exists\, e \in B$ such that $A + e \in \mathcal{M}$.

We call a set $A \subseteq 2^V$ *independent*, if $A \in \mathcal{M}$, and *dependent* otherwise. An independent set that is maximal with respect to inclusion is called a *base*; all the bases of a matroid share the same cardinality k, which is referred to as the *rank* of the matroid. Dually to basis, for any $A \notin \mathcal{M}$, a *circuit* $C(A)$ is defined as a minimal dependent subset of A, i.e., $C(A) \notin \mathcal{M}$ such that all its proper subsets are independent. The span or closure of a set A is defined as the set $\mathrm{cl}(A)$ of all the elements $e \in V$ such that $r_{\mathcal{M}}(A + e) = r_{\mathcal{M}}(A)$[2]. Given any matroid $\mathcal{M}$ and set S, it is possible to define two auxiliary matroids:

- the *contraction* of $\mathcal{M}$ by S, written $\mathcal{M}/S$, which is the matroid on $V \setminus S$ with rank function $r(T) = r_{\mathcal{M}}(T \cup S) - r_{\mathcal{M}}(S)$,
- the *restriction* of $\mathcal{M}$ to S, which is the matroid $\mathcal{M}'$ on S whose independent sets are the independent sets of $\mathcal{M}$ that are contained in S.

For a comprehensive overview of matroids and their properties, we refer to Schrijver (2003).

2.1 The Deletion Robust Model

The deletion robust model consists of two phases. The input of the first phase is the ground set V and a robustness parameter d, while the input of the second phase is an adversarial set of d deleted elements $D \subset V$, along with the outputs of the first phase. The goal is to construct a small size summary $W \subseteq V$ that is robust to deletions in the first phase, and a solution $S \subseteq W \setminus D$ that is independent with respect to matroid $\mathcal{M}$ in the second phase. The problem's difficulty lies in the fact that the summary W has to be robust against *any* possible choice of set D by an adversary oblivious to the randomness of the algorithm.
For any set of deleted elements D, the optimum solution denoted by $\mathrm{OPT}(V \setminus D)$ is

$$\mathrm{OPT}(V \setminus D) = \operatorname*{argmax}_{R \subseteq V \setminus D, R \in \mathcal{M}} f(R).$$

We say that a two-phase algorithm is an α-approximation for this problem if for all $D \subseteq V$ s.t. $|D| \leq d$, it holds that

$$f(\mathrm{OPT}(V \setminus D)) \leq \alpha \cdot \mathbb{E}\left[f(S_D)\right],$$

where S_D is the solution produced in Phase II when set D is deleted, and the expectation is with respect to the internal randomization of the algorithm. Besides the approximation factor α, an important feature of a two-phase algorithm is its summary size, i.e., the cardinality of the set W returned by the first phase.
In this paper, we also consider the streaming version of the problem, where the elements in V are presented in some arbitrary order (Phase I). At the end of the online phase, the algorithm outputs a deletion-robust summary W. Finally, the deleted set D is revealed and

2. We denote with $r_{\mathcal{M}}$ the rank function associated to matroid $\mathcal{M}$ (we omit the dependence on the matroid if clear from the context). The rank of a generic subset S of V is the cardinality of the largest independent set in S.

Phase II on $W \setminus D$ takes place offline. The quality of an algorithm for the streaming problem is not only assessed by its approximation guarantee and summary size but also in terms of its *memory*, i.e., the cardinality of the buffer of elements maintained in the online phase.

2.2 Preliminary Results

We conclude this section with three preliminary results: the folklore Sampling Lemma by Buchbinder et al. (2014), a Combinatorial Lemma derived from the well-known Hall's Marriage Theorem, and a graph theoretical result from Feldman et al. (2018). While the first lemma is our main tool to address non-monotonicity, the other two results are used for the charging arguments in the analysis of our algorithms.
We start with the Sampling Lemma, originally proved by Feige et al. (2011), and that we report as in Lemma 2.2 of Buchbinder et al. (2014). It states that the value of a random set cannot decrease much if no single element has a probability too large to be sampled.

Lemma 1 (Sampling Lemma) *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a (possibly not normalized) submodular set function, let $X \subseteq V$ and let $X(p)$ be a random subset, where each element of X appears with probability at most p (not necessarily independent). Then $\mathbb{E}[f(X(p))] \geq (1-p)f(\emptyset)$.*

We move now to our Combinatorial Lemma. At a high level, it is a general tool that maps injectively elements in a dependent set to those of an independent set under some conditions.

Lemma 2 (Combinatorial Lemma) *Consider a matroid $\mathcal{M} \subseteq 2^V$, set $F \subseteq V$, and set $G \in \mathcal{M}$. Suppose that for all $x \in G \setminus F$ there exist a set $F_x \subseteq F, F_x \in \mathcal{M}$ such that $F_x + x \notin \mathcal{M}$. Then there exists a mapping $h : G \setminus F \to F$ such that*

- *$h(x) \in F_x$ for all $x \in G \setminus F$*
- *$h(x) \neq h(y)$ for all $x, y \in G \setminus F$ with $x \neq y$.*

Intuitively, h as a *semi-matching* that matches all elements in $G \setminus F$ to an element in $\cup_{x \in G \setminus F} F_x$, while some of the elements in $\cup_{x \in G \setminus F} F_x \subseteq F$ can remain unmatched. To prove the lemma, we use the combinatorial version of Hall's Marriage Theorem (Hall, 1935), which concerns set systems and the existence of a transversal (a.k.a. system of distinct representatives). Formally, let $\mathcal{S}$ be a family of finite subsets of a base set X ($\mathcal{S}$ may contain the same set multiple times). A *transversal* is an injective function $f : \mathcal{S} \to X$ such that $f(S) \in S$ for every set $S \in \mathcal{S}$. In other words, f selects one representative from each set in S so that no two of these representatives are equal.

Theorem 3 (Hall's Theorem) *A family of sets $\mathcal{S}$ has a transversal if and only if $\mathcal{S}$ satisfies the marriage condition: for each subfamily $\mathcal{W} \subseteq \mathcal{S}$, it holds that $|\mathcal{W}| \leq \left|\bigcup_{F \in \mathcal{W}} F\right|$.*

We are ready to present the proof of our Combinatorial Lemma.
Proof of Lemma 2 Let W be any subset of $G \setminus F$ and define $F_W = \bigcup_{x \in W} F_x$. We want to show that $|F_W| = \left|\bigcup_{x \in W} F_x\right| \geq |W|$, this would conclude the proof by Theorem 3.
In order to do that we first show that $\mathrm{cl}(W \cup F_W) = \mathrm{cl}(F_W)$, where $\mathrm{cl}(.)$ denotes the *closure* (or *span*) of a set. We have that $W \subseteq \mathrm{cl}(F_W)$, in fact, for each element $x \in W$, there

Algorithm 1 Centralized Algorithm Phase I

Input: Precision ε and deletion parameter d
$V_d \leftarrow$ elements with the d largest values.
$V \leftarrow V \setminus V_d$, $A \leftarrow \emptyset$.
$\Delta \leftarrow \text{argmax}\{f(e) \mid e \in V\}$. ▷ Value of $(d+1)^{\text{st}}$ largest element
$T = \left\{(1+\varepsilon)^i \mid i \in \mathbb{Z},\ \varepsilon \cdot \frac{\Delta}{(1+\varepsilon)k} < (1+\varepsilon)^i \leq \Delta\right\}$
for $\tau \in T$ in decreasing order **do**
  $B_\tau \leftarrow \{e \in V \mid A + e \in \mathcal{M}, f(e \mid A) \geq \tau\}$
  while $|B_\tau| \geq \frac{k+d}{\varepsilon}$ **do**
    $e \leftarrow$ a random element sampled independently and uniformly from B_τ.
    $V \leftarrow V - e$, $A \leftarrow A + e$.
    $B_\tau \leftarrow \{e \in V \mid A + e \in \mathcal{M}, f(e \mid A) \geq \tau\}$
  Remove B_τ from V.
$B \leftarrow V_d \cup \bigcup_{\tau \in T} B_\tau$.
return A, B.

exists a subset $F_x \subseteq F_W$ such that $F_x + x \notin \mathcal{M}$ and therefore $x \in \text{cl}(F_x)$, which implies, by monotonicity of the closure with respect to the inclusion, that $x \in \text{cl}(F_W)$. We also know that $F_W \subseteq \text{cl}(F_W)$, hence $F_W \cup W \subseteq \text{cl}(F_W)$. If we apply the closure to both sets, we get

$$\text{cl}(F_W \cup W) \subseteq \text{cl}(\text{cl}(F_W)) = \text{cl}(F_W) \subseteq \text{cl}(F_W \cup W),$$

where the equality follows by noting that the closure of the closure is the closure itself. This shows that $\text{cl}(F_W)$ and $\text{cl}(F_W \cup W)$ coincide, as claimed.
Now let's look at the matroid $\mathcal{M}$ restriction to $\text{cl}(F_W \cup W) = \text{cl}(F_W)$. Afterward, contract this matroid by $(F \cap G) \cap \text{cl}(F_W)$ and call the resulting matroid with $\mathcal{M}'$. We claim that W is independent in this new matroid $\mathcal{M}'$. This is due to $W \subseteq G \setminus F$ and $W \cup (F \cap G) \subseteq G$ and $G \in \mathcal{M}$. If we call r' the rank of the matroid $\mathcal{M}'$, we have $r_{\mathcal{M}}(\text{cl}(F_W)) \geq r_{\mathcal{M}}(\text{cl}(F_W) \setminus (F \cap B)) = r' \geq |W|$. Finally, by the property of the rank function and of the closure, it holds that $|F_W| \geq r_{\mathcal{M}}(F_W) = r_{\mathcal{M}}(\text{cl}(F_W))$. Putting these two chains of inequalities together, we obtain $|F_W| \geq |W|$ as claimed. ■

The last preliminary result we present is Lemma 13 of Feldman et al. (2018). It concerns directed acyclic graphs (DAGs) where the nodes are also elements of a matroid. Under some assumptions, it guarantees the existence of an injective mapping between elements in an independent set and the sinks of the DAG. As a convention, we denote with $\delta^+(u)$ the out-neighborhood of any node u.

Lemma 4 *Consider an arbitrary directed acyclic graph $G = (V, E)$ whose vertices are elements of some matroid $\mathcal{M}$. If every non-sink vertex u of G is spanned by $\delta^+(u)$ in $\mathcal{M}$, then for every set S of vertices of G which is independent in $\mathcal{M}$ there must exist an injective function ψ such that, for every vertex $u \in S$, $\psi(u)$ is a sink of G which is reachable from u.*

3. Centralized Algorithm

In this section, we present a centralized algorithm for the Deletion Robust Submodular Maximization problem subject to a matroid constraint. We start by introducing some notations: $\tilde{e}$ denotes the $(d+1)$-th element with the largest value according to f, and Δ is its value.

Phase I. Our Phase-I algorithm takes in input a generic precision parameter ε, and uses it to define the set of relevant thresholds T:

$$T = \left\{ (1+\varepsilon)^i \mid i \in \mathbb{Z},\ \varepsilon \cdot \frac{\Delta}{(1+\varepsilon)k} < (1+\varepsilon)^i \leq \Delta \right\}.$$

The first phase of our algorithm constructs iteratively the summary W using two main sets: a candidate solution A and a reservoir B of good elements. The elements in the reservoir are selected according to their marginal value and are grouped into buckets of "similar" elements. The algorithm goes over the thresholds in T in decreasing order and updates sets A, B, and V (for loop in the pseudocode). Let τ be the threshold considered at some point, then B_τ contains any element $e \in V$ such that $f(e \mid A) \geq \tau$ and $A + e \in \mathcal{M}$ (lines 7 and 11). These high-contribution elements can be added to A while maintaining its independence. As long as the size of B_τ is larger than $(k+d)/\varepsilon$ (line 8), the algorithm chooses uniformly at random an element from B_τ (line 9), adds it to A (line 10) and recomputes B_τ (line 11). We observe that A is robust to deletions, i.e., when d elements are deleted, the probability of one specific element in A being deleted is intuitively at most ε. Moreover, since each element added to A is drawn from a pool of elements with *similar* marginals, the value of this set after the deletions decreases at most by a factor $(1-\varepsilon)$ in expectation (this is a very slack bound).
As soon as the cardinality of B_τ drops below $(k+d)/\varepsilon$, no more elements can be added directly from it while keeping A robust and feasible. Therefore, we remove these elements from V and save them for Phase II (line 12). During the execution of the algorithm, we need to take special care of the top d element with the highest f values. To avoid complications, we remove them from the instance before starting the procedure and add them to set B at the end. This does not affect the general logic and only simplifies the presentation and the proofs. The pseudocode of the centralized algorithm for Phase I is given in Algorithm 1.

Phase II. The summary W computed at the end of Phase I is composed by the union of A and B and is then passed to the second phase of our algorithm, which uses as routine an arbitrary algorithm Alg for centralized submodular maximization subject to matroid constraint. Alg takes in input a set of elements, function f and matroid $\mathcal{M}$ and returns a β-approximate solution (as a convention, we denote with $\textsc{Alg}(X)$ the solution output by the algorithm on input X). In this phase, we simply use Alg to compute a solution among all the elements in A and B that survived the deletion and return the best among the computed solution and the value of the surviving elements in A.

Theorem 5 *Consider deletion-robust submodular maximization with matroid constraints in the centralized setting. For any constant $\delta \in (0,1)$, the Centralized Algorithm (Algorithms 1 and 2) with precision parameter $\varepsilon \in (0, \delta)$ is in expectation a $(2+\beta+O(\varepsilon))$-approximation algorithm with summary size $O(\frac{k+d}{\varepsilon^2} \log \frac{k}{\varepsilon})$, where β is the approximation factor of* Alg.

Algorithm 2 Algorithm Phase II

1: **Input:** A and B from phase I, set D of deleted elements and optimization routine Alg
2: $A' \leftarrow A \setminus D$, $B' \leftarrow B \setminus D$
3: $\tilde{S} \leftarrow \textsc{Alg}(A' \cup B')$
4: **return** $S \leftarrow \text{argmax}\{f(A'), f(\tilde{S})\}$

Proof We start by analyzing the summary size, composed of the sets returned by Algorithm 1 at the end of phase I. Set A belongs to $\mathcal{M}$, so its size is no more than the rank of $\mathcal{M}$: $|A| \leq k$. Set B is the union of V_d (containing exactly d elements) and $\bigcup_{\tau \in T} B_\tau$. Each set B_τ has at most $\frac{k+d}{\varepsilon}$ element and there are at most $\frac{2}{\varepsilon} \log \frac{k}{\varepsilon}$ such sets (by design of T). Therefore

$$|A| + |B| \leq k + d + \tfrac{2(k+d)}{\varepsilon^2} \log \tfrac{k}{\varepsilon} \in O\left(\tfrac{k+d}{\varepsilon^2} \log \tfrac{k}{\varepsilon}\right).$$

We move to the approximation factor. We fix any set D with $|D| \leq d$ and bound the ratio between the expected value of $f(S)$ and $f(\text{OPT})$ (we omit the dependence on D since it is clear from the context). As a first step, we relate the value of OPT with that of $\text{OPT} \cup A$. This passage is immediately true when f is monotone but needs some work otherwise.

Lemma 6 *It holds that* $(1-\varepsilon) f(\text{OPT}) \leq \mathbb{E}\left[f(\text{OPT} \cup A)\right]$.

Proof of Lemma 6 Every time a new random element is added to the candidate solution A, it is drawn uniformly at random from a large pool of elements of cardinality at least $(k+d)/\varepsilon$. Denote with x_i the i^{th} element added to the solution by the algorithm, by the above consideration, it holds that $\mathbb{P}(x = x_i) \leq \varepsilon/(k+d)$ for any fixed $x \in V$ and $i = 1, \dots k$. By union bound, we have the following:

$$\mathbb{P}(x \in A) \leq \sum_{i=1}^{k} \mathbb{P}(x = x_i) \leq \varepsilon \frac{k}{k+d} \leq \varepsilon, \quad \forall x \in V. \tag{1}$$

Consider the submodular function $g : 2^V \to \mathbb{R}_{\geq 0}$, defined as $g(T) = f(\text{OPT} \cup T)$, for all $T \subseteq V$. Applying the Sampling Lemma (Theorem 1) on g, and using Equation (1), yields:

$$\mathbb{E}\left[f(\text{OPT} \cup A)\right] = \mathbb{E}\left[g(A)\right] \geq (1-\varepsilon) g(\emptyset) = (1-\varepsilon) f(\text{OPT}).$$

This concludes the proof of the Lemma. ■

We now partition $V \setminus D$ into subsets whose elements can be accounted for in a similar way. First, recall that we denoted by B' the subset of B surviving the deletion: $B' = B \setminus D$; it holds that $B \cap \text{OPT} \subseteq B'$. Then, define L as the set of elements of low value:

$$L = \left\{ e \in V \setminus D \mid f(e \mid A) \leq \varepsilon \cdot \tfrac{f(\text{OPT})}{k} \right\}.$$

Intuitively, L contains all the low-value elements: these elements do not increase the value of the submodular function considerably if added to A (so that we do not lose much by ignoring them). Finally, let F be the surviving elements not added to the summary because

of feasibility issues; formally, $F = V \setminus (A \cup B \cup L \cup D)$. We can now decompose the contributions of elements in $\mathrm{OPT} \cup A$ using the sets we introduced:

$$\begin{aligned}(1-\varepsilon)f(\mathrm{OPT}) \leq & \mathbb{E}\left[f(\mathrm{OPT} \cup A)\right] && \text{(by Lemma 6)}\\ \leq & \mathbb{E}\left[f(A)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap B' \mid A)\right] && \text{(by submodularity)}\\ & + \mathbb{E}\left[f(\mathrm{OPT} \cap F \mid A)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap L \mid A)\right] && (2)\end{aligned}$$

We study separately the terms in Equation (2). Recall that A' is the candidate solution once the deleted elements D are revealed; we use its expected value to bound the first term of the inequality. Intuitively, for any element in A, the probability of being deleted is $O(\varepsilon)$ since it has been sampled from ${}^{(d+k)}/_{\varepsilon}$ *similar* elements uniformly at random, and only d elements are deleted. We formalize this idea in the following lemma.

Lemma 7 *It holds that* $\mathbb{E}\left[f(A)\right] \leq \frac{1+\varepsilon}{1-\varepsilon}\, \mathbb{E}\left[f(A')\right]$.

Proof of Lemma 7 The proof is similar to that of Lemma 2 in Kazemi et al. (2017), but without relying on the monotonicity of f. For the sake of the analysis, for each threshold $\tau \in T$, let A_τ be the set of elements in A that were added during an iteration of the for loop corresponding to threshold τ in Algorithm 1. Moreover, let $n_\tau = |A_\tau|$ and sort the elements in $A_\tau = \{a_1^\tau, a_2^\tau, \ldots, a_{n_\tau}^\tau\}$ according to the order in which they are added to A_τ. We can decompose the value of A using the definition of marginal value and a telescopic argument:

$$f(A) = \sum_{\tau \in T} \sum_{\ell=1}^{n_\tau} f(a_\ell^\tau \mid \cup_{t>\tau} A_t \cup \{a_1^\tau, a_2^\tau, \ldots, a_{\ell-1}^\tau\}).$$

The elements added in a specific iteration of the for loop share the same marginal up to an $(1+\varepsilon)$ factor (because they all have marginal contribution larger than the threshold τ of that iteration but did not pass the test with the previous threshold $\tau \cdot (1+\varepsilon)$):

$$\tau \leq f(a_\ell^\tau \mid \cup_{t>\tau} A_t \cup \{a_1^\tau, a_2^\tau, \ldots, a_{\ell-1}^\tau\}) \leq \tau \cdot (1+\varepsilon), \quad \forall \tau \in T, \quad \forall \ell = 1, \ldots, n_\ell. \tag{3}$$

Thus, summing up these contributions, we have

$$\sum_{\tau \in T} |A_\tau| \cdot \tau \leq f(A) \leq (1+\varepsilon) \sum_{\tau \in T} |A_\tau| \cdot \tau. \tag{4}$$

We decompose the value of $A' = A \setminus D$ similarly. The crucial argument is that any a_ℓ^τ is drawn uniformly at random from a set of cardinality at least ${}^{(d+k)}/_{\varepsilon}$ where at most d elements lie in D, so

$$\mathbb{P}\left(a_\ell^\tau \notin D \mid \ell \leq n_\tau\right) \geq 1 - \varepsilon \tfrac{d}{k+d} \geq 1 - \varepsilon. \tag{5}$$

Note that the randomness here is with respect to the random draw of the algorithm in Phase I, as D is fixed but unknown to the algorithm. Let $A'_\tau = A_\tau \setminus D = A_\tau \cap A'$, we have

$$\begin{aligned}
f(A') &= \sum_{\tau\in T}\sum_{\ell=1}^{n_\tau} \mathbb{I}\left\{a_\ell^\tau \notin D\right\} f(a_\ell^\tau \mid \cup_{t>\tau} A'_t \cup (\{a_1^\tau,\dots,a_{\ell-1}^\tau\}\cap D)) \\
&\geq \sum_{\tau\in T}\sum_{\ell=1}^{n_\tau} \mathbb{I}\left\{a_\ell^\tau \notin D\right\} \cdot f(a_\ell^\tau \mid \cup_{t>\tau} A_t \cup \{a_1^\tau,\dots,a_{\ell-1}^\tau\}) && \text{(by submodularity)} \\
&\geq \sum_{\tau\in T}\sum_{\ell=1}^{n_\tau} \mathbb{I}\left\{a_\ell^\tau \notin D\right\} \cdot \tau && \text{(by Equation (3))} \\
&= \sum_{\tau\in T} |A'_\tau| \cdot \tau.
\end{aligned}$$

We apply the expected value to the previous inequality, which results in

$$\mathbb{E}\left[f(A')\right] \geq \sum_{\tau\in T} \mathbb{E}\left[|A'_\tau|\right] \cdot \tau \geq (1-\varepsilon) \sum_{\tau\in T} \mathbb{E}\left[|A_\tau|\right] \cdot \tau \geq \frac{1-\varepsilon}{1+\varepsilon} \mathbb{E}\left[f(A)\right],$$

while the last inequality follows from the right-hand side of Equation (4), the argument behind the second one is based on the linearity of expectation and the bound on the probability as in Equation (5):

$$\begin{aligned}
\mathbb{E}\left[|A'_\tau|\right] &= \mathbb{E}\left[\sum_{\ell=1}^{n_\tau} \mathbb{I}\left\{a_\ell^\tau \notin D\right\}\right] = \sum_{\ell=1}^{+\infty} \mathbb{E}\left[\mathbb{I}\left\{a_\ell^\tau \notin D\right\} \cdot \mathbb{I}\left\{|A_\tau| \geq t\right\}\right] \\
&= \sum_{\ell=1}^{+\infty} \mathbb{P}\left(a_\ell^\tau \notin D \middle| |A_\tau| \geq \ell\right) \cdot \mathbb{P}\left(|A_\tau| \geq t\right) \\
&\geq (1-\varepsilon) \sum_{\ell=1}^{+\infty} \mathbb{P}\left(|A_\tau| \geq \ell\right) && \text{(by Equation (5))} \\
&= (1-\varepsilon)\mathbb{E}\left[|A_\tau|\right]. && (6)
\end{aligned}$$

This concludes the proof of the Lemma. ■

For the second term of Equation (2), observe that $\mathrm{OPT} \cap B' \in \mathcal{M}$ and is contained in the set of elements passed to Alg, so its value is dominated by β times that of $\textsc{Alg}(A' \cup B')$, by the approximation guarantees of the routine Alg:

$$\mathbb{E}\left[f(\mathrm{OPT} \cap B' \mid A)\right] \leq \beta \cdot \mathbb{E}\left[f(S)\right]. \tag{7}$$

Recall that S is the final solution output by the algorithm. Bounding the third term is more involved and is handled in the following lemma via a charging argument that uses our Combinatorial Lemma (Lemma 2).

Lemma 8 *It holds that* $\mathbb{E}\left[f(\mathrm{OPT} \cap F \mid A)\right] \leq (1+\varepsilon)\mathbb{E}\left[f(A)\right]$.

Proof of Lemma 8 Sort the elements in $\mathrm{OPT} \cap F = \langle x_1, x_2, \dots \rangle$ according to the order in which they were removed from some B_τ because of feasibility constraint (since they are in $V \setminus (A \cup B \cup D \cup L)$ it must be the case). Once an element fails the feasibility test and is removed from some B_τ, it never becomes feasible again because only new elements are inserted in A during the execution of the first phase. Furthermore, for each such x_i, let F_i be the set A when x_i fails the feasibility test. We know that F_i is independent since A is maintained independent throughout the execution of Algorithm 1; moreover, by definition of F_i, $F_i + x_i \notin \mathcal{M}$. By Lemma 2, there exists an injective function $h : \mathrm{OPT} \cap F \to A$ such that $h(x_i) \in F_i$ for all i. Let A_i denote the set A when the element $h(x_i)$ gets added to it. Since $h(x_i)$ has been added to A before x_i, it holds that $A_i \subseteq F_i$. We have

$$\begin{aligned} f(\mathrm{OPT} \cap F \mid A) &\leq \sum_{x \in \mathrm{OPT} \cap F} f(x \mid A_i) && \text{(by submodularity and } A_i \subseteq A) \\ &\leq \sum_{x \in \mathrm{OPT} \cap F} (1+\varepsilon) \cdot f(h(x_i) \mid A_i) \\ &\leq (1+\varepsilon) \cdot f(A). \end{aligned}$$

The second inequality uses the fact that if x_i was still feasible to add when $h(x_i)$ was added, then their marginals are at most a $(1+\varepsilon)$ factor away. The last inequality follows from a telescopic decomposition of A, the fact that an element is added to the candidate solution A only if its marginal is positive (larger than a certain non-negative threshold), and the fact that h is injective. Applying the expectation to both extremes of the chain of inequalities concludes the proof of the lemma. ■

The fourth term of Equation (2) refers to at most k elements and can be bounded based on the definition of L: any element e in L is such that $f(e|A) \leq \varepsilon \cdot \frac{f(\mathrm{OPT})}{k}$, by submodularity we have then that

$$\mathbb{E}\left[f(\mathrm{OPT} \cap L \mid A)\right] \leq \mathbb{E}\left[\sum_{e \in \mathrm{OPT} \cap L} f(e \mid A)\right] \leq \varepsilon \cdot f(\mathrm{OPT}). \tag{8}$$

Finally, we compose the four bounds in Equation (2), and obtain the following:

$$\begin{aligned} &(1-\varepsilon) f(\mathrm{OPT}) \\ &\quad \leq \mathbb{E}\left[f(A)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap B' \mid A)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap F \mid A)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap L \mid A)\right] \\ &\quad \leq (2+\varepsilon)\,\mathbb{E}\left[f(A)\right] + \beta \mathbb{E}\left[f(S)\right] + \varepsilon f(\mathrm{OPT}) \quad \text{(by Equations (7, 8) and Lemma 8)} \\ &\quad \leq (2+\varepsilon)\frac{1+\varepsilon}{1-\varepsilon}\mathbb{E}\left[f(A')\right] + \beta \cdot \mathbb{E}\left[f(S)\right] + \varepsilon f(\mathrm{OPT}) \quad \text{(by Theorem 7)} \\ &\quad \leq \left[(2+\varepsilon)\frac{1+\varepsilon}{1-\varepsilon} + \beta\right] \cdot \mathbb{E}\left[f(S)\right] + \varepsilon \cdot f(\mathrm{OPT}) \quad \text{(by definition of } S) \end{aligned}$$

Rearranging terms, we get

$$\begin{aligned} f(\mathrm{OPT}) &\leq \frac{\mathbb{E}\left[f(\mathrm{OPT} \cup A)\right]}{1-2\varepsilon} \leq \left[\frac{(2+\varepsilon)\,(1+\varepsilon)}{(1-2\varepsilon)(1-\varepsilon)} + \frac{\beta}{1-2\varepsilon}\right] \cdot \mathbb{E}\left[f(S)\right] \\ &\leq \left[2 + \beta + O(\varepsilon)\right] \cdot \mathbb{E}\left[f(S)\right]. \end{aligned}$$

We conclude with a comment on what we mean by using $O(\varepsilon)$ here and throughout the paper: for any fixed constant β and fixed $\delta \in (0,1)$, there exists a constant C such that the $O(\varepsilon)$ term can be replaced with $C \cdot \varepsilon$, for any $\varepsilon \in (0, \delta)$. ■

Combining the previous theorem with some routines for non-monotone submodular maximization with matroid constraints, we get an approximation factor of ≈ 4.494 if we use the state-of-the-art algorithm as in Buchbinder and Feldman (2024), or of ≈ 4.73 if we use the less involved measured continuous greedy (Feldman et al., 2011). Formally, we have the following corollary.

Corollary 9 *Consider deletion-robust submodular maximization with matroid constraints in the centralized setting. For any constant $\delta \in (0,1)$, there exists a constant C_δ such that for any $\varepsilon \in (0, \delta)$, a $(4.494 + \varepsilon \cdot C_\delta)$-approximation algorithm with summary size $O(\frac{k+d}{\varepsilon^2} \log \frac{k}{\varepsilon})$ exists.*

Let's move our attention to monotone objectives. We get an improved approximation factor of 4 if we use as subroutine the greedy algorithm (Fisher et al., 1978) or $2 + \frac{e}{e-1} \leq 3.582$ if we use continuous greedy as in Călinescu et al. (2011). Furthermore, it is possible to slightly improve the bound on the summary size.

Corollary 10 *Consider deletion-robust submodular maximization with matroid constraints in the centralized setting. For any constant $\delta \in (0,1)$, there exists a constant C_δ such that for any $\varepsilon \in (0, \delta)$, a $(3.582 + \varepsilon \cdot C_\delta)$-approximation algorithm with summary size $O(k + \frac{d}{\varepsilon^2} \log \frac{k}{\varepsilon})$ exists.*

Proof First, we study the improved bound on the summary size. Since the function is monotone, we do not need to argue as in Lemma 6 to bound $f(\mathrm{OPT})$ with $f(\mathrm{OPT} \cup A)$; thus we can modify the condition on the while loop of Algorithm 1 to simply consider $|B_\tau| \geq d/\varepsilon$. With this tweak, there are still at most $\frac{1}{\varepsilon} \log \frac{k}{\varepsilon}$ buckets, each of cardinality at most d/ε, thus the summary size is $O(k + \frac{d}{\varepsilon^2} \log \frac{k}{\varepsilon})$. From the approximation point of view, Lemma 7 still holds, as the crucial property that $\mathbb{P}(I(\ell, \tau)) \geq 1 - \varepsilon$ is verified (we are sampling uniformly at random from a bucket of cardinality at least d/ε where there are at most d deleted elements). The rest of the analysis of Theorem 5 goes through without any other difference, and we get that

$$f(\mathrm{OPT}) \leq \mathbb{E}[f(\mathrm{OPT} \cup A)] \leq \left[(2+\varepsilon)\frac{1+\varepsilon}{1-\varepsilon} + \beta\right] \cdot \mathbb{E}[f(S)] + \varepsilon \cdot f(\mathrm{OPT})$$

We use as optimization routine Alg continuous greedy, therefore we can plug in $\beta = \frac{e}{e-1}$ and, rearranging the terms we obtain

$$\begin{aligned} f(\mathrm{OPT}) &\leq \left(\frac{e}{(e-1)(1-\varepsilon)} + \frac{(2+\varepsilon)(1+\varepsilon)}{(1-\varepsilon)^2}\right) \mathbb{E}[f(S)] \\ &\leq \Big[\frac{e}{e-1} + 2 + \underbrace{\frac{8e - 7 - \delta(2e-1)}{(e-1)(\delta-1)^2}}_{C_\delta} \cdot \varepsilon\Big] \mathbb{E}[f(S)], \end{aligned}$$

where the last inequality can be numerically verified and holds for any $\varepsilon \in (0, \delta)$. ■

4. Streaming Setting

In this section, we consider the streaming setting, wherer the elements of V arrive on a stream in the first phase, and the goal is to compute a small summary W using limited (online) memory. Our approach consists in carefully combining the swapping algorithm (Chakrabarti and Kale, 2015) with subsampling (Feldman et al., 2018) in a deletion robust fashion. Our algorithm maintains an independent candidate solution A and buckets B_τ that contain *small* reservoirs of elements with similar marginal contributions with respect to A. Beyond the candidate solution and the buckets, the algorithm keeps an extra buffer V_d containing the best d elements.

We start explaining the algorithm by defining the thresholds values. We do not have an a priori estimate of OPT in the streaming setting. Thus, we do not know the thresholds corresponding to *high quality* elements upfront. This issue is overcome by initially considering *all* the powers of $(1+\varepsilon)$ and progressively removing the ones too small with respect to Δ, i.e., the $(d+1)$-st largest value seen so far. Note that at any point of its execution, the algorithm actually maintains at most a small (finite) subset of these thresholds.

Every new arriving element e' is first used to update V_d (line 7 in the pseudocode): if its value is larger than the minimum in V_d, then we add it to V_d and pop out e, the smallest value element from V_d. If the value of the newly arrived element is smaller than the smallest value element in V_d, we keep the set V_d intact. For simplicity, we denote with e to be the newly arrived element e' in this case. In other words, element e is either the new element or the element that lost its place in V_d to the new element. In either case, we use element e in the next steps of the algorithm. The value of Δ is updated if $f(e) > \Delta$ (line 8) and all the buckets corresponding to thresholds that are too small are deleted, thus guaranteeing that only a logarithmic number of active buckets is kept (line 9). At this point, element e is put into the correct bucket B_τ, if such bucket still exists (line 11).

Now, new elements are repeatedly drawn uniformly at random from the buckets B_τ with size at least d/ε as long as no bucket contains more than d/ε elements (while loop line 12). These elements drawn from the buckets are added to A if and only if the subsampling is successful (line 13) and it is either feasible to add them directly or they can be *swapped* with a *less important* element in A.

We formalize this notion of "importance". Each element in the solution is associated with a weight, defined as its marginal value to A when it was first considered for addition to the solution (line 14). An element in A is swapped for a more promising one only if the new one has a larger weight by at least a $(1+\gamma)$ factor and it is sampled independently with some probability p, while maintaining A independent.

Every time A changes, the buckets B_τ are wholly updated to maintain the invariant that B_τ contains only elements whose marginal value with respect to the current solution A is within τ and $(1+\varepsilon)\tau$ (line 26). This property is crucial to ensure deletion robustness. Every bucket contains *similar* elements, i.e., whose marginal density with respect to the current solution is at most a multiplicative $(1+\varepsilon)$ factor away. When the stream terminates, the algorithm passes the deletion robust summary W (composed by the candidate solution A and B, containing V_d and the surviving buckets) to Algorithm 2. Note that both the centralized and the streaming pipelines share the same algorithm for the second phase.

Algorithm 3 Streaming Algorithm Phase I

Input: Precision ε, deletion parameter d, sampling probability p, swapping factor γ
$T \leftarrow \{(1+\varepsilon)^i \mid i \in \mathbb{Z}\}$
$A \leftarrow \emptyset$, $\Delta \leftarrow 0$, $\tau_{min} \leftarrow 0$, $B_\tau \leftarrow \emptyset$ for all $\tau \in T$
$L \leftarrow \emptyset, \Gamma \leftarrow \emptyset, K \leftarrow \emptyset, R \leftarrow \emptyset, F \leftarrow \emptyset$ ▷ Auxiliary sets for the analysis
$V_d \leftarrow$ the first d arriving elements
for every arriving element e' **do**
  Add e' to V_d and then pop element e with smallest value from V_d
  $\Delta \leftarrow \max\{f(e), \Delta\}$, $\tau_{min} \leftarrow \frac{\varepsilon}{1+\varepsilon} \cdot \frac{\Delta}{k}, T \leftarrow \{\tau \in T \mid \tau \geq \tau_{min}\}$
  Remove all B_τ s.t. $\tau \notin T$ ▷ The deleted elements are added to L
  if $\tau_{min} > f(e \mid A)$ **then** Discard e and **continue** ▷ $L \leftarrow L + e$
  Find the largest threshold $\tau \in T$ s.t. $f(e \mid A) \geq \tau$ and add e to B_τ
  while $\exists \tau \in T$ such that $|B_\tau| \geq d/\varepsilon$ **do**
    Remove one element g from B_τ independently and u.a.r. ▷ $\Gamma \leftarrow \Gamma + g$
    $w(g) \leftarrow f(g \mid A)$
    Draw X_g independently from a Bernoulli with parameter p
    if $A + g \in \mathcal{M}$ **then**
      if $X_g = 1$, **then** $A \leftarrow A + g$
      else Discard g ▷ $R \leftarrow R + g$
    else
      $k_g \leftarrow \text{argmin}\{w(k) \mid k \in C(A+g)\}$
      if $w(g) > (1+\gamma) \cdot w(k_g)$ **then**
        if $X_g = 1$, **then** $A \leftarrow A + g - k_g$ ▷ $K \leftarrow K + k_g$
        else Discard g ▷ $R \leftarrow R + g$
      else
        Discard g ▷ $F \leftarrow F + g$
    Update $\{B_\tau\}$ according to A ▷ The deleted elements are added to L
$B \leftarrow V_d \cup_{\tau \in T} B_\tau$
return A, B

The pseudocode of the streaming algorithm is presented in Algorithm 3. Besides the operations already mentioned, part of the pseudocode relates only to the analysis (sets L, Γ, K, R, and F are indeed only functional to the analysis and play no role in the algorithm). Before stating and proving our theorem, we describe some of the properties of the weight function w that governs the swapping.

Lemma 11 *Let w be the weight function, A the candidate solution, $A' = A \setminus D$, and let K be the set of all the elements that, at some point, were in A but were later swapped. Then, the following properties hold:*

(*i*) $\gamma \cdot w(K) \leq w(A) \leq f(A)$

(*ii*) $w(A') \leq f(A')$

(*iii*) $f(A \cup K) \leq w(A \cup K)$

Proof The proof of the (i) first inequality is similar to the one of Lemma 9 in Chakrabarti and Kale (2015). Crucially, the weight function w is linear, and once an element enters A, its weight is fixed forever as the marginal value it contributed when entering A. Moreover, the weight of all the elements added to A is strictly positive. During the run of the algorithm, every time an element k_g is removed from A, the weight of A increases by $w(g) - w(k_g)$ by its replacement with some element g. Moreover, $\gamma \cdot w(k_g) \leq w(g) - w(k_g)$ for every element $k_g \in K$ since $(1+\gamma)w(k_g) \leq w(g)$. Summing over all the elements in K, it holds that

$$w(K) = \sum_{k_g \in K} w(k_g) \leq \frac{1}{\gamma} \sum_{k_g \in K} [w(g) - w(k_g)] \leq \frac{1}{\gamma} w(A).$$

We show the part of the first point. Let $\{a_1, a_2, \dots a_\ell\}$ be the elements in A, sorted in the order in which they were added to A. We have that

$$f(A) = \sum_{i=1}^{\ell} f(a_i \mid \{a_1, \dots, a_{i-1}\}) \geq \sum_{i=1}^{\ell} f(a_i \mid A_{a_i}) = \sum_{i=1}^{\ell} w(a_i) = w(A),$$

where A_{a_i} is the solution set right before a_i is added to A. The second inequality (ii) follows from submodularity, because $\{a_1, \dots, a_{i-1}\} \subseteq A_{a_i}$ (A_{a_i} may also contains elements that are later removed from the solution). We have the following chain of inequalities

$$\begin{aligned}
f(A') &= \sum_{i=1}^{\ell} \mathbb{I}\{a_i \notin D\} \cdot f(a_i \mid \{a_1, \dots, a_{i-1}\} \cap D) && \text{(by a telescopic argument)} \\
&\geq \sum_{i=1}^{\ell} \mathbb{I}\{a_i \notin D\} \cdot f(a_i \mid \{a_1, \dots, a_{i-1}\}) && \text{(by submodularity)} \\
&\geq \sum_{i=1}^{\ell} \mathbb{I}\{a_i \notin D\} \cdot f(a_i \mid A_{a_i}) = w(A'). \\
& && \text{(by submodularity and } \{a_1, \dots, a_{i-1}\} \subseteq A_{a_i})
\end{aligned}$$

For the last (iii) inequality, let $\{x_1, x_2, \dots x_t\}$ be the elements in $A \cup K$, sorted by the order in which they were added to A. We have that

$$f(A \cup K) = \sum_{i=1}^{t} f(x_i \mid \{x_1, \dots, x_{i-1}\}) \leq \sum_{i=1}^{t} f(x_i \mid A_{x_i}) = \sum_{i=1}^{t} w(x_i) = w(A \cup K),$$

where A_{x_i} is the solution set right before x_i is added to A. The inequality follows from submodularity: A_{x_i} contains all the elements entered in A before x_i but not those that have already been removed. ■

We are finally ready for the main result of the Section. Instead of stating a general black box result as in Theorem 5, we consider directly the algorithm from Buchbinder and Feldman (2024) as Alg and analyze the relative algorithm for a suitable choice of p and γ. This is just for clarity of exposition, as the dependence on β of the generic approximation is more involved than in the centralized case.

Theorem 12 *Consider the problem of deletion robust monotone submodular maximization with matroid constraints in the streaming scenario. For any constant $\delta \in (0,1)$, there exists a constant C_δ such that for any $\varepsilon \in (0,\delta)$, a $(9.294 + C_\delta \cdot \varepsilon)$-approximation algorithm with summary and memory size $O(k + \frac{d}{\varepsilon^2}\log\frac{k}{\varepsilon})$ exists.*

Proof We prove that our streaming pipeline (Algorithm 3 and Algorithm 2) verifies the theorem for the right choice of parameters and optimization routine. We start by bounding the memory used by the algorithm. Sets A and V_d always contains at most k, respectively d, elements. Every time a new element of the stream is considered, all the active B_τ contain at most d/ε elements; furthermore, there are always at most $O(\frac{1}{\varepsilon}\log\frac{k}{\varepsilon})$ of them. The invariant ensures that the active thresholds are smaller than Δ (by submodularity) and larger than $\tau_{\min}$. Overall, the memory of the algorithm and the summary size is thus $O(k + \frac{d}{\varepsilon^2}\log\frac{k}{\varepsilon})$.
As in the analysis of Theorem 5, we fix any set D and study the relative expected performance of our algorithm. In the pseudocode, we have introduced some auxiliary sets that are useful in the analysis: the base set V is partitioned into the elements that are considered at some point in line 13 (contained in Γ), those in the buckets or V_d (in set B) and the ones with low marginals (in set L). We have all the ingredients to decompose $\mathrm{OPT} \cup A \cup K$ accordingly:

$$\begin{aligned}(1-p)f(\mathrm{OPT}) \leq& \mathbb{E}\left[f(\mathrm{OPT} \cup A \cup K)\right] \\ =& \mathbb{E}\left[f(A \cup K)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap L \mid A \cup K)\right] \\ &+ \mathbb{E}\left[f(\mathrm{OPT} \cap B \mid A \cup K)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap \Gamma \mid A \cup K)\right] \end{aligned} \tag{9}$$

Note that the first inequality follows from the uniform sampling step (line 13) and the Sampling Lemma (Lemma 1): each fixed element belongs to $A \cup K$ with a probability that is at most p.
The second and third terms of the decomposition are fairly easy to bound. If an element e is added to L at some point it means that its marginal contribution with respect to the current solution A is smaller than the current estimate τ_{min} of $\varepsilon/k f(\mathrm{OPT})$ (of which it is consistently a lower bound), by submodularity this implies that also its contribution with respect to the final $A \cup K$ is such that $f(e \mid A \cup K) \leq \varepsilon/k f(\mathrm{OPT})$, all in all:

$$f(\mathrm{OPT} \cap L \mid A \cup K) \leq \sum_{x \in \mathrm{OPT} \cap L} f(x \mid A \cup K) \leq \varepsilon \cdot f(\mathrm{OPT}). \tag{10}$$

We know that $\mathrm{OPT} \cap D = \emptyset$, this implies that $\mathrm{OPT} \cap B$ is contained in B' and thus in the summary W, which is passed to the second phase algorithm. Plus, note that $\mathrm{OPT} \cap B$ is independent because it is contained in OPT. So, if we use as Alg a routine for centralized submodular maximization with matroid constraint that guarantees a β approximation, we have:

$$\mathbb{E}\left[f(\mathrm{OPT} \cap B \mid A \cup K)\right] \leq \mathbb{E}\left[f(\mathrm{OPT} \cap B)\right] \leq \beta \cdot \mathbb{E}\left[f(S)\right]. \tag{11}$$

At this point, we get back to the decomposition in Equation (9) and bound the first term. We show that the value of all the elements that *at some point* were in the candidate solutions (corresponding to set $A \cup K$), is comparable with $f(A')$. This result overcomes two challenges: on the one hand, it shows that the total value of the elements swapped is upper bounded by $1/\gamma$ that of those that are not swapped; while on the other hand, that A is indeed robust to deletion.

Lemma 13 *It holds that*

$$\mathbb{E}\left[f(A \cup K)\right] \leq \frac{1+\varepsilon}{1-\varepsilon}\left(1+\frac{1}{\gamma}\right)\mathbb{E}\left[f(A')\right]. \tag{12}$$

Proof of Lemma 13 As a first step, we show that $w(A)$ is close to $f(A')$, at least in expectation with respect to the randomness of the algorithm. Let A_τ be the subset of elements that were added into A coming from B_τ and $A'_\tau = A_\tau \setminus D$. Moreover, let a_ℓ^τ be the ℓ^{th} element added to A_τ (if any), and denote with n_τ the random cardinality of $|A_\tau|$. The crucial observation is that when the algorithm decides to add an element to A from B_τ, then the probability that the element belongs to D is at most ε. Let A_a be the elements in A when a is added, so that $w(a) = f(a \mid A_a)$. We have

$$w(A) = \sum_{\tau \in T}\sum_{\ell=1}^{n_\tau} w(a_\ell^\tau), \text{ while } w(A') = \sum_{\tau \in T}\sum_{\ell=1}^{n_\tau} \mathbb{I}\left\{a \notin D\right\} w(a_\ell^\tau)$$

We know that, for any thresholds τ, the weight of any element a_ℓ^τ coming from B_τ respects

$$\tau \leq w(a_\ell^\tau) < (1+\varepsilon)\cdot\tau \tag{13}$$

We can then proceed similarly to the proof of Lemma 7:

$$\begin{aligned}
\mathbb{E}\left[w(A')\right] &= \sum_{\tau \in T}\mathbb{E}\left[\sum_{\ell=1}^{n_\tau}\mathbb{I}\left\{a \notin D\right\} w(a_\ell^\tau)\right] \\
&\geq \sum_{\tau \in T}\tau \cdot \mathbb{E}\left[|A'_\tau|\right] && \text{(by Equation (13))} \\
&\geq (1-\varepsilon)\cdot\sum_{\tau \in T}\tau \cdot \mathbb{E}\left[|A_\tau|\right] && \text{(by the same argument as in Equation (6))} \\
&\geq \frac{(1-\varepsilon)}{(1+\varepsilon)}\mathbb{E}\left[w(A)\right], && (14)
\end{aligned}$$

where, the last inequality follows from Equation (13). Now, let's move our attention to $w(A) + w(K)$. Using the properties of the weight function (Theorem 11), we have that

$$\begin{aligned}
\mathbb{E}\left[f(A \cup K)\right] &\leq \mathbb{E}\left[w(A \cup K)\right] && \text{(Property (iii))} \\
&= \mathbb{E}\left[w(A) + w(K)\right] && \text{(Linearity of } w\text{)} \\
&\leq \left(1+\frac{1}{\gamma}\right)\mathbb{E}\left[w(A)\right] && \text{(by Property (i))} \\
&\leq \left(1+\frac{1}{\gamma}\right)\frac{1+\varepsilon}{1-\varepsilon}\cdot\mathbb{E}\left[w(A')\right] && \text{(by Equation (14))} \\
&\leq \left(1+\frac{1}{\gamma}\right)\frac{1+\varepsilon}{1-\varepsilon}\cdot\mathbb{E}\left[f(A')\right] && \text{(by Property (ii))} \\
&\leq \left(1+\frac{1}{\gamma}\right)\frac{1+\varepsilon}{1-\varepsilon}\cdot\mathbb{E}\left[f(S)\right]. && \text{(By definition of } S\text{)}
\end{aligned}$$

This concludes the proof of the lemma. ■

We are left with bounding the marginal contribution of elements in $\mathrm{OPT} \cap \Gamma$ with respect to $A \cup K$. $\Gamma \setminus (A \cup K)$ is partitioned into two disjoint sets, F and R. Set F contains all the elements that failed the swapping test (line 21), while R all the elements that were removed because of sampling either in line 18 or 23.

Lemma 14 *Choosing $p = 1/(\gamma+2)$, it holds that*

$$\mathbb{E}\left[f(\mathrm{OPT} \cap \Gamma \mid A \cup K)\right] \leq (1+\gamma)\frac{1+\varepsilon}{1-\varepsilon} \cdot \mathbb{E}\left[f(S)\right]. \tag{15}$$

Proof of Lemma 14 Let g be any element in $\mathrm{OPT} \cap \Gamma$, and fix any compatible history $\mathcal{E}_g$ of the algorithm up to the point when g is considered in line 13 of Algorithm 3. The history $\mathcal{E}_g$ deterministically induces the candidate solution A_g when g is considered. Suppose g is not in F. In that case, it is possible to either add it directly or swap it with some $k_g \in A_g$ such that $f(g \mid A_g) = w(g) \geq (1+\gamma)w(k_g)$, then we have two possibilities: either g is successfully sampled and swapped with some element in the current candidate solution A_g, or g is added to R and discarded. Conditioning on $\mathcal{E}_g$, we have:

$$p \cdot \mathbb{E}\left[\mathbb{I}\left\{g \in R\right\} w(g) \mid \mathcal{E}_g\right] = p \cdot (1-p)\mathbb{E}\left[w(g) \mid \mathcal{E}_g\right] = (1-p) \cdot \mathbb{E}\left[\mathbb{I}\left\{g \in A \cup K\right\} w(g) \mid \mathcal{E}_g\right].$$

Summing over all such g and taking the expectation over all the compatible $\mathcal{E}_g$, we get:

$$p \cdot \mathbb{E}\left[w(\mathrm{OPT} \cap R)\right] = (1-p) \cdot \mathbb{E}\left[w(\mathrm{OPT} \cap (A \cup K))\right]. \tag{16}$$

We now move our attention to the elements in $\mathrm{OPT} \cap F$, i.e., the good elements in OPT that have been discarded because of the matroid constraint, and we fix any realization of the randomness of the algorithm (so that both F and R are fixed). As a first step, we show the existence of the injection $h : (\mathrm{OPT} \cap \Gamma) \setminus R \to A$ with the following properties:

(*a*) h restricted to $A \cap \mathrm{OPT}$ is the identity

(*b*) $w(g) \leq (1+\gamma)w(h(g))$ for all $g \in \mathrm{OPT} \cap F$

(*c*) $w(g) \leq w(h(g))$ for all $g \in \mathrm{OPT} \cap K$

We prove the existence of such h by constructing a suitable graph G where we apply Lemma 4. The nodes of G are the elements of Γ, while the edges are created iteratively as the algorithm considers elements in A. Let u be a generic element arriving in $\Gamma \setminus R$, and let A_u be the candidate solution at that point. If $A_u + u \in \mathcal{M}$, then no edge is created at that iteration; otherwise, a unique cycle $C(A_u + u)$ in $A_u + u$ containing u exists. Now, we have two cases: either u is swapped with some element x in $C(A_u + u) \cap A_u$, in which case we create directed edges from x to each element in $C(A_u + u) - x$, or u is rejected. In the latter case, we create directed edges from u to each element in $C(A_u + u) - u$. Note that, in both cases, the out-neighborhood of the node that gets discarded spans the element corresponding to the node itself. It is also clear that the graph G generated at the end of the procedure is acyclic, as an out-edge is only created when an element is discarded towards elements that have not been discarded yet. Thus, the graph respects the assumption of Lemma 4 and there exists an injective function h that associates every vertex/element $u \in (\mathrm{OPT} \cap \Gamma) \setminus R$

to a sink $h(u)$ that is reachable from u. From a different perspective, this h is a charging function that associates each element in $(\mathrm{OPT} \cap \Gamma) \setminus R$ with an element in the final solution A that *accounts* for it.
It is easy to see that h respects the three properties we want to enforce. For Property (a), function h restricted to $A \cap \mathrm{OPT}$ is the identity as elements in A are sinks. For the other two properties, consider any element $g \in \mathrm{OPT} \cap F$. Element g has been discarded without being added to the solution, meaning that the outgoing edges from g verify $w(g) \leq (1+\gamma)w(u)$, for all $u \in \delta^+(g)$. If we focus on the unique path from g to $h(g)$, we see that after the first step, when the weight decreases by at most a $(1+\gamma)$ factor, all the following edges do not decrease the weight, this is because an outgoing edge from an element of the solution either points to an element it is swapped with (whose weight is larger by at least a $(1+\gamma)$ factor) or to another element in the solution but with larger weight (as we swap the element with smallest weight in the cycle). This settles Property (b). To see why Property (c) holds, note that the same argument implies that, for all the elements already in the solution that are later swapped ($\mathrm{OPT} \cap K$), the weight does not decrease along the path to the sink.
We can use the properties of the injective function h to bound $w(\mathrm{OPT} \cap F)$.

$$\begin{aligned}
\mathbb{E}[w(\mathrm{OPT} \cap F)] &\leq (1+\gamma)\mathbb{E}\left[w(h(\mathrm{OPT} \cap F))\right] && \text{(by Property } (b)\text{)} \\
&\leq (1+\gamma)\mathbb{E}\left[w(h(\mathrm{OPT} \cap F)) + w(h(\mathrm{OPT} \cap K)) - w(\mathrm{OPT} \cap K))\right] && \text{(by Property } (c)\text{)} \\
&= (1+\gamma)\mathbb{E}\left[w(h(\mathrm{OPT} \cap (F \cup K)) - w(\mathrm{OPT} \cap K) + w(\mathrm{OPT} \cap A) - w(\mathrm{OPT} \cap A)\right] \\
&= (1+\gamma)\mathbb{E}\left[w(h((\mathrm{OPT} \cap \Gamma) \setminus R)) - w(\mathrm{OPT} \cap (A \cup K)))\right] && \text{(by Property } (a)\text{)} \\
&\leq (1+\gamma)\mathbb{E}\left[w(A)\right] - (1+\gamma)(w(\mathrm{OPT} \cap (A \cup K))) && (17)
\end{aligned}$$

Observe that in the last inequality we use that $w(a) \geq 0$ for all $a \in A$, as $h((\mathrm{OPT} \cap \Gamma) \setminus R)$ may be a strict subset of A. We can finally proceed to prove the lemma.

$$\begin{aligned}
&\mathbb{E}[f(\mathrm{OPT} \cap \Gamma \mid A \cup K)] \\
&\leq \mathbb{E}\left[\sum_{x \in \mathrm{OPT} \cap F} f(x \mid A \cup K)\right] + \mathbb{E}\left[\sum_{x \in \mathrm{OPT} \cap R} f(x \mid A \cup K)\right] && \text{(by submodularity)} \\
&\leq \mathbb{E}\left[w(\mathrm{OPT} \cap F)\right] + \mathbb{E}\left[w(\mathrm{OPT} \cap R)\right] && \text{(by submodularity)} \\
&\leq (1+\gamma)\mathbb{E}\left[w(A)\right] + \left(\tfrac{1-p}{p} - 1 - \gamma\right)\mathbb{E}\left[w(\mathrm{OPT} \cap (A \cup K))\right] && \text{(by Equations (16) and (17))} \\
&= (1+\gamma)\mathbb{E}\left[w(A)\right] && \text{(by choice of } p\text{)} \\
&\leq (1+\gamma)\tfrac{1+\varepsilon}{1-\varepsilon} \cdot \mathbb{E}\left[f(S)\right]. && \text{(as in Lemma 13)}
\end{aligned}$$

Note that the second inequality follows from submodularity and the fact that the candidate solution when an element is considered (and thus used to compute its weight) is always contained in $A \cup K$. ■

Plugging Equations (10) to (12) and (15) into Equation (9) we obtain the following:

$$\begin{aligned}
\left(1 - \frac{1}{2+\gamma}\right) f(\mathrm{OPT}) \leq{}& \mathbb{E}\left[f(A \cup K)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap L \mid A \cup K)\right] \\
&+ \mathbb{E}\left[f(\mathrm{OPT} \cap B \mid A \cup K)\right] + \mathbb{E}\left[f(\mathrm{OPT} \cap \Gamma \mid A \cup K)\right] \\
\leq{}& \left[\left(2 + \gamma + \frac{1}{\gamma}\right)\frac{1+\varepsilon}{1-\varepsilon} + \beta\right] \cdot \mathbb{E}\left[f(S)\right] + \varepsilon \cdot f(\mathrm{OPT})
\end{aligned}$$

Rearranging terms, one gets

$$\begin{aligned} f(\mathrm{OPT}) &\leq \left(1-\varepsilon-\frac{1}{2+\gamma}\right)^{-1}\left[\left(2+\gamma+\frac{1}{\gamma}\right)\frac{1+\varepsilon}{1-\varepsilon}+\beta\right]\cdot\mathbb{E}\left[f(S)\right] \\ &= \frac{(2+\gamma)[\gamma^2+(\beta+2)\gamma+1+\varepsilon(\gamma^2+(2-\beta)\gamma+1)]}{\gamma(1-\varepsilon)[1+\gamma-\varepsilon(\gamma+2)]}\cdot\mathbb{E}\left[f(S)\right] \\ &\leq \frac{(2+\gamma)[\gamma^2+(\beta+2)\gamma+1]}{\gamma[1+\gamma-\varepsilon(\gamma+2)]}\frac{1+\varepsilon}{1-\varepsilon}\cdot\mathbb{E}\left[f(S)\right] \end{aligned}$$

The theorem follows by using as optimization routine the state-of-the-art algorithm for centralized (non-monotone) submodular maximization with matroid constraint ($\beta \approx 2.494$ as in Buchbinder and Feldman (2024)) and then choosing γ (and accordingly p) as to minimize the multiplicative factor. In particular, if we set $\gamma \approx 1.732$, then for any fixed δ there exists a constant G_δ such that we get an approximation of the form $(9.294+C_\delta\cdot\varepsilon)$ for all ε in $(0,\delta)$. ■

As a corollary of the previous theorem, we get an improved approximation for monotone objectives. If the objective function is monotone, there is no need for the sampling procedure (as the value of OPT is trivially upper bounded by that of $\mathrm{OPT}\cup A\cup K$). Moreover, we can use the more effective greedy (Fisher et al., 1978) or continuous greedy (Călinescu et al., 2011) subroutine as Alg.

Corollary 15 *Consider the problem of deletion robust monotone submodular maximization with matroid constraints in the streaming setting. For any constant $\delta \in (0,1)$, there exists a constant C_δ such that for any $\varepsilon \in (0,\delta)$, a $(5.582+\varepsilon\cdot C_\delta)$-approximation algorithm with summary size and memory $O(k+\frac{d}{\varepsilon^2}\log\frac{k}{\varepsilon})$ exists.*

Proof The proof is similar to the one of Theorem 12. The only difference is that we do not need to subsample as $f(\mathrm{OPT})$ is smaller than $f(\mathrm{OPT}\cup A\cup K)$ by monotonicity. Thus, we set $p=1$ and $\gamma=1$ and then Equation (9) becomes:

$$\begin{aligned} f(\mathrm{OPT}) \leq& \mathbb{E}\left[f(\mathrm{OPT}\cup A\cup K)\right] \\ =& \mathbb{E}\left[f(A\cup K)\right]+\mathbb{E}\left[f(\mathrm{OPT}\cap L \mid A\cup K)\right] \\ &+\mathbb{E}\left[f(\mathrm{OPT}\cap B \mid A\cup K)\right]+\mathbb{E}\left[f(\mathrm{OPT}\cap\Gamma \mid A\cup K)\right] \\ \leq& 2\frac{1+\varepsilon}{1-\varepsilon}\mathbb{E}\left[f(S)\right]+\varepsilon f(\mathrm{OPT})+\beta\cdot\mathbb{E}\left[f(S)\right]+\mathbb{E}\left[f(\mathrm{OPT}\cap\Gamma \mid A\cup K)\right] \end{aligned}$$

For the last term, we use a more general statement. For all $X\subseteq\Gamma$, $X\in\mathcal{M}$, then it holds that $w(X)\leq 2\cdot w(A)$. This is a direct application of Theorem 1 of Varadaraja (2011) (using a single matroid and setting $r=2$). More specifically, we can imagine restricting the stream to consider only the elements in Γ, with the order in which they are considered in line 13. Choosing $X=\mathrm{OPT}\cap\Gamma$, we then get that

$$\mathbb{E}\left[f(\mathrm{OPT}\cap\Gamma \mid A\cup K)\right]\leq\mathbb{E}\left[w(\mathrm{OPT}\cap\Gamma)\right]\leq 2\mathbb{E}\left[w(A)\right].$$

We use as optimization routine Alg continuous greedy, therefore we can plug in $\beta = \frac{e}{e-1}$ and, rearranging the terms we obtain

$$\begin{aligned} f(\mathrm{OPT}) &\leq \left(2\frac{1+\varepsilon}{(1-\varepsilon)^2} + \frac{3e-2}{(1-\varepsilon)(e-1)}\right)\mathbb{E}\left[f(S)\right] \\ &\leq \Big[\frac{e}{e-1} + 4 + \underbrace{\frac{9e-8-\delta(5e-4)}{(e-1)(1-\delta)^2}}_{C_\delta}\cdot\varepsilon\Big]\mathbb{E}\left[f(S)\right]. \end{aligned}$$

The last inequality can be numerically verified and holds for any $\varepsilon \in (0, \delta)$. ■

5. Conclusion

We present the first space-efficient constant-factor approximation algorithms for deletion-robust submodular maximization over matroids in both the centralized and the streaming settings. In particular, we are also the first to design space-efficient deletion robust algorithms for non-monotone objectives with *any* type of constraints.

Acknowledgments

Federico's work was supported by the FAIR project (funded by theNextGenerationEU program within the PNRR-PE-AI scheme. M4C2, investment 1.3, line on Artificial Intelligence), the ERC Advanced Grant 788893 AMDROMA and PNRR MUR project IR0000013-SoBigData.it. Part of this work was done while Federico was an intern at Google Research, hosted by Paul Dütting.

References

Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Rebecca Reiffenhäuser. Submodular maximization subject to a knapsack constraint: Combinatorial algorithms with near-optimal adaptive complexity. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 231–242. PMLR, 2021.

Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, and Rebecca Reiffenhäuser. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. *J. Artif. Intell. Res.*, 74:661–690, 2022.

Dmitrii Avdiukhin, Slobodan Mitrovic, Grigory Yaroslavtsev, and Samson Zhou. Adversarially robust submodular maximization under knapsack constraints. In *KDD*, pages 148–156, 2019.

Francis R. Bach. Structured sparsity-inducing norms through submodular functions. In *NIPS*, pages 118–126, 2010.

Ramakrishna Bairi, Rishabh K. Iyer, Ganesh Ramakrishnan, and Jeff A. Bilmes. Summarization of multi-document topic hierarchies using submodular mixtures. In *ACL*, pages 553–563, 2015.

Kiarash Banihashem, Leyla Biabani, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Dynamic constrained submodular optimization with polylogarithmic update time. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 1660–1691. PMLR, 2023.

Kiarash Banihashem, Leyla Biabani, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. Dynamic algorithms for matroid submodular maximization. In *SODA*, pages 3485–3533. SIAM, 2024.

Rafael da Ponte Barbosa, Alina Ene, Huy L Nguyen, and Justin Ward. A new framework for distributed submodular maximization. In *FOCS*, pages 645–654, 2016.

Ilija Bogunovic, Slobodan Mitrovic, Jonathan Scarlett, and Volkan Cevher. Robust submodular maximization: A non-uniform partitioning approach. In *ICML*, pages 508–516, 2017.

Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Math. Oper. Res.*, 44(3):988–1005, 2019.

Niv Buchbinder and Moran Feldman. Constrained submodular maximization via new bounds for dr-submodular functions. In *STOC*, pages 1820–1831. ACM, 2024.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, pages 1433–1452. SIAM, 2014.

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.

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

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

Abhimanyu Das, Anirban Dasgupta, and Ravi Kumar. Selecting diverse features via spectral regularization. In *NIPS*, pages 1592–1600, 2012.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. Deletion robust submodular maximization over matroids. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 5671–5693. PMLR, 2022.

Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. Fully dynamic submodular maximization over matroids. *ACM Trans. Algorithms*, 21(1):11:1–11:23, 2025.

Alina Ene, Huy L Nguyen, and Adrian Vladu. Submodular maximization with matroid and packing constraints in parallel. In *STOC*, pages 90–101, 2019.

Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Submodular optimization over sliding windows. In *WWW*, pages 421–430, 2017.

Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *FOCS*, pages 461–471. IEEE Computer Society, 2007.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, 2011.

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

Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *NeurIPS*, pages 730–740, 2018.

Moran Feldman, Paul Liu, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. Streaming submodular maximization under matroid constraints. In *ICALP*, volume 229 of *LIPIcs*, pages 59:1–59:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions—ii. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

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

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 42:427–486, 2011.

P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10 (1):26–30, 1935.

Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Deletion-robust submodular maximization at scale. *CoRR*, abs/1711.07112, 2017.

Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In *ICML*, volume 80, pages 2549–2558, 2018.

Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *J. Mach. Learn. Res.*, 9(12), 2008.

Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Jakub Tarnawski, and Morteza Zadimoghaddam. Fully dynamic algorithm for constrained submodular optimization. In *NeurIPS*, 2020.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *HLT*, pages 510–520, 2011.

Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *STOC*, pages 153–162, 2015.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*, pages 2049–2057, 2013.

Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Deletion-robust submodular maximization: Data summarization with "the Right to be Forgotten". In *ICML*, pages 2449–2458, 2017.

Slobodan Mitrovic, Ilija Bogunovic, Ashkan Norouzi-Fard, Jakub Tarnawski, and Volkan Cevher. Streaming robust submodular maximization: A partitioned thresholding approach. In *NIPS*, pages 4557–4566, 2017.

Morteza Monemizadeh. Dynamic submodular maximization. In *NeurIPS*, 2020.

James B Orlin, Andreas S Schulz, and Rajan Udwani. Robust monotone submodular function maximization. *Math. Program.*, 172(1):505–537, 2018.

Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

Matthew Staib, Bryan Wilder, and Stefanie Jegelka. Distributionally robust submodular maximization. In *AISTATS*, pages 506–516, 2019.

Ashwinkumar Badanidiyuru Varadaraja. Buyback problem - approximate matroid intersection with cancellation costs. In *ICALP*, volume 6755, pages 379–390, 2011.

Guangyi Zhang, Nikolaj Tatti, and Aristides Gionis. Coresets remembered and items forgotten: submodular maximization with deletions. In *ICDM*, pages 676–685. IEEE, 2022.

Junzhou Zhao, Shuo Shang, Pinghui Wang, John C. S. Lui, and Xiangliang Zhang. Submodular optimization over streams with inhomogeneous decays. In *AAAI*, pages 5861–5868, 2019.

Jingjing Zheng, Zhuolin Jiang, Rama Chellappa, and P. Jonathon Phillips. Submodular attribute selection for action recognition in video. In *NIPS*, pages 1341–1349, 2014.