Coresets remembered and items forgotten: submodular maximization with deletions

Guangyi Zhang
KTH Royal Institute of Technology
Division of Theoretical Computer Science
Sweden
guaz@kth.se

Nikolaj Tatti
HIIT, University of Helsinki
Department of Computer Science
Finland
nikolaj.tatti@helsinki.fi

Aristides Gionis
KTH Royal Institute of Technology
Division of Theoretical Computer Science
Sweden
argioni@kth.se

Abstract—In recent years we have witnessed an increase on
the development of methods for submodular optimization, which have been motivated by the wide applicability of submodular functions in real-world data-science problems. In this paper, we contribute to this line of work by considering the problem of robust submodular maximization against unexpected deletions, which may occur due to privacy issues or user preferences. Specifically, we consider the minimum number of items an algorithm has to remember, in order to achieve a non-trivial approximation guarantee against adversarial deletion of up to \( d \) items. We refer to the set of items that an algorithm has to keep before adversarial deletions as a deletion-robust coreset.

Our theoretical contributions are two-fold. First, we propose a single-pass streaming algorithm that yields a \((1 - 2\epsilon)/(4p)\)-approximation for maximizing a non-decreasing submodular function under a general \( p \)-matroid constraint and requires a coreset of size \( k + d/\epsilon \), where \( k \) is the maximum size of a feasible solution. To the best of our knowledge, this is the first work to achieve an (asymptotically) optimal coreset, as no constant-factor approximation is possible with a coreset of size sublinear in \( d \).

Second, we devise an effective offline algorithm that guarantees stronger approximation ratios with a coreset of size \( O(d \log (k)/\epsilon) \). We also demonstrate the superior empirical performance of the proposed algorithms in real-life applications.

Index Terms—robust optimization, submodular maximization, streaming algorithms, approximation algorithms

I. INTRODUCTION

Submodular maximization has attracted much attention in the data-science community in recent years. Its popularity is due to the ubiquity of the “diminishing-returns” property in different problem settings and the rich toolbox that has been developed during the past decades [1]. The problem of maximizing a non-decreasing submodular function can be used to cast a wide range of applications, including viral marketing in social networks [2], data subset selection [3], and document summarization [4].

However, in a world full of uncertainty, a pre-computed high-quality solution may cease to be feasible due to unexpected deletions.

As an example, consider an application in movie recommendation, where we ask to select a subset of movies to recommend to a user so as to maximize a certain non-decreasing submodular utility function. It is possible that the user has already seen some or all movies in the recommended set, and the rest are insufficient for providing a good-quality recommendation. Such unexpected deletions may happen in other scenarios, for example, a user may exercise their “right to be forgotten”, specified by EU’s General Data Protection Regulation (GDPR) [5], and request at any point certain data to be removed.

Instead of re-running the recommendation algorithm over the dataset excluding the deleted items, which typically is time-consuming, a better way to handle unexpected deletions is to extract a small and robust coreset of the dataset, from which we can quickly select a new solution set. To quantify robustness, we require the coreset to be robust against adversarial deletions up to \( d \) items. Naturally, the coreset size has to depend on the number of deletions \( d \).

Different adversarial models have been studied in the literature. For the most powerful adversary, called an adaptive adversary, we assume that the coreset is known to the adversary and deletions occur in a worst-case manner. Such malicious deletions lead to weak quality guarantees [6], or require a larger coreset and longer running time [7].

In many applications, though, deletions are typically more benign and may only mildly corrupt the coreset. For example, in the movie-recommendation scenario, the probability that a user has watched a movie may depend on the popularity of the movie, and be independent on whether the movie has been added in a coreset. An adversary who is oblivious to the contents of the coreset is called a static adversary.

In this paper, we derive bounds for the smallest possible coreset size that suffices to provide a non-trivial quality guarantee against item deletions by a static adversary. It is known that no constant-factor approximation is possible with a coreset whose size is sublinear in \( d \) [8]. We assume that the adversary has unlimited computational power, and knows everything about the algorithm except for its random bits.

More concretely, given a non-decreasing submodular function, we study the problem Robust Coreset for submodular maximization under Cardinality constraint (RCC) against a static adversary. In addition, we study the generalization of RCC over a more general \( p \)-matroid constraint (RC\( p \)M), or a \( p \)-system constraint (RC\( p \)). Recall that a cardinality constraint is known as a uniform matroid. Throughout the paper, we are mostly interested in the more challenging case where \( k = O(d) \) and \( k \) is the maximum size of a feasible solution.
TABLE I: A summary of existing results on robust coresets. For simplicity, it is assumed that $k = O(d)$.

| Adversarial model | Constraint type | Approximation factor | Coreset size | Streaming/Offline |
|-------------------|-----------------|----------------------|--------------|-------------------|
| Mitrović et al. [6] | adaptive cardinality | $\approx 0.149$ | $O(d \log^3(k)/\epsilon)$ | S |
| Mirzasoleiman et al. [7]; Badanidiyuru et al. [9] | adaptive cardinality | $(1 - \epsilon)/2$ | $O(dk/\epsilon)$ | S |
| Mirzasoleiman et al. [7]; Chakrabarti and Kale [10] | adaptive $p$-matroids | $1/(4p)$ | $O(dk)$ | S |
| Kazemi et al. [11] | static cardinality | $(1 - \epsilon)/2$ | $O(d \log^2(k)/\epsilon^2)$ | S |
| This paper | static cardinality | $(1 - \epsilon)/2$ | $O(d \log^2(k)/\epsilon^2)$ | S |
| Dütting et al. [8] | static matroid | $\frac{1}{3} \epsilon^{-1} + O(1)$ | $O(d \log(k)/\epsilon^2)$ | S |
| This paper | static $p$-matroids | $(1 - 2\epsilon)/(4p)$ | $k + d/\epsilon$ | S |
| Feldman et al. [12] | adaptive cardinality | $0.514$ | $O(dk)$ | O |
| Kazemi et al. [11] | static cardinality | $(1 - \epsilon)/2$ | $O(d \log(k)/\epsilon^2)$ | O |
| This paper | static cardinality | $(1 - \epsilon)/2$ | $O(d \log(k)/\epsilon^2)$ | O |
| Dütting et al. [8] | static matroid | $\frac{1}{3} \epsilon^{-1} + 2 + O(1)$ | $O(d \log(k)/\epsilon^2)$ | O |
| This paper | static matroid | $\frac{1}{3} \epsilon^{-1} + 2 + O(1)$ | $O(d \log(k)/\epsilon^2)$ | O |
| This paper | static $p$-system | $\frac{1}{3} \epsilon^{-1} + 2 + O(1)$ | $O(d \log(k)/\epsilon^2)$ | O |

Our contributions are summarized as follows.

- We offer a randomized $\frac{1 - 2\epsilon}{4p}$-approximation, single-pass streaming algorithm for the RCCpM problem, with a coreset of size $k + d/\epsilon$. The coreset size is asymptotically optimal. Prior to our work, the best-known coreset size is $O(d \log(k)/\epsilon^2)$ (more details about related work are shown in Table I and discussed in Section III).
- In addition, we introduce and analyze a natural greedy algorithm, which keeps multiple backups for each selected item. We show that this algorithm offers stronger approximation ratios at the expense of a larger coreset size. Specifically, we devise an offline algorithm that requires a coreset of size $O(d \log(k)/\epsilon)$ and yields $\frac{1 - 2\epsilon}{4p}$ and $\frac{1}{2(3p+1)+O(1)}$ approximation for RCC and RCCp, respectively. Besides, the greedy algorithm is empirically effective even against an adaptive adversary.
- The proposed algorithms are evaluated empirically and are shown to achieve superior performance in many application scenarios.

Our techniques can be extended to obtain a $(1 - \epsilon)/2$-approximation one-pass streaming algorithm for the RCC problem with a coreset of size $O(d \log(k)/\epsilon^2)$, an improvement over $O(d \log(k)^2/\epsilon^2)$ in Kazemi et al. [11]. Our approximation ratio for the RCCpM problem can be further strengthened to $\frac{1}{3} \epsilon^{-1} + 2 + O(1)$ when the constraint is a single matroid. A summary of the existing results is displayed in Table I. Our implementation can be found at a Github repository$^{1}$.

The rest of the paper is organized as follows. We formally define the robust coreset problem in Section II. Related work is discussed in Section III. The proposed streaming and offline algorithms are presented and analyzed in Sections IV and V, respectively. Our empirical evaluation is conducted in Section VI, followed by a short conclusion in Section VII.

II. PROBLEM DEFINITION

In this section we define the concept of robust coreset that we consider in this paper. Before discussing the concept of coreset and formally define the problem we study, we briefly review the definitions of submodularity, $p$-matroid, and $p$-system.

Submodularity. Given a set $Y$, a function $f: 2^Y \rightarrow \mathbb{R}_+$ called submodular if for any $X \subseteq Y \subseteq V$ and $v \in V \setminus Y$, it holds $f(v \cup Y) \leq f(v \cup X)$, where $f(v \cup Y) = f(Y \cup v) - f(Y)$ is the marginal gain of $v$ with respect to set $Y$. Function $f$ is called non-decreasing if for any $X \subseteq Y \subseteq V$, it holds $f(Y) \geq f(X)$. Without loss of generality, we can assume that the function $f$ is normalized, i.e., $f(\emptyset) = 0$.

$p$-matroid. For a set $V$, a family of subsets $\mathcal{M} \subseteq 2^V$ is called matroid if it satisfies the following two conditions: (1) downward closeness: if $X \subseteq Y$ and $X \in \mathcal{M}$, then $X \in \mathcal{M}$; (2) augmentation: if $X, Y \in \mathcal{M}$ and $|X| < |Y|$, then $X + v \in \mathcal{M}$ for some $v \in V \setminus X$. For a constant $p$, a $p$-matroid $\mathcal{M} \subseteq 2^V$ is defined as the intersection of $p$ matroids $\{\mathcal{M}_j\}_{j \in [p]}$. The rank of a $p$-matroid $\mathcal{M}$ is defined as $k = \max_{E \in \mathcal{M}} |E|$. A $p$-system for a set $V$ and a constant $p$, a $p$-system $\mathcal{M} \subseteq 2^V$ is defined as follows. Given a set $Y \subseteq V$, a set $X$ is called a base of $Y$ if $X$ is a maximal subset of $Y$, i.e., $X, Y \in \mathcal{M}$, $X \subsetneq Y$ and $X + v \notin \mathcal{M}$ for any $v \in V \setminus X$. We denote the set of bases of $Y$ by $B(Y)$. A tuple $(V, \mathcal{M})$ forms a $p$-system if for any $Y \subseteq V$, it holds $\max_{E \in \mathcal{M}} |E|/\min_{E \in \mathcal{M}} |E| \leq p$. A $p$-matroid is a special case of a $p$-system.

Robust coreset. We consider a set $V$, a non-decreasing submodular function $f: 2^V \rightarrow \mathbb{R}_+$, an integer $d$, and a $p$-matroid $\mathcal{M} \subseteq 2^V$ of rank $k$. A procedure for selecting a solution subset for $f$, which is robust under deletions, is specified by the following three stages.

1) Upon receiving all items in $V$, an algorithm $A_1$ returns a small subset $\hat{R} \subseteq V$ as the coreset.
2) A static adversary deletes a subset $D \subseteq V$ of size at most $d$.
3) An algorithm $A_2$ extracts a feasible solution $I \subseteq R \setminus D$ and $I \in \mathcal{M}$.

$^{1}$https://github.com/Guangyi-Zhang/robust-subm-coreset
The algorithms $A_1$ and $A_2$ can be randomized. We call an adversary static if the adversary is unaware of the random bits used by the algorithm. Thus, one can assume the set of deletions $D$ are fixed before the algorithm is run. The quality of the solution $\text{ALG} = I$ is measured by evaluating the function $f$ on the solution set $\text{ALG}$. We aim that the quality $f(\text{ALG})$ is close to the optimum after deletion, $f(\text{OPT}(D))$, where $\text{OPT}(D) = \arg\max_{S \subseteq V, D \subseteq M} f(S)$. We omit $D$ in $\text{OPT}(D)$ when it is clear from the context. We say that a pair of randomized algorithms $(A_1, A_2)$ yield a $(\alpha, m)$-coreset if

$$\mathbb{E}[f(A_2(\text{ret}(A_1), D))] \geq \alpha f(\text{OPT}) \text{ and } |R(A_1)| \leq m, \quad (1)$$

where algorithm $A_1$ returns a tuple of sets $\text{ret}(A_1) = \{S_i\}$, and $R(A_1) = \bigcup S_i$ is the coreset.

**Problem 1** (Robust coreset for submodular maximization under $p$-matroid constraint ($\text{RCpM}$)). Given a set $V$, a non-decreasing submodular function $f : 2^V \rightarrow \mathbb{R}_+$, a $p$-matroid $M$ of rank $k$, and an unknown set $D$, find an $(\alpha, m)$-coreset $R$.

Notice that there is a trade-off between the approximation ratio $\alpha$ and the coreset size $m$. We typically aim for $m$ to be independent of $|V|$ and grow slowly in $d$. Note that even in the case $D = \emptyset$, i.e., no deletions, extracting an optimal solution from the items of $V$ for the $\text{RCpM}$ problem is an intractable problem. In fact, no polynomial-time algorithms can approximate $f(\text{OPT})$ with a factor better than $1 - 1/e$ in offline computation [13], or better than $1/2$ in a single pass [12].

We refer to Problem 1 with cardinality constraint, single matroid constraint, and $p$-system constraint, as $\text{RCC}$, $\text{RCM}$, and $\text{RCP}$, respectively.

### III. RELATED WORK

**Deletion-robust submodular maximization.** For simplicity, we assume $k = O(d)$. Mitrović et al. [6] study robust submodular maximization against an adaptive adversary who can inspect the coreset before deletion. They give a one-pass constant-approximation algorithm for the $\text{RCC}$ problem with a coreset of size $O(d \log^2(k/e))$. Mirzasoleiman et al. [7] provide another simple and flexible algorithm, which sequentially constructs $d + 1$ solutions by running any existing streaming algorithm. This gives an $1/(4p)$-approximation algorithm for the $\text{RCpM}$ problem in a single pass at the expense of a coreset of larger size $O(dk)$. When an offline coreset procedure is allowed, Feldman et al. [12] propose a $0.514$-approximation algorithm for the $\text{RCC}$ problem with a coreset of size $O(dk)$ by using a two-player protocol.

For a static adversary, Kazemi et al. [11] achieve $(1 - \varepsilon)/2$ approximation for the $\text{RCC}$ problem with coreset size $O(d \log^2(k/e^2))$ and $O(d \log^2(k/e^3))$ for offline and one-pass streaming settings, respectively. Very recently, Dutting et al. [8] generalize the work of Kazemi et al. [11] into a matroid constraint, requiring a coreset size of $O(d \log(k/e)/\varepsilon^2)$.

Compared to prior work, our algorithm achieves the best-known coreset size against a static adversary.

**Max-min robust submodular maximization.** Another line of work on robust submodular maximization studies a different notion of robustness, where adversarial deletions are performed directly on the solution and no further updates to the solution are allowed [14, 15, 16].

**Dynamic submodular maximization.** The input in the dynamic model consists of a stream of updates, which could be either an insertion or a deletion of an item. It is similar to the $\text{RCC}$ setting if all deletions arrive at the end of the stream. However, the focus in the dynamic model is time complexity instead of space complexity. Methods aim to maintain a good-quality solution at any time with a small amortized update time [17, 18, 19].

**Submodular maximization.** The first single-pass streaming algorithm for a cardinality constraint proposed by Badanidiyuru et al. [9] relies on a thresholding technique. This simple technique turns out to yield tight $1/2$-approximation unless the memory depends on $n$ [12, 20]. The memory requirement is later improved from $O(k \log(k)/\varepsilon)$ to $O(k/\varepsilon)$ [21]. For a general $p$-matroid constraint, $1/4p$-approximation has been known [22, 10].

In the offline setting, it is well-known that a simple greedy algorithm achieves optimal $1 - 1/e$ approximation for a cardinality constraint [13, 23]. A modified greedy algorithm obtains $1/(p + 1)$-approximation for a general $p$-system constraint [24, 25]. More sophisticated algorithms with tight $(1 - 1/e)$-approximation for a matroid appeared later [25, 26].

### IV. THE PROPOSED STREAMING ALGORITHM

In this section, we first describe a non-robust streaming algorithm $\text{Exc}$, and then introduce a novel method that enhances it for the $\text{RCpM}$ problem. We also discuss an improved streaming algorithm for the simpler $\text{RCC}$ problem.

Chakrabarti and Kale [10] provide a simple $1/(4p)$-approximation streaming algorithm for non-decreasing submodular maximization under a $p$-matroid constraint. We call their algorithm $\text{Exc}$ because it maintains one feasible solution at all time by exchanging cheap items $W$ in the current solution $I$, for any new valuable item $v$ that cannot be added in $I$ without making the solution infeasible. The $\text{Exc}$ algorithm measures the value of an item by a weight function $w : V \rightarrow \mathbb{R}_+$, which is defined as $w(v) = f(v | I_v)$, where $I_v$ is the feasible solution before processing item $v$. Furthermore, the algorithm measures the value of a subset by extending $w$ with $w(S) = \sum_{u \in S} w(u)$. The algorithm replaces $W$ with $v$ in $I$ when $w(v) \geq (1 + \gamma)w(W)$, for a parameter $\gamma$. We restate the $\text{Exc}$ algorithm of Chakrabarti and Kale [10] in Algorithm 1 and their main result in Theorem 1. We stress that the $\text{Exc}$ algorithm is non-robust and Theorem 1 holds only in the absence of deletions $D$.

**Theorem 1** (Chakrabarti and Kale [10]). Suppose Algorithm 1 is run over items $V$. For any $\gamma > 0$ and any feasible solution $S \subseteq V$ under a $p$-matroid constraint, Algorithm 1 returns a feasible solution $I$ that satisfies

$$f(S) \leq C_\gamma w(I) \leq C_\gamma f(I),$$
Theorem 2. For any $\gamma > 0$, Algorithms 2 and 3 yield an approximation guarantee $(1 - (1 + 1/\gamma)e)/C_\gamma$ for the RCPM problem using a coreset of size $k + d/\epsilon$, where $C_\gamma = (p(\gamma + 1) - 1)(\gamma + 1)/\gamma + 1 + 1/\gamma$. In particular, when $\gamma = 1$, we obtain a $\frac{1-2e}{4p}$-approximation guarantee.

Algorithm 1: Exc streaming algorithm in Chakrabarti and Kale (2015)

Input: parameter $\gamma$
1. $I \leftarrow \emptyset$
2. for $v \in V$ do
3.   $w(v) \leftarrow f(v \mid I)$
4.   $W \leftarrow \text{Exchange}(v,I)$
5.   if $w(v) \geq (1 + \gamma)w(W)$ then
6.      $I \leftarrow I + v - W$
7. return $I$

Algorithm 2: Robust Exc streaming algorithm (RExc)

Input: parameter $\epsilon, \gamma$
1. $I \leftarrow \emptyset, C \leftarrow \emptyset$
2. for $v' \in V$ do
3.   $C \leftarrow C + v'$
4.   if $|C| \geq d/\epsilon$ then
5.      Sample and remove an item $v$ from $C$ with probability proportional to $1/f(v \mid I)$
6.      $w(v) \leftarrow f(v \mid I)$
7.      $W \leftarrow \text{Exchange}(v,I)$
8.      if $w(v) \geq (1 + \gamma)w(W)$ then
9.         $I \leftarrow I + v - W$
10. return $I$ and $C$

where $C_\gamma = (p(\gamma + 1) - 1)(\gamma + 1)/\gamma + 1 + 1/\gamma$. In particular, when $\gamma = 1$, we have $f(S) \leq 4p w(I) \leq 4p f(I)$.

In this paper, we develop a robust extension of the Exc algorithm (RExc), displayed in Algorithms 2 and 3. Algorithm 2 simply inserts a randomized buffer $C$ between the data stream and the Exc algorithm, and Algorithm 3 continues to process items in $C \setminus D$ after deletions and returns a final solution. Note that Algorithms 2 and 3 can be seen as two stages of the Exc algorithm. A parameter $\epsilon$ is used to set the size of the buffer $C$ to $d/\epsilon$. A smaller value of $\epsilon$ and a larger buffer lead to a stronger robust guarantee.

It is easy to see that Algorithm 2 requires a coreset of size at most $k + d/\epsilon$ and at most $O(nd/\epsilon)$ queries to function $f$, where $n = |V|$. Besides, it successfully preserves almost the same approximation guarantee as the non-robust Exc algorithm in the presence of adversarial deletions.

Theorem 3. There exists a one-pass streaming algorithm that yields $(1 - 2e)/2$ approximation guarantee for the RCC problem, with a coreset size $O((d/\epsilon + k) \log(k)/\epsilon)$.

In the rest of this section, we prove Theorem 2.

A. Proof of Theorem 2

As we mentioned before, Algorithms 2 and 3 can be seen as two stages of the non-robust Exc algorithm with input $(V \setminus C) + (C \setminus D)$. To be more specific, first, Algorithm 2 finds a solution $S_1$ by running the Exc algorithm with input $V \setminus C$. Then, Algorithm 2 returns solution $S_1$ and buffer $C$. Then, Algorithm 3 finds a solution $S_2$ by processing the items that are preserved in $C \setminus D$, while starting from feasible solution $S_1$. Finally, Algorithm 3 returns solution $\text{ALG} = S_2 \setminus D$.

By Theorem 1 we know that the solution $S_2$ returned by Algorithm 3 before deletions occur is provably good. This observation is formally stated in the following corollary.

Corollary 4. For any $\gamma > 0$, the feasible solution $S_2$ returned by Algorithm 3, before the deletion of items in $S_2 \cap D$, satisfies $f(O) \leq C_\gamma w(S_2)$, where $O \in \mathcal{M}$ is the optimal solution over data $(V \setminus C) + (C \setminus D)$.

To complete the proof of Theorem 2, we need to show that the solution $S_2$ is robust against deletions, in expectation. We first derive the expected loss in marginal gain of a sampled item by importance sampling due to the adversarial deletions. Intuitively, among a candidate set of items with varied marginal gain, we need to downsample items with larger gain. Otherwise, the adversary could target those items and we are likely to suffer a great loss.

Lemma 5. Consider sets $C, S, D \subseteq V$. Define $d = |D|$. Let $v \in C$ be an item sampled with probability proportional
to $1/f(v | S)$. Then the expected loss in marginal gain of the
item $v$ after deleting $D$ is
\[
\mathbb{E}[f(v | S)1[v \in D]] \leq \frac{d}{|C|} \mathbb{E}[f(v | S)].
\]

**Proof.** We know
\[
\mathbb{E}[f(v | S)] = \sum_{v \in C} f(v | S)p_v = |C|/z,
\]
where $p_v = \frac{1/f(v|S)}{z}$ and $z = \sum_{v \in C} 1/f(v | S)$. If the
sampled item $v$ is in $D$, we suffer a loss of $f(v | S)$, and this
happens with probability $p_v$. Thus, the expected loss is
$f(v | S)p_v = 1/z$. That is to say, every item leads to the same
amount of expected loss. The expected loss after any deletion
set is
\[
\mathbb{E}[f(v | S)1[v \in D]] = \sum_{v \in C} f(v | S)p_v1[v \in D]
= \sum_{v \in C} \mathbb{I}[v \in D]/z \leq d/z = \frac{d}{|C|} \mathbb{E}[f(v | S)]
\]
proving the claim. $\square$

We proceed to show that solution $S_2$ is robust.

**Lemma 6.** For any $\gamma > 0$, given the feasible solution $S_2$ found
by Algorithm 3 before removing items in $S_2 \cap D$, we have
\[
\mathbb{E}[w(S'_2)] \geq (1 - (1 + 1/\gamma)e)\mathbb{E}[w(S_2)],
\]
where $S'_2 = S_2 \setminus D$.

**Proof.** Let $U$ be the set of items that are ever accepted into
the tentative feasible solution in Algorithms 2 and 3, i.e.,
including $S_2$ and those that are first accepted but later swapped.
We first show that $U$ is robust in the sense that $\mathbb{E}[w(U')] \geq
(1 - e)\mathbb{E}[w(U)]$, where $U' = U \setminus D$.

Let $v_i$ be the $i$-th item added into $U$, where $i \leq n = |V|$. We
know that item $v_i$ is sampled from a candidate set $C_i$ with
a probability proportional to $1/w(v_i)$. Besides, $|C_i| \geq d/\epsilon \geq
|D|/\epsilon$. Thus,
\[
\mathbb{E}[w(U')] = \mathbb{E}\left[ \sum_{v \in U} w(v)(1 - 1[v \in D]) \right]
= \mathbb{E}[w(U)] - \mathbb{E}\left[ \sum_{v \in U} w(v)\mathbb{I}[v \in D] \right]
= \mathbb{E}[w(U)] - \sum_{i \leq n} \mathbb{E}\left[ w(v_i)\mathbb{I}[v_i \in D] \right]
\geq \mathbb{E}[w(U)] - \sum_{i \leq n} \mathbb{E}\left[ \frac{|D|}{|C_i|}w(v_i) \right]
= \mathbb{E}[w(U)] - \epsilon \mathbb{E}[w(U)],
\]
where the first inequality is due to Lemma 5.

Next, we show that $S_2$ is robust, too. Let $K = U \setminus S_2$, and
$K' = K \setminus D$. A useful property about $K$ that is shown in

**Algorithm 4:** Offline robust coreset for $RC_p$

**Input:** parameter $\epsilon$
1. $R \leftarrow$ the set of top-$d$ items in $V$ according to $f(\{v\})$
2. $V \leftarrow V \setminus R$, $j \leftarrow 1$, $I_j \leftarrow \emptyset$
3. do
   4. $C_j \leftarrow \text{top-}(\max\left\{ \frac{d}{\epsilon}, 1 \right\})$ items in $V$ w.r.t. $f(v | I_j)$
   5. $R \leftarrow R \cup C_j$
   6. if $|C_j| \geq \frac{d}{\epsilon}$ then
      7. Sample an item $v_j$ from $C_j$ with a probability proportional to $1/f(v_j | I_j)$
      8. $I_{j+1} \leftarrow I_j + v_j$
      9. $V \leftarrow \{v \in V \setminus C_j : I_{j+1} + v \in M\}$, $j \leftarrow j + 1$
4. while $|V| > 0$
5. return $(R, \{I_j\}_j)$

Chakrabarti and Kale [10, Lemma 2] is that $w(S_2)/\gamma \geq w(K)$.
Therefore,
\[
\mathbb{E}[w(K') + w(S'_2)] = \mathbb{E}[w(U')]
\geq (1 - \epsilon)\mathbb{E}[w(U)] = (1 - \epsilon)\mathbb{E}[w(K) + w(S_2)].
\]

By linearity of expectation and rearranging, we have
\[
\mathbb{E}[w(S'_2)] \geq (1 - \epsilon)(\mathbb{E}[w(K)] + \mathbb{E}[w(S_2)]) - \mathbb{E}[w(K')]
\geq (1 - \epsilon)(\mathbb{E}[w(K)] + \mathbb{E}[w(S_2)]) - \mathbb{E}[w(K)]
= (1 - \epsilon)\mathbb{E}[w(S_2)] - \epsilon\mathbb{E}[w(K)]
\geq (1 - \epsilon)\mathbb{E}[w(S_2)] - \epsilon\mathbb{E}[w(S_2)]/\gamma
= (1 - (1 + 1/\gamma)\epsilon)\mathbb{E}[w(S_2)],
\]
completing the proof. $\square$

Finally, we complete the proof of Theorem 2.

**Proof of Theorem 2.** We know that $OPT$ is the optimal so-
lution over data $V \setminus D$, which is worse than the optimum
solution $O$ over $(V \setminus C) + (C \setminus D)$, that is, $f(O) \geq f(OPT)$. Therefore,
\[
\mathbb{E}[f(ALG)] \geq \mathbb{E}[w(ALG)]
\geq \mathbb{E}[w(S_2 \setminus D)]
\geq (1 - (1 + 1/\gamma)\epsilon)\mathbb{E}[w(S_2)] \quad \triangleright \text{Lemma 6}
\geq (1 - (1 + 1/\gamma)\epsilon)f(O)/C_\gamma \quad \triangleright \text{Corollary 4}
\geq (1 - (1 + 1/\gamma)\epsilon)f(OPT)/C_\gamma,
\]
completing the proof. $\square$
V. THE PROPOSED OFFLINE ALGORITHM

We start our exposition by presenting a unified Algorithm 4 to construct a robust coreset for both \(p\)-system and cardinality constraints. However, different algorithms (Algorithms 5 and 7 [27], respectively) are needed to extract the final solution after the deletion of items by the adversary.

Algorithm 4 constructs a robust coreset by iteratively collecting the items with the largest marginal gains with respect to a tentative solution \(I\), and in each iteration, sampling an item from the collected set and adding it into \(I\). Algorithm 5 or 7 extracts the final solution after deletion. The running time in terms of query complexity, i.e., the number of calls to function \(f\), of Algorithms 4, 5 and 7 is \(O(nk)\), \(O((d \log(k)/\epsilon + k)k)\), and \(O((d \log(k)/\epsilon + k) \log(k)/\epsilon)\), respectively. Our main results are stated below.

**Theorem 7.** Algorithms 4 and 5 yield a \(\frac{1}{p+1+(p+1)/(1-\epsilon)}\) approximation guarantee for the \(RCP\) problem, using a coreset of size \(O(d \log(k)/\epsilon + k)\).

Using a proof similar to the one of Theorem 7, we can obtain a stronger approximation ratio for a single matroid constraint, by replacing the greedy algorithm (Step 2) in Algorithm 5 by a more advanced continuous greedy algorithm [25].

**Theorem 8.** Algorithms 4 and a modified Algorithm 5 yield a \(\frac{1}{c_1c_2(1-\epsilon)^2}\) approximation for the \(RCM\) problem, using a coreset of size \(O(d \log(k)/\epsilon + k)\).

In the simpler case of a cardinality constraint, we can achieve a better approximation ratio by a Sieve-like algorithm [9] to extract the final solution.

**Theorem 9.** Algorithms 4 and 7 yield a \((1-2\epsilon)/2\) approximation guarantee for the \(RCC\) problem, using a coreset of size \(O(d \log(k)/\epsilon + k)\).

We will devote the rest of this section for proving Theorem 7. Proof for Theorem 9 is deferred to Appendix [27].

A. **Proof of Theorem 7**

The strategy in Algorithms 4 is to sample-and-keep disjoint candidate sets, which forces the adversary to invest its deletions among these disjoint sets. To ensure a bounded expected loss due to the deletions, we perform importance sampling (also known as “uselessness” sampling) in Lemma 5 among each candidate set. We further show that it is safe to reduce the size of candidate sets harmonically, as the expected marginal gain of the sampled items is non-increasing.

For the remainder of the section, we will adopt the following notation. Let \(\{I_i\}\) be the partial solutions discovered by Algorithm 4, and let \(v_i\) be the item added to \(I_i\), that is, \(I_{i+1} = I_i + v_i\). Let \(C_i\) be the sets from which Algorithm 4 samples \(v_i\). In addition, let \(D\) be the set of deleted items by the adversary. Finally, we write \(I'_i = I_i \setminus D\).

Next we show that the gain of item \(v_j\) is non-increasing in \(j\).

**Lemma 10.** For any \(j < i\), we have \(f(v_j | I_j) \geq f(v_i | I_i)\).

Proof. Deflected to Section A due to space limitation [27].

The following lemma shows the robustness of the tentative partial solution \(I\) built in Algorithm 4, in the sense that \(E[fi(I'_i)]\) is close to \(E[fi(I_i)]\). Intuitively, the expected loss of the first item in \(I\) is small as its candidate set \(C_1\) has a large size \(d/\epsilon\). A subsequent item in \(I\) can be sampled with a decreasing candidate size, because previously added items can help compensate if its candidate set is attacked by the adversary.

**Lemma 11.** \(E[fi(I'_i)] \geq (1 - \epsilon)E[fi(I_i)]\).

Proof. We start by bounding \(E[fi(I'_i)]\),

\[
E[fi(I'_i)] = E\left[\sum_{j < i} f(v_j | I_j \setminus D)\mathbb{1}[v_j \notin D]\right]
\]

\[
\geq E\left[\sum_{j < i} f(v_j | I_j)\mathbb{1}[v_j \notin D]\right]
\]

\[
= E[fi(I_i)] - E\left[\sum_{j < i} f(v_j | I_j)\mathbb{1}[v_j \in D]\right],
\]

where the inequality is due to submodularity.

Now we bound further the second term. For simplicity let us write \(g_j = f(v_j | I_j)\). Recall that \(C_j\) is the set from which Algorithm 4 samples \(v_j\). Note that \(|C_j| \geq \frac{d}{\epsilon}\), and that the sets \(\{C_j\}\) do not overlap. Define \(D_j = C_j \setminus D\). Note that \(D_j\) is also a random variable like \(C_j\), which depends on previously sampled items \(I_j\). Then

\[
E\left[\sum_{j < i} g_j\mathbb{1}[v_j \in D]\right] = E\left[\sum_{j < i} E[g_j | v_j \in D_j] | I_j]\right]
\]

\[
\leq \sum_{j < i} E\left[\frac{|D_j|}{d/j\epsilon} E[g_j | I_j]\right] = \frac{\epsilon}{d} E\left[\sum_{j < i} |D_j| j g_j\right],
\]

where for each \(j\), the outer expectation is taken over \(I_j\) and the inner expectation is over \(v_j\). The inequality follows from Lemma 5.

Let \(\eta = \arg \max_j j g_j\) be the index yielding the highest summand. Since \(g_j\) is non-increasing in \(j\) by Lemma 10, we have

\[
\sum_{j < \eta} |D_j| j g_j \leq \sum_{j < \eta} |D_j| j g_j \leq \sum_{j < \eta} j g_j \leq d \sum_{j < \eta} g_j.
\]

Therefore, we have

\[
\frac{\epsilon}{d} E\left[\sum_{j < i} |D_j| j g_j\right] \leq \frac{\epsilon}{d} E\left[d \sum_{j < i} g_j\right] = \epsilon E[fi(I_i)].
\]

Combining the three inequalities proves that \(E[fi(I'_i)] \geq E[fi(I_i)] - \epsilon E[fi(I_i)]\), completing the proof.

The next lemma follows immediately.

**Lemma 12.** Let \(S\) be a set of items. Then for any \(i\),

\[
E[fi(I'_i \cup S)] \geq (1 - \epsilon)E[fi(I_i \cup S)].
\]

Proof. Deferred to Section A due to space limitation [27].

Finally, we are ready to prove Theorem 7.
Proof of Theorem 7. Let \( i \) be the largest index used by Algorithm 4, and write \( I = I_{i+1} \) be the maximal partial solution in Algorithm 4. Write also \( I' = I \setminus D \). Similarly, \( R \) is our coreset and \( R' = R \setminus D \). To prove the claim, we compare \( I \) with OPT.

\[
\begin{align*}
    f(OPT) & \leq f(I \cup OPT) \leq f(I) + f(OPT \setminus I | I) \\
& \leq f(I) + f((OPT \setminus I) \cap R' | I) + f((OPT \setminus I) \cap R' | I) \\
& \leq f(I) + (p+1)f(S) + f(OPT \setminus R' | I).
\end{align*}
\]

The last step is because any feasible solution in \( R' \), including \((OPT \setminus I) \cap R'\), is within \( p+1 \) approximation of the greedy solution \( H \) of Algorithm 5 [24, 25]. Now we deal with the last term. Note that \( OPT \setminus R' = OPT \setminus R \). Then

\[
    f(OPT \setminus R | I) \leq \sum_{u \in OPT \setminus R} f(u | I) = \sum_{u \in O} f(u | I),
\]

where \( O = \{u \in OPT \setminus R : I + u \notin I \} \). Here the last step is due to the fact that the chosen \( I \) is maximal, and an item will be discarded only when it is infeasible to \( I \).

Let \( O = \{u_1, \ldots, u_{|O|}\} \) be the order in which Algorithm 4 discards the items in \( O \). Define a function \( \pi \) with \( \pi(u_i) = \lfloor \ell/p \rfloor \). Let

\[
    O_j = \{u \in O : I_{j+1} + u \notin I \}.
\]

Note that \( O_j, I_{j+1} \in I \) and \( I_{j+1} + u \notin I \) for every \( u \in O_j \). Thus \( I_{j+1} \) is a maximal independent set in \( Y = I_{j+1} \cup O_j \), and, by definition of \( p \)-system, \( |O_j| \leq p|I_{j+1}| = p|j| \) for all \( j \).

Let \( u = u_\ell \in O_j \setminus O_{j-1} \). Then, \( \ell \leq pj \) and \( \pi(u_\ell) \leq j \).

Since \( u \) is discarded after \( v_j \) is added, \n
\[
    f(u | I) \leq f(v_j | I_j) \leq f(v_j | I_j) \leq f(u | I_j).
\]

Lastly, we have

\[
    \sum_{u \in O} f(u | I) \leq \sum_{u \in O} f(v_{\pi(u)} | I_{\pi(u)}) \leq p \sum_{j} f(v_j | I_j) = pf(I)
\]

Putting everything together, we have

\[
    f(OPT) \leq f(I) + (p+1)f(S) - pf(I) \leq (p+1)f(S) - pf(I)
\]

where the last step is due to Lemma 11.

To bound the coreset size, note that \( |R| \) is bounded by

\[
    d + \sum_{j=1}^{k} |C_j| \leq d + \sum_{j=1}^{k} \max \left\{ 1, \frac{d}{\epsilon} \right\} \leq d + \frac{k \ln(d)+1}{\epsilon},
\]

completing the proof.

VI. EXPERIMENTS

In this section, we evaluate the proposed algorithms against state-of-the-art baselines. All methods are tasked with various subset-selection applications over real-life data. Statistics of the datasets used are summarized in Table II. The applications are described below (Sections VI-A–VI-E) followed by a discussion of experimental results (Section VI-F) and an evaluation of running time (Section VI-G). Further details of the experiments are deferred to Section D [27]. We introduce the competing algorithms and adversaries below.

### a) Algorithms:

Competing algorithms include:

- RExc, the robust Exc algorithm presented in Algorithm 2, Section IV;
- RExc-2, two cascading instances of the RExc algorithm;
- RGrd, the offline robust greedy algorithm presented in Algorithm 4, Section V;
- Exc-dk, a flexible reduction proposed by Mirzasoleiman et al. [7] that constructs \( d+1 \) cascading Exc instances;
- Exc-M, the previous state-of-the-art robust Exc algorithm by Dütting et al. [8] that performs uniform sampling on top of multiple candidate sets, each associated with an increasing threshold on marginal gain.

Their objective values of all methods are normalized by that of an omniscient greedy algorithm, which is aware of deleted items in advance. Algorithms RGrd and Exc-dk are also challenged to an adaptive adversary. A fixed parameter \( \epsilon = 0.5 \) is used to avoid large coresets.

### b) Adversary:

We consider two types of adversaries, static and adaptive, which make deletions over the whole universe of items \( V \) or only over the coreset, respectively. To introduce randomness in a principled way, given an integer \( d \), we simulate an adversary by running the Stochastic Greedy algorithm [33] and obtain a deletion set \( D \) of size \( d \). Concretely, in each iteration, we add into \( D \) the greedy item among a multiple of \( z/d \) random items, where \( z = n \) for a static adversary and \( z \) is the coreset size for an adaptive one.

As a general strategy, we let the adversary delete \( 100 \) items, and we gradually increase the parameter \( d \) in each algorithm until it reaches 100.

### A. Personalized movie recommendation

Robust recommendation is favorable in practice due to uncertain deletions caused by user preference. A popular approach to personalized recommendation [6] is to optimize the following submodular function,

\[
    f_u(S) = (1 - \lambda) \sum_{v \in S} \sim(u,v) + \frac{\lambda k}{|V|} \sum_{w \in V} \max \sim(w,v),
\]

such that \( |S| \leq k \). Here \( \sim(u,v) \) measures the relevance of an item \( v \) to the target user \( u \), and \( \sim(w,v) \) the similarity between two items \( w,v \). The second term represents a notion of representativeness, i.e., for every non-selected item \( w \in V \), there exists some item \( v \in S \) that is similar enough to \( w \).

We choose the Movielens dataset [28], which consists of 9,724 movies and hundreds of users. We obtain feature vectors for users and movies by applying SVD on the user-movie rating matrix, and let \( \sim(\cdot, \cdot) \) be the natural dot product. A
random user is chosen for the recommendation task. A movie may belong to more than one of 20 genres, and we further impose a 2-matroid on a feasible solution \( S \), i.e., every movie can be selected at most once and at most one movie can be selected for each genre. Tradeoff parameter \( \lambda \) is fixed to 0.5. The results are reported in Figure 1(a).

\[
g(S) = \sum_{u \in V} \min_{v \in S} d(u, v),
\]

where \( d(u, v) \) measures the distance between \( u \) and \( v \). Intuitively, for each item \( v \) in the data, there should exist some item \( w \) in the summary \( S \) that is close to \( v \). The above function can be turned into a submodular maximization problem by measuring the total reduction of distance with respect to some item \( w \in V \) instead, i.e., \( f(S) = g(\{w\}) - g(S + \{w\}) \) [34]. We let \( w \) be an arbitrary random item.

We experiment with a dataset of facial images [29], under a partition matroid according to races (5 images per race and \( k = 25 \)), with the distance function being the \( \ell_1 \) metric. The results are reported in Figure 1(b).

C. Geolocation data summarization

We experiment with a similar task as in Section VI-B, except for a different dataset, Uber pickups [31]. Every data point indicates a location of Uber pickups in New York City in April, 2014. We measure the distance by a natural \( \ell_1 \) metric. A partition matroid is imposed according to the base companies (at most 5 pickups for each company and \( k = 25 \)). The results are reported in Figure 1(c).

D. Network influence maximization

For viral-marketing applications in social networks, the goal is to identify a small set of seed nodes who can influence many other users. For popular diffusion models, the number of influenced nodes is a submodular function of the seed set [2]. Here, we consider deletion-robust viral marketing for a simple diffusion model, where a seed node always influences all its neighbors, i.e., \( f(S) = |\bigcup_{v \in S} N(v)| \) returns a dominating set, where \( N(v) \) represents neighbors of \( v \). We choose the dataset of Github social network [30], and specify a cardinality limit of \( k = 20 \). The results are reported in Figure 1(d).

E. Popular song selection

Given song-by-song listening history of users, one wishes to select a set of popular songs \( S \) that can “cover” the most users. A user is covered if she likes at least one song in \( S \). That is, \( f(S) = |\bigcup_{v \in S} L(v)| \), where \( L(v) \) represents the set of users who like song \( v \). We aim for a deletion-robust coreset for such popular songs. Concretely, we use the million song dataset [32], consisting of triples representing a user, song, and play count. We assume that a user likes a song if the song is played more than once. We impose a cardinality limit of \( k = 20 \). The results are reported in Figure 1(e).

B. Facial image selection

Exemplar-based applications, such as nearest-neighbor models and recommender systems, are ubiquitous in data science. However, the “right to be forgotten” can lead to the case where some items must be be deleted [5]. In such cases, a robust coreset is desirable, so as to maintain a representative summary for applications after data-item deletions.

A dataset \( V \) can be summarized by a representative subset of data \( S \) via minimizing the classic \( k \)-medoid function,

\[
g(S) = \sum_{u \in V} \min_{v \in S} d(u, v),
\]

where \( d(u, v) \) measures the distance between \( u \) and \( v \). Intuitively, for each item \( v \) in the data, there should exist some item \( w \) in the summary \( S \) that is close to \( v \). The above function can be turned into a submodular maximization problem by measuring the total reduction of distance with respect to some item \( w \in V \) instead, i.e., \( f(S) = g(\{w\}) - g(S + \{w\}) \) [34]. We let \( w \) be an arbitrary random item.

We experiment with a dataset of facial images [29], under a partition matroid according to races (5 images per race and \( k = 25 \)), with the distance function being the \( \ell_1 \) metric. The results are reported in Figure 1(b).
F. Discussion of results

Overall, against a static adversary, the proposed RGrd algorithm performs and converges with the smallest coreset, while the proposed RExc algorithm achieves relatively good performance while requiring the most parsimonious coreset. Note that the size difference in the coreset will become more extreme as \( k \) increases. For an adaptive adversary, the RGrd algorithm remains the most robust.

The Exc-M algorithm has the worst performance most of the time, except for tasks with a simple cardinality constraint and a relatively large coreset size. This behavior illustrates the insufficient efficacy of uniform sampling. Uniform sampling on top of thresholded candidate sets, which is adopted by many previous robust algorithms [11, 8], faces a dilemma between a large coreset or a crude distinction of item importance (i.e., few crude thresholds due to large \( \epsilon \)). This issue is properly addressed by the non-uniform sampling technique in this paper.

On the other hand, cascading instances in the Exc-dk algorithm appears to be another promising way for preserving valuable items in stream computation. However, this approach comes with a cost of expensive computation (see Section VI-G). Besides, its coreset size explodes even with a moderate value of \( d \). Staying with a small \( d \) parameter, however, fails to secure a theoretical guarantee when more items are deleted.

Algorithms RExc and Exc-dk preserve valuable and compatible items in two different ways. This naturally suggests that one can combine the best of both worlds by constructing a small number of cascading RExc instances. Then one is expected to further enhance the performance while maintaining a parsimonious coreset and a strong guarantee. This is indeed the case as reflected by the remarkable performance of the RExc-2 algorithm, which uses merely two instances of RExc.

In summary, we conclude that the RGrd algorithm is a reliable choice if an offline algorithm is allowed. In a streaming setting, a small number of cascading RExc instances is recommended.

G. Running time analysis

The running time of all algorithms over the song dataset is shown in Figure 2. The most significant message of Figure 2 is that the Exc-dk algorithm is computationally costly when the value of parameter \( d \) grows.

VII. Conclusion

In the presence of adversarial deletions up to \( d \) items, we propose a single-pass streaming algorithm that yields \((1 - 2\epsilon)/(4p)\)-approximation for maximizing a non-decreasing submodular function under a general \( p \)-matroid constraint and requires an (asymptotically) optimal coreset size \( k + d/\epsilon \), where \( k \) is the maximum size of a feasible solution. Besides, we develop an offline greedy algorithm that guarantees stronger approximation ratios, and performs effectively even against an adaptive adversary.

One vital tool for robustness in the proposed algorithms is “uselessness” sampling that preserves valuable items within the candidate set and avoids great loss caused by adversarial deletions in expectation. Another insight is a close connection between robustness and streaming algorithms. The latter ensures a quality guarantee given an arbitrary arrival order of items, including the specific random order introduced by the sampling.

Potential directions for future work include a potentially stronger approximation ratio in the offline setting, extensions to non-monotone submodular maximization, and a stronger adaptive adversary.

Acknowledgment

This research is supported by the Academy of Finland projects MALSOme (343045), AIDA (317085) and MLDB (325117), the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project SoBigData++ (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] A. Krause and D. Golovin, “Submodular function maximization.” Tractability, vol. 3, pp. 71–104, 2014.
[2] D. Kempe, J. Kleinberg, and É. Tardos, “Maximizing the spread of influence through a social network,” Theory Of Computing, vol. 11, no. 4, pp. 105–147, 2015.
[3] K. Wei, R. Iyer, and J. Bilmes, “Submodularity in data subset selection and active learning,” in International Conference on Machine Learning. PMLR, 2015, pp. 1954–1963.
[4] H. Lin and J. Bilmes, “A class of submodular functions for document summarization,” in Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, 2011, pp. 510–520.
[5] P. Voigt and A. Von dem Bussche, “The eu general data protection regulation (gdpr),” A Practical Guide. 1st Ed., Cham: Springer International Publishing, vol. 10, no. 3152676, pp. 10–5555, 2017.
[6] S. Mitrovic, I. Bogunovic, A. Norouzi-Fard, J. Tarnawski, and V. Cevher, “Streaming robust submodular maximization: A partitioned thresholding approach,” arXiv preprint arXiv:1711.02598, 2017.
[7] B. Mirzasoleiman, A. Karbasi, and A. Krause, “Deletion-robust submodular maximization: Data summarization
with “the right to be forgotten”,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 2449–2458.

[8] P. Düttting, F. Fusco, S. Lattanzi, A. Norouzi-Fard, and M. Zadimoghaddam, “Deletion robust submodular maximization over matroids,” *arXiv preprint arXiv:2201.13128*, 2022.

[9] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. 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, pp. 671–680.

[10] A. Chakrabarti and S. Kale, “Submodular maximization meets streaming: Matchings, matroids, and more,” *Mathematical Programming*, vol. 154, no. 1, pp. 225–247, 2015.

[11] E. Kazemi, M. Zadimoghaddam, and A. Karbasi, “Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints,” in *International conference on machine learning*. PMLR, 2018, pp. 2544–2553.

[12] M. Feldman, A. Norouzi-Fard, O. Svensson, and R. Zenklusen, “The one-way communication complexity of submodular maximization with applications to streaming and robustness,” in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 1363–1374.

[13] G. L. Nemhauser and L. A. Wolsey, “Best algorithms for approximating the maximum of a submodular set function,” *Mathematics of operations research*, vol. 3, no. 3, pp. 177–188, 1978.

[14] A. Krause, H. B. McMahan, C. Guestrin, and A. Gupta, “Robust submodular observation selection,” *Journal of Machine Learning Research*, vol. 9, no. 12, 2008.

[15] J. B. Orlin, A. S. Schulz, and R. Udwani, “Robust monotone submodular function maximization,” *Mathematical Programming*, vol. 172, no. 1, pp. 505–537, 2018.

[16] I. Bogunovic, S. Mitrovic, J. Scarlett, and V. Cevher, “Robust submodular maximization: A non-uniform partitioning approach,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 508–516.

[17] S. Lattanzi, S. Mitrović, A. Norouzi-Fard, J. Tarnawski, and M. Zadimoghaddam, “Fully dynamic algorithm for constrained submodular optimization,” in *NeurIPS*, 2020.

[18] M. Monemizadeh, “Dynamic submodular maximization,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[19] X. Chen and B. Peng, “On the complexity of dynamic submodular maximization,” *arXiv preprint arXiv:2111.03198*, 2021.

[20] A. Norouzi-Fard, J. Tarnawski, S. Mitrovic, A. Zandieh, A. Mousavifar, and O. Svensson, “Beyond 1/2-approximation for submodular maximization on massive data streams,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 3829–3838.

[21] E. Kazemi, M. Mitrovic, M. Zadimoghaddam, S. Lattanzi, and A. Karbasi, “Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 3311–3320.

[22] C. Chekuri, S. Gupta, and K. Quanrud, “Streaming algorithms for submodular function maximization,” in *International Colloquium on Automata, Languages, and Programming*. Springer, 2015, pp. 318–330.

[23] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions—I,” *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[24] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, “An analysis of approximations for maximizing submodular set functions—II,” in *Polyhedral combinatorics*. Springer, 1978, pp. 73–87.

[25] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák, “Maximizing a monotone submodular function subject to a matroid constraint,” *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[26] Y. Filim and J. Ward, “A tight combinatorial algorithm for submodular maximization subject to a matroid constraint,” in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 2012, pp. 659–668.

[27] G. Zhang, N. Tatti, and A. Gionis, “Coresets remembered and items forgotten: submodular maximization with deletions,” *arXiv preprint arXiv:2203.01241*, 2022.

[28] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[29] Z. Zhang, Y. Song, and H. Qi, “Age progression/regression by conditional adversarial autoencoder,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5810–5818.

[30] B. Rozemberczki, C. Allen, and R. Sarkar, “Multi-scale attributed node embedding,” 2019.

[31] Kaggle, “Uber pickups in new york city,” 2020. [Online]. Available: https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city

[32] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[33] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause, “Lazier than lazy greedy,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[34] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, “Distributed submodular maximization: Identifying representative elements in massive data,” *Advances in Neural Information Processing Systems*, vol. 26, 2013.