Coresets remembered and items forgotten: submodular maximization with deletions

Anonymous Author(s)
Affiliation
Address
email

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 $\mathcal{O}(d\log(k)/\epsilon)$. 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 (RCpM), or a p-system constraint (RCp). Recall that a cardinality constraint is known as a uniform matroid. Throughout the paper, we are mostly interested in the more challenging case where $k=\mathcal{O}(d)$ and k is the maximum size of a feasible solution.

Our contributions are summarized as follows.

- We offer a randomized $\frac{1-2\epsilon}{4p}$-approximation, single-pass streaming algorithm for the RCpM problem, with a coreset of size $k+d/\epsilon$. The coreset size is asymptotically

TABLE I: A summary of existing results on robust coresets. For simplicity, it is assumed that $k = \mathcal{O}(d)$.

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

optimal. Prior to our work, the best-known coreset size is $\mathcal{O}(d\log(k)/\epsilon^2)$ (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 $\mathcal{O}(d\log(k)/\epsilon)$ and yields $\frac{1-\epsilon}{2}$ and $\frac{1}{2(p+1)+\mathcal{O}(\epsilon)}$ approximation for RCC and RCp, 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 $\mathcal{O}(d\log(k)/\epsilon^2)$, an improvement over $\mathcal{O}(d\log(k)^2/\epsilon^3)$ in Kazemi et al. [11]. Our approximation ratio for the RCpM problem can be further strengthened to $\frac{1}{e/(e-1)+2+\mathcal{O}(\epsilon)}$ when the constraint is a single matroid. A summary of the existing results is displayed in Table I. Our implementation can be found at an anonymous 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 V, a function $f : 2^V \to \mathbb{R}_+$ is called *submodular* if for any $X \subseteq Y \subseteq V$ and $v \in V \setminus Y$, it holds $f(v \mid Y) \le f(v \mid X)$, where $f(v \mid Y) = f(Y+v) - f(Y)$ 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 $f(Y) \ge f(X)$. 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 $Y \in \mathcal{M}$, then $X \in \mathcal{M}$; (2) augmentation: if $X, Y \in \mathcal{M}$ and $|X| < |Y|$, then $X + v \in \mathcal{M}$ for some $v \in Y \setminus X$. 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 $k = \max_{S\in\mathcal{M}} |S|$.

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., $X \in \mathcal{M}$, $X \subseteq Y$ and $X + v \notin \mathcal{M}$ for any $v \in Y \setminus X$. We denote the set of bases of Y by $\mathcal{B}(Y)$. A tuple $(V, \mathcal{M})$ forms a p-system if for any $Y \subseteq V$, it is $\frac{\max_{X\in\mathcal{B}(Y)} |X|}{\min_{X\in\mathcal{B}(Y)} |X|} \le p$. A p-matroid is a special case of a p-system.

Robust coreset. We consider a set V, a non-decreasing submodular function $f : 2^V \to \mathbb{R}_+$, 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 $\mathcal{A}_1$ returns a small subset $R \subseteq V$ as the coreset.
2) A static adversary deletes a subset $D \subseteq V$ of size at most d.
3) An algorithm $\mathcal{A}_2$ extracts a feasible solution $I \subseteq R \setminus D$ and $I \in \mathcal{M}$.

The algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ 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

[1] https://github.com/anon-repo/robust-subm-coreset

function f on the solution set ALG. We aim that the quality $f(\mathrm{ALG})$ is close to the optimum after deletion, $f(\mathrm{OPT}(D))$, where $\mathrm{OPT}(D) = \arg\max_{S \subseteq V \setminus D, S \in \mathcal{M}} f(S)$. We omit D in $\mathrm{OPT}(D)$ when it is clear from the context. We say that a pair of randomized algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ yield a (α, m)-*coreset* if

$$\mathsf{E}[f(\mathcal{A}_2(\mathrm{ret}(\mathcal{A}_1), D))] \geq \alpha f(\mathrm{OPT}) \quad \text{and} \quad |R(\mathcal{A}_1)| \leq m, \tag{1}$$

where algorithm $\mathcal{A}_1$ returns a tuple of sets $\mathrm{ret}(\mathcal{A}_1) = \{S_i\}$, and $R(\mathcal{A}_1) = \cup_i S_i$ is the coreset.

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

Notice that there is a trade-off between the approximation ratio α 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 RCpM problem is an intractable problem. In fact, no polynomial-time algorithms can approximate $f(\mathrm{OPT})$ 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 RCC, RCM, and RCp, respectively.

III. Related work

Deletion-robust submodular maximization. For simplicity, we assume $k = \mathcal{O}(d)$. 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 RCC problem with a coreset of size $\mathcal{O}(d \log^3(k)/\epsilon)$. 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 RCpM problem in a single pass at the expense of a coreset of larger size $\mathcal{O}(dk)$. When an offline coreset procedure is allowed, Feldman et al. [12] propose a 0.514-approximation algorithm for the RCC problem with a coreset of size $\mathcal{O}(dk)$ by using a two-player protocol.

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

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 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 $\mathcal{O}(k \log(k)/\epsilon)$ to $\mathcal{O}(k/\epsilon)$ [21]. For a general p-matroid constraint, $1/4p$-approximation has been known [10, 22].

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 Exc, and then introduce a novel method that enhances it for the RCpM problem. We also discuss an improved streaming algorithm for the simpler RCC problem.

Chakrabarti and Kale [22] provide a simple $1/(4p)$-approximation streaming algorithm for non-decreasing submodular maximization under a p-matroid constraint. We call their algorithm 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 Exc algorithm measures the value of an item by a weight function $w : V \to \mathbb{R}_+$, which is defined as $w(v) = f(v \mid I_v)$, 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 γ. We restate the Exc algorithm of Chakrabarti and Kale [22] in Algorithm 1 and their main result in Theorem 1. We stress that the Exc algorithm is non-robust and Theorem 1 holds only in the absence of deletions D.

Theorem 1 (Chakrabarti and Kale [22])**.** *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*

$$f(S) \leq C_\gamma w(I) \leq C_\gamma f(I),$$

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

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

```
Input: parameter $\gamma$
$I \leftarrow \emptyset$
for $v \in V$ do
    $w(v) \leftarrow f(v \mid I)$
    $W \leftarrow$ Exchange$(v, I)$
    if $w(v) \geq (1+\gamma)w(W)$ then
        $I \leftarrow I + v - W$
return $I$

Function Exchange $(v, I)$:
    for $j \in [p]$ do
        if $I + v \notin \mathcal{M}_j$ then
            $u_j \leftarrow \arg\min_{u \in I: I+v-u \in \mathcal{M}_j} w(u)$
    return $\{u_j\}_{j \in [p]}$
```

Algorithm 2: Robust Exc streaming algorithm (RExc)

```
Input: parameter $\epsilon$, $\gamma$
$I \leftarrow \emptyset, C \leftarrow \emptyset$
for $v' \in V$ do
    $C \leftarrow C + v'$
    if $|C| \geq d/\epsilon$ then
        Sample and remove an item $v$ from $C$ with
          probability proportional to $1/f(v \mid I)$
        $w(v) \leftarrow f(v \mid I)$
        $W \leftarrow$ Exchange$(v, I)$
        if $w(v) \geq (1+\gamma)w(W)$ then
            $I \leftarrow I + v - W$
return $I$ and $C$
```

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 ϵ is used to set the size of the buffer C to d/ϵ. A smaller value of ϵ 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 $\mathcal{O}(nd/\epsilon)$ 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 2. *For any $\gamma > 0$, Algorithms 2 and 3 yield an approximation guarantee $(1-(1+1/\gamma)\epsilon)/C_\gamma$ 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 $\frac{1-2\epsilon}{4p}$-approximation guarantee.*

For the simpler cardinality constraint, we can obtain a tighter approximation ratio at the expense of a larger coreset size $\mathcal{O}(d\log(k)/\epsilon^2)$. This is an improvement over the state-of-the-art $\mathcal{O}(d\log^2(k)/\epsilon^3)$ in Kazemi et al. [11]. The main idea is the utilization of importance sampling (Lemma 5) on top of the robust Sieve algorithm in Kazemi et al. [11]. We defer the details to Section B in Appendix [27].

Algorithm 3: Construction of RExc solution after deletions

```
Input: $I$, $C$, $D$, and parameter $\gamma$
for $v \in C \setminus D$ do
    $w(v) \leftarrow f(v \mid I)$
    $W \leftarrow$ Exchange$(v, I)$
    if $w(v) \geq (1+\gamma)w(W)$ then
        $I \leftarrow I + v - W$
return $I \setminus D$
```

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

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 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 $1/f(v \mid S)$. Then the expected loss in marginal gain of the item v after deleting D is*

$$\mathsf{E}\left[f(v \mid S)\mathbb{1}[v \in D]\right] \leq \frac{d}{|C|}\mathsf{E}[f(v \mid S)].$$

Proof. We know

$$\mathsf{E}[f(v \mid S)] = \sum_{v \in C} f(v \mid S)p_v = |C|/z,$$

where $p_v = \frac{1/f(v|S)}{z}$ and $z = \sum_{v \in C} 1/f(v \mid S)$. If the sampled item v is in D, we suffer a loss of $f(v \mid S)$, and this happens with probability p_v. Thus, the expected loss is $f(v \mid S)p_v = 1/z$. That is to say, every item leads to the same amount of expected loss. The expected loss after any deletion set is

$$\begin{aligned}\mathsf{E}[f(v \mid S)\mathbb{1}[v \in D]] &= \sum_{v \in C} f(v \mid S)p_v\mathbb{1}[v \in D] \\ &= \sum_{v \in C} \mathbb{1}[v \in D]/z \le d/z = \frac{d}{|C|}\mathsf{E}[f(v \mid S)]\end{aligned}$$

proving the claim. □

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*

$$\mathsf{E}[w(S_2')] \ge (1 - (1 + 1/\gamma)\epsilon)\mathsf{E}[w(S_2)],$$

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 $\mathsf{E}[w(U')] \ge (1 - \epsilon)\mathsf{E}[w(U)]$, where $U' = U \setminus D$.

Let v_i be the i-th item added into U, where $i \le n = |V|$. We know that item v_i is sampled from a candidate set C_i with a probability proportional to $1/w(v_i)$. Besides, $|C_i| \ge d/\epsilon \ge |D|/\epsilon$. Thus,

$$\begin{aligned}\mathsf{E}[w(U')] &= \mathsf{E}\big[\sum_{v \in U} w(v)(1 - \mathbb{1}[v \in D])\big] \\ &= \mathsf{E}[w(U)] - \mathsf{E}\big[\sum_{v \in U} w(v)\mathbb{1}[v \in D]\big] \\ &= \mathsf{E}[w(U)] - \sum_{i \le n} \mathsf{E}\big[w(v_i)\mathbb{1}[v_i \in D]\big] \\ &\ge \mathsf{E}[w(U)] - \sum_{i \le n} \mathsf{E}\big[\frac{|D|}{|C_i|}w(v_i)\big] \\ &\ge \mathsf{E}[w(U)] - \sum_{i \le n} \mathsf{E}\big[\epsilon w(v_i)\big] \\ &= \mathsf{E}[w(U)] - \epsilon\mathsf{E}[w(U)],\end{aligned}$$

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 Chakrabarti and Kale [22, Lemma 2] is that $w(S_2)/\gamma \ge w(K)$. Therefore,

$$\begin{aligned}&\mathsf{E}[w(K') + w(S_2')] = \mathsf{E}[w(U')] \\ &\qquad \ge (1 - \epsilon)\mathsf{E}[w(U)] = (1 - \epsilon)\mathsf{E}[w(K) + w(S_2)].\end{aligned}$$

By linearity of expectation and rearranging, we have

$$\begin{aligned}\mathsf{E}[w(S_2')] &\ge (1 - \epsilon)(\mathsf{E}[w(K)] + \mathsf{E}[w(S_2)]) - \mathsf{E}[w(K')] \\ &\ge (1 - \epsilon)(\mathsf{E}[w(K)] + \mathsf{E}[w(S_2)]) - \mathsf{E}[w(K)] \\ &= (1 - \epsilon)\mathsf{E}[w(S_2)] - \epsilon\mathsf{E}[w(K)] \\ &\ge (1 - \epsilon)\mathsf{E}[w(S_2)] - \epsilon\mathsf{E}[w(S_2)]/\gamma \\ &= (1 - (1 + 1/\gamma)\epsilon)\mathsf{E}[w(S_2)],\end{aligned}$$

completing the proof. □

Algorithm 4: Offline robust coreset for RCp

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

Algorithm 5: Construction of RCp solution after deletion

```
Input: Coreset and auxiliary information $(R, \{I_j\}_j)$
       returned by Algorithm 4, set of deleted items $D$
$I \leftarrow I_i$ where $i = \max_j j$
$H \leftarrow$ a greedy solution using items in $R \setminus D$
return the best solution among $\{I \setminus D, H\}$
```

Finally, we complete the proof of Theorem 2.

Proof of Theorem 2. We know that OPT is the optimal solution over data $V \setminus D$, which is worse than the optimum solution O over $(V \setminus C) + (C \setminus D)$, that is, $f(O) \ge f(\mathrm{OPT})$. Therefore,

$$\begin{aligned}\mathsf{E}[f(\mathrm{ALG})] &\ge \mathsf{E}[w(\mathrm{ALG})] \\ &= \mathsf{E}[w(S_2 \setminus D)] \\ &\ge (1 - (1 + 1/\gamma)\epsilon)\mathsf{E}[w(S_2)] && \triangleright \text{Lemma 6} \\ &\ge (1 - (1 + 1/\gamma)\epsilon)f(O)/C_\gamma && \triangleright \text{Corollary 4} \\ &\ge (1 - (1 + 1/\gamma)\epsilon)f(\mathrm{OPT})/C_\gamma,\end{aligned}$$

completing the proof. □

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, 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 $\mathcal{O}(nk)$, $\mathcal{O}((d\log(k)/\epsilon + k)k)$, and $\mathcal{O}((d\log(k)/\epsilon + k)\log(k)/\epsilon)$, 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* $\mathcal{O}(d\log(k)/\epsilon + k)$.

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* $\frac{1}{e/(e-1)+2/(1-\epsilon)}$ *approximation for the RCM problem, using a coreset of size* $\mathcal{O}(d\log(k)/\epsilon + k)$.

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* $\mathcal{O}(d\log(k)/\epsilon + k)$.

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* $f(v_j \mid I_j) \geq f(v_i \mid I_i)$.

Proof. Deferred 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 $\mathsf{E}[f(I_i')]$ is close to $\mathsf{E}[f(I_i)]$. Intuitively, the expected loss of the first item in I is small as its candidate set C_1 has a large size d/ϵ. 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. $\mathsf{E}[f(I_i')] \geq (1-\epsilon)\mathsf{E}[f(I_i)]$.

Proof. We start by bounding $\mathsf{E}[f(I_i')]$,

$$\begin{aligned}\mathsf{E}[f(I_i')] &= \mathsf{E}[\sum_{j<i} f(v_j \mid I_j \setminus D)\mathbb{1}[v_j \notin D]] \\ &\geq \mathsf{E}[\sum_{j<i} f(v_j \mid I_j)\mathbb{1}[v_j \notin D]] \\ &= \mathsf{E}[f(I_i)] - \mathsf{E}[\sum_{j<i} f(v_j \mid I_j)\mathbb{1}[v_j \in D]],\end{aligned}$$

where the inequality is due to submodularity.

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

$$\begin{aligned}\mathsf{E}\Big[\sum_{j<i} g_j \mathbb{1}[v_j \in D]\Big] &= \sum_{j<i} \mathsf{E}\big[\mathsf{E}\left[g_j \mathbb{1}[v_j \in D_j] \mid I_j\right]\big] \\ \leq \sum_{j<i} \mathsf{E}\left[\frac{|D_j|}{d/j\epsilon}\mathsf{E}[g_j \mid I_j]\right] &= \frac{\epsilon}{d}\mathsf{E}\Big[\sum_{j<i} |D_j| j g_j\Big],\end{aligned}$$

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

$$\sum_{j<i} |D_j| j g_j \leq \sum_{j<i} |D_j| \eta g_\eta \leq d\eta g_\eta \leq d\sum_{j\leq\eta} g_j \leq d\sum_{j<i} g_j.$$

Therefore, we have

$$\frac{\epsilon}{d}\mathsf{E}\Big[\sum_{j<i} |D_j| j g_j\Big] \leq \frac{\epsilon}{d}\mathsf{E}\Big[d\sum_{j<i} g_j\Big] = \epsilon\mathsf{E}[f(I_i)].$$

Combining the three inequalities proves that $\mathsf{E}[f(I_i')] \geq \mathsf{E}[f(I_i)] - \epsilon\mathsf{E}[f(I_i)]$, completing the proof. □

The next lemma follows immediately.

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

$$\mathsf{E}[f(I_i' \cup S)] \geq (1-\epsilon)\mathsf{E}[f(I_i \cup S)].$$

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 let $I = I_{i+1}$ 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.

$$\begin{aligned}&f(\text{OPT}) \leq f(I \cup \text{OPT}) \leq f(I) + f(\text{OPT} \setminus I \mid I) \\ &\leq f(I) + f((\text{OPT} \setminus I) \cap R' \mid I) + f((\text{OPT} \setminus I) \setminus R' \mid I) \\ &\leq f(I) + (p+1)f(H) + f(\text{OPT} \setminus R' \mid I).\end{aligned}$$

The last step is because any feasible solution in R', including $(\text{OPT} \setminus I) \cap R'$, is within $p+1$ approximation of the greedy

TABLE II: Datasets statistics

Dataset	$n = \|V\|$	k	$\mathcal{M}$
Movielens [28]	22 046	20	2-matroid
Facial images [29]	23 705	25	1-matroid
Github social network [30]	37 700	20	cardinality
Uber pickups [31]	50 000	25	1-matroid
Songs [32]	137 543	20	cardinality

solution H of Algorithm 5 [24, 25]. Now we deal with the last term. Note that $\mathrm{OPT} \setminus R' = \mathrm{OPT} \setminus R$. Then

$$f(\mathrm{OPT} \setminus R \mid I) \le \sum_{u \in \mathrm{OPT}\setminus R} f(u \mid I) = \sum_{u \in O} f(u \mid I),$$

where $O = \{u \in \mathrm{OPT} \setminus R : I + u \notin I\}$. 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 $O = u_1, \dots, u_{|O|}$ be the order in which Algorithm 4 discards the items in O. Define a function π with $\pi(u_\ell) = \lceil \ell/p \rceil$. 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, $|O_j| \le p|I_{j+1}| = pj$ for all j.

Let $u = u_\ell \in O_j \setminus O_{j-1}$. Then, $\ell \le pj$ and $\pi(u_\ell) \le j$. Since u is discarded after v_j is added,

$$f(u \mid I) \le f(u \mid I_j) \le f(v_j \mid I_j) \le f(v_{\pi(u)} \mid I_{\pi(u)}).$$

Lastly, we have

$$\sum_{u \in O} f(u \mid I) \le \sum_{u \in O} f(v_{\pi(u)} \mid I_{\pi(u)}) \le p \sum_{j} f(v_j \mid I_j) = pf(I).$$

Putting everything together, we have

$$\begin{aligned} f(\mathrm{OPT}) &\le f(I) + (p+1)f(H) + pf(I) \\ &\le (p + 1 + \frac{p+1}{1-\epsilon})\mathsf{E}[f(\mathrm{ALG})], \end{aligned}$$

where the last step is due to Lemma 11.

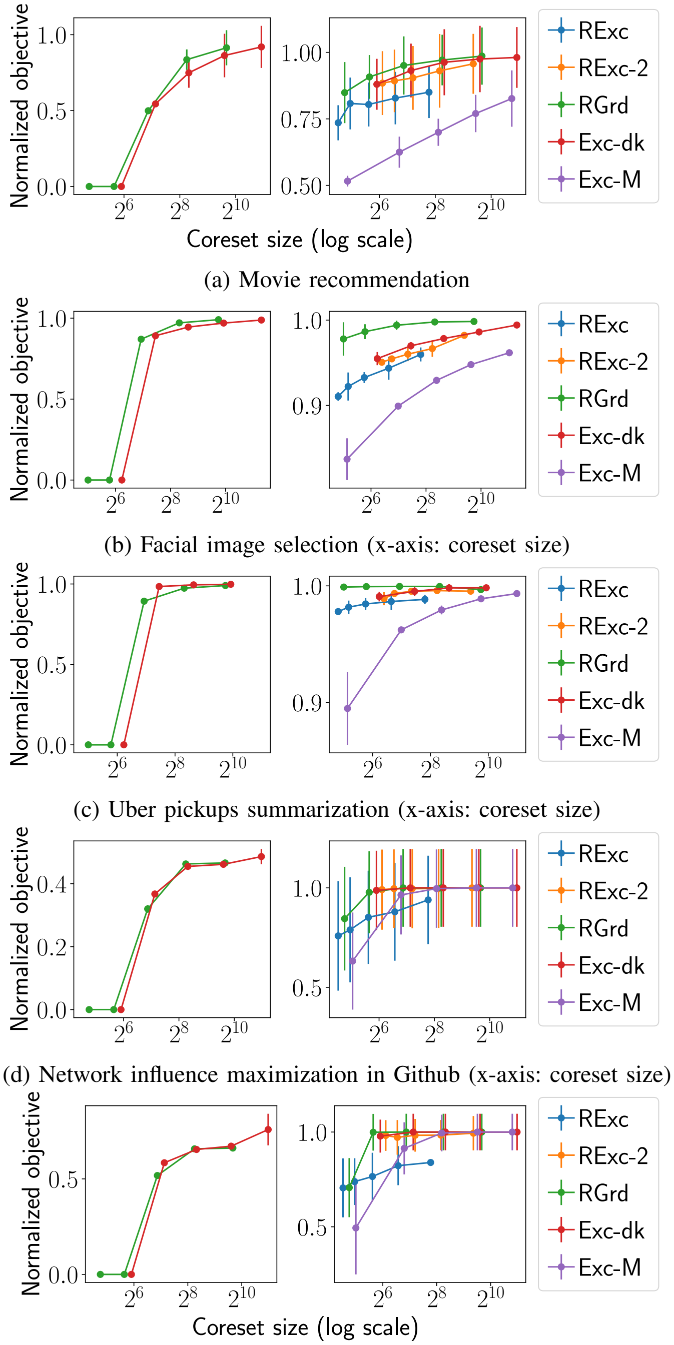
To bound the coreset size, note that $|R|$ is bounded by

$$d + \sum_{j=1}^{k} |C_j| \le d + \sum_{j=1}^{k} \max\left\{1, \frac{d}{\epsilon j}\right\} \le d + k + d(\ln(k)+1)/\epsilon,$$

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) Movie recommendation

(b) Facial image selection (x-axis: coreset size)

(c) Uber pickups summarization (x-axis: coreset size)

(d) Network influence maximization in Github (x-axis: coreset size)

(e) Popular song selection

Fig. 1: Experiment results. The adversary (left: adaptive, right: static) deletes items of a fixed size 100. Parameter d (and coreset size) in each algorithm is gradually increased to 100.

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.

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,

$$f_u(S) = (1-\lambda)\sum_{v\in S}\mathrm{sim}(u,v) + \frac{\lambda k}{|V|}\sum_{w\in V}\max_{v\in S}\mathrm{sim}(w,v),$$

such that $|S| \le k$. Here $\mathrm{sim}(u,v)$ measures the relevance of an item v to the target user u, and $\mathrm{sim}(w,v)$ 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 $\mathrm{sim}(\cdot,\cdot)$ 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 λ is fixed to 0.5. The results are reported in Figure 1(a).

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,

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

where $d(u,v)$ measures the distance between u and v. Intuitively, for each item v in the data, there should exist some item u 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 the 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 ℓ_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 ℓ_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., $f(S) = |\cup_{v\in S} N(v)|$ 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, $f(S) = |\cup_{v\in S} L(v)|$, 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).

F. Discussion of results

Overall, against a static adversary, the proposed RGrd algorithm performs the best 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

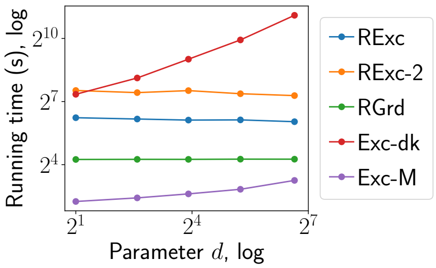


Fig. 2: Running time on the song dataset

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 ϵ). 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 an 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.

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. Mitrović, 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. Dütting, 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] 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.

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

[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. Filmus 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] A. Author(s), "Appendix," 2022. [Online]. Available: https://github.com/anon-repo/robust-subm-coreset/blob/main/supp.pdf

[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. Vondrák, 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.

Algorithm 6: Streaming robust coreset for RCC

```
   Input: parameter $\epsilon$
1  $P \leftarrow$ the set of top-$d$ singletons seen so far
2  $\Delta_d \leftarrow$ the value of the top $(d+1)$-th singleton
3  for each new item $v$ in the stream do
4      Update $\Delta_d$ and $P$
5      if $P$ is changed then
6          $v \leftarrow$ the swapped-out item in $P$
7      $T \leftarrow \{(1+\epsilon)^i : \frac{\Delta_d}{2k(1+\epsilon)} \leq (1+\epsilon)^i \leq \Delta_d, i \in \mathbb{N}\}$
8      for each $\tau \in T$ in parallel do
9          if $f(v \mid I_\tau) \geq \tau$ and $|I_\tau| < k$ then
10             $C_\tau \leftarrow C_\tau + v$
11         if $|C_\tau| \geq d/\epsilon$ then
12             Sample an item $v$ from $C_\tau$ with a probability
                 proportional to $1/f(v \mid I_\tau)$
13             $I_\tau \leftarrow I_\tau + v$
14             $C_\tau \leftarrow \{v \in C_\tau : f(v \mid I_\tau) \geq \tau\}$
15 $R \leftarrow P \cup \left(\bigcup_{\tau \in T} C_\tau\right) \cup \left(\bigcup_{\tau \in T} I_\tau\right)$
16 return $R, \{I_\tau\}_{\tau \in T}$
```

Appendix

A. Omitted proofs

Proof of Lemma 10. Note that v_j is sampled from C_j and v_i is sampled from C_i, and $C_j \cap C_i = \emptyset$. Since C_j includes top items sorted by $f(\cdot \mid I_j)$ and $v_i \notin C_j$, we have $f(v_j \mid I_j) \geq f(v_i \mid I_j) \geq f(v_i \mid I_i)$, where the second inequality is due to submodularity. □

Proof of Lemma 12. Since $I'_i \subseteq I_i$, submodularity and Lemma 11 imply that

$$\begin{aligned} \mathsf{E}[f(I_i \cup S)] - \mathsf{E}[f(I'_i \cup S)] &\leq \mathsf{E}[f(I_i)] - \mathsf{E}[f(I'_i)] \\ &\leq \epsilon \mathsf{E}[f(I_i)] \leq \epsilon \mathsf{E}[f(I_i \cup S)], \end{aligned}$$

proving the claim. □

B. Streaming algorithm for the RCC problem

A well-known technique developed by Badanidiyuru et al. [9] enables a one-pass streaming algorithm for non-robust submodular maximization. That is, the algorithm is restricted to read items in V only once with very limited memory. The key is to make multiple guesses at the threshold τ^* such that $\tau^* \leq \frac{f(\mathrm{OPT})}{2k} \leq (1+\epsilon)\tau^*$, and then build a candidate solution for each guessed threshold in parallel. The guesses depend on the top singleton encountered so far, and are dynamically updated along the process.

As pointed out in Kazemi et al. [11], a natural extension for the robust setting is to make guesses according to the set of top $d+1$ singletons we have seen so far. To be more specific, we dynamically maintain a set of geometrically-increasing thresholds within range $[\Delta_d/2k, \Delta_d]$, where Δ_d is the value of the current $(d+1)$-th top singleton. Besides, to cope with a static adversary, every item in a candidate solution is randomly sampled from a candidate set of large size.

We adopt the same approach as Kazemi et al. [11], but significantly simplify their algorithm and improve the coreset size, thanks to the importance sampling technique in Lemma 5.

We prove Theorem 3 in the rest of this section, that is, Algorithm 6 and 7 yield $(1-2\epsilon)/2$ approximation guarantee for the RCC problem, with a coreset size $\mathcal{O}((d/\epsilon + k)\log(k)/\epsilon)$. Algorithm 6 constructs the coreset in one-pass using $\mathcal{O}\left(\frac{\log(k)}{\epsilon}\left(n + k\frac{d}{\epsilon}\right)\right)$ queries. Algorithm 7 needs to be slightly modified when receiving the coreset from Algorithm 6. Specifically, at Step 5, I_τ is given directly.

The proof is very similar to that of Theorem 9, except that we replace Lemma 15 and 11 with the following two new lemmas.

Lemma 13. *Let $\tau \in T$ be a threshold, I_τ its associated partial solution, and C_τ candidate set. If $|I_\tau| < k$, then for every item $v \in V \setminus (C_\tau \cup P)$, we have $f(v \mid I_\tau) < \tau$.*

Proof. Select $\tau \in T$ and $v \in V \setminus (C_\tau \cup P)$. Since $v \notin P$, consider the iteration when v is properly processed either due to being a new item or being an item leaving P. Let Δ'_d, C'_τ, I'_τ, and T' be the variables of Algorithm 6 right before the inner for-loop (Step 8).

Assume that $\tau \in T'$. If $f(v \mid I_\tau) \geq \tau$, then $f(v \mid I'_\tau) \geq \tau$ and v is added to C'_τ and never filtered out, that is, $v \in C_\tau$ which is a contradiction. Thus, $f(v \mid I_\tau) < \tau$.

If $\tau \notin T'$, then $\tau > \Delta'_d$ since Δ_d can only increase. Consequently, $f(v \mid I_\tau) \leq f(v) \leq \Delta'_d < \tau$. □

Lemma 14. *For each threshold τ and its associated partial solution I_τ kept by Algorithm 6, we have $\mathsf{E}[f(I'_\tau)] \geq (1-\epsilon)\mathsf{E}[f(I_\tau)]$, where $I'_\tau = I_\tau \setminus D$.*

Proof. We fix an arbitrary threshold τ, and write $I = I_\tau = \{v_1, \ldots, v_i\}$. Let us write $I_j = \{v_1, \ldots, v_{j-1}\}$, and define C_j to the candidate set C_τ from which we sample v_j. Then

$$\begin{aligned} \mathsf{E}[f(I')] &= \mathsf{E}\Big[\sum_{j \leq i} f(v_j \mid I_j \setminus D)\mathbb{1}[v_j \notin D]\Big] \\ &\geq \mathsf{E}\Big[\sum_{j \leq i} f(v_j \mid I_j)\mathbb{1}[v_j \notin D]\Big] \\ &= \mathsf{E}[f(I)] - \sum_{j \leq i} \mathsf{E}\Big[f(v_j \mid I_j)\mathbb{1}[v_j \in D]\Big] \\ &\geq \mathsf{E}[f(I)] - \sum_{j \leq i} \frac{|D|}{|C_j|}\mathsf{E}[f(v_j \mid I_j)] && \triangleright \text{Lemma 5} \\ &\geq \mathsf{E}[f(I)] - \epsilon \sum_{j \leq i} \mathsf{E}[f(v_j \mid I_j)] && \triangleright |C_j| \geq d/\epsilon \\ &= (1-\epsilon)\mathsf{E}[f(I)], \end{aligned}$$

completing the proof. □

Proof of Theorem 3. The proof for approximation is essentially the same as in the proof of Theorem 9, except that Lemma 15 is replaced with Lemma 13 and Lemma 11 is replaced with Lemma 14. We omit the details to avoid repetition.

```
Algorithm 7: Construction of RCC solution after deletion
Input: Coreset and auxiliary information $(R, \{I_j\}_j)$
       returned by Algorithm 4, set of deleted items $D$,
       parameter $\epsilon$
$R' \leftarrow R \setminus D$
$\Delta \leftarrow$ value of the top singleton in $R'$ according to $f(\{v\})$
$T \leftarrow \left\{(1+\epsilon)^i : \frac{\Delta}{2k(1+\epsilon)} \leq (1+\epsilon)^i \leq \Delta, i \in \mathbb{N}\right\}$
for $\tau \in T$ do
    $I_\tau \leftarrow I_{j+1}$, where
      $j \leftarrow \max\{j : f(I_{j+1} \setminus I_j \mid I_j) \geq \tau\}$
    $I'_\tau \leftarrow I_\tau \setminus D$
    for $v \in R'$ do
        if $f(v \mid I'_\tau) \geq \tau$ and $|I'_\tau| < k$ then
            $I'_\tau \leftarrow I'_\tau + v$
return the best solution among $\{I'_\tau\}_{\tau \in T}$
```

We complete the proof by calculating the coreset size. Note that during each iteration $|C_\tau|$ can increase only by 1. If after addition $|C_\tau| > d/\epsilon$, then v is sampled from C_τ and will be filtered out since $f(v \mid I_\tau) = 0$. Thus, in the end $|C_\tau| \leq d/\epsilon$. There are $\mathcal{O}(\log(k)/\epsilon)$ different thresholds, and for each threshold τ we keep at most d/ϵ items in C_τ and a partial solution I_τ of at most size k. Hence, the total coreset size is $\mathcal{O}((d/\epsilon + k)\log(k)/\epsilon)$. □

C. Offline algorithm for the RCC problem

In the case of cardinality constraint, we can obtain a stronger bound than in the general case, by executing a different Algorithm 7 after receiving a coreset from Algorithms 4. Algorithm 7 is inspired by the Sieve algorithm in Badanidiyuru et al. [9], which makes multiple guesses at the threshold τ^* such that $\tau^* \leq \frac{f(\mathrm{OPT})}{2k} \leq (1+\epsilon)\tau^*$, and then builds a candidate solution for each guessed threshold in parallel.

Let us write $I_\tau = I_{i+1}$ and $I'_\tau = I'_{i+1}$, where $i = \max\{j : f(v_j \mid I_j) \geq \tau\}$ is the subset of the tentative solutions built in Algorithms 4 and 7 filtered by a threshold τ. The next step is to show that we do not miss in our coreset any feasible item v with marginal gain $f(v \mid I_\tau) \geq \tau$ among items in V.

Lemma 15. *Assume $\tau > 0$ and let $i = \max\{j : f(v_j \mid I_j) \geq \tau\}$. Shorten $I = I_{i+1}$. Let v be an item that can be added to I with a gain of at least τ, that is, $f(v \mid I) \geq \tau$ and $I + v \in \mathcal{M}$. Let R be the coreset returned by Algorithm 4. Then v is a member of R.*

Proof. If I is the last set in the loop of Algorithm 4, then there is nothing to prove, as by definition there are no items that can be added to I without violating $\mathcal{M}$. Thus, we assume I is not the last set and v_{i+1} and C_{i+1} exist.

By definition of i, we have $f(v_{i+1} \mid I) < \tau$. Since $f(v \mid I) \geq \tau$ either $v \in C_{i+1}$ or v has been removed earlier, that is $v \in C_j$ for $j \leq i$. In either case, v is in R. □

The solutions I'_τ constructed by Algorithm 7 are of form $I'_i \cup S$. By Lemma 12, we already know that $f(I'_i \cup S)$ is close to $f(I_i \cup S)$ in expectation, so we are free to study $I_i \cup S$ instead of $I'_i \cup S$. Next we will bound the former using a thresholding argument. Similar arguments have been used by Badanidiyuru et al. [9] and Kazemi et al. [11].

Lemma 16. *For any τ, write S_τ the additional items added to I'_τ by Algorithm 7. Define $G_\tau = I_\tau \cup S_\tau$. Then*

$$f(G_\tau) \geq \min\{f(\mathrm{OPT}) - k\tau, k\tau\}.$$

Proof. For simplicity, let us shorten G_τ, G'_τ, I_τ, I'_τ, and S_τ with G, G', I, I', and S, respectively. We will prove the lemma by considering three cases.

Case 1: Assume $|I| = k$. Then $f(G) \geq f(I) \geq k\tau$.

Case 2: Assume $|G'| = k$. Then $f(G) \geq f(G') \geq k\tau$.

Case 3: Assume that $|I| < k$ and $|G'| < k$. The main idea is to show that every item in OPT has a marginal gain less than τ with respect to G. Write

$$\begin{aligned} f(\mathrm{OPT}) &\leq f(\mathrm{OPT} \cup G) \leq f(G) + \sum_{v \in \mathrm{OPT} \setminus G} f(v \mid G) \\ &= f(G) + \sum_{v \in (\mathrm{OPT} \setminus G) \cap R} f(v \mid G) + \sum_{v \in (\mathrm{OPT} \setminus G) \setminus R} f(v \mid G). \end{aligned}$$

We discuss the last two terms separately.

To bound the first term, let $v \in (\mathrm{OPT} \setminus G) \cap R$. Since $\mathrm{OPT} \subseteq V \setminus D$, we have $v \in R \setminus D$. If $f(v \mid G) \geq \tau$, then $v \in I'$ or, since $|G'| < k$, the item v is added to S by Algorithm 7. Thus, $f(v \mid G) < \tau$.

To bound the second term, let $v \in (\mathrm{OPT} \setminus G) \setminus R$. Since $|I| < k$, Lemma 15 implies that $\tau > f(v \mid I) \geq f(v \mid G)$.

Since $|\mathrm{OPT}| \leq k$, we have $f(\mathrm{OPT}) \leq f(G) + k\tau$, proving the lemma. □

The next step is to show that there exists $\tau^* \in T$ in thresholds T enumerated by Algorithm 7 such that $\tau^* \leq \frac{f(\mathrm{OPT})}{2k} \leq (1+\epsilon)\tau^*$.

Lemma 17. *Let T be the set of thresholds enumerated by Algorithm 7. There exists $\tau^* \in T$ such that $\tau^* \leq \frac{f(\mathrm{OPT})}{2|\mathrm{OPT}|} \leq (1+\epsilon)\tau^*$.*

Proof. Let $\Delta = \max_{v \in R'} f(v)$ be the value of the top singleton in R'. Note that R contains the top $d+1$ singletons as it contains d top singletons and C_1. Thus $\Delta = \max_{v \in V \setminus D} f(v)$. Consequently, $\Delta \leq f(\mathrm{OPT}) \leq |\mathrm{OPT}|\Delta$.

Since $|\mathrm{OPT}| \leq k$, $\frac{\mathrm{OPT}}{2|\mathrm{OPT}|}$ lies within the range $[\frac{\Delta}{2k}, \Delta]$. An approximately close threshold τ^* can be found by enumeration in an exponential scale with a base $1+\epsilon$. □

Finally, we are ready to prove Theorem 9.

Proof of Theorem 9. Let τ^* be as given by Lemma 17. Write G and G' to be G_{τ^*} and G'_{τ^*} as given in Lemma 16.

Lemmas 17 and 16 state that

$$\begin{aligned} \mathsf{E}[f(G)] &\geq \min\{f(\mathrm{OPT}) - k\tau^*, k\tau^*\} \\ &\geq \min\{f(\mathrm{OPT}) - f(\mathrm{OPT})/2, (1-\epsilon)f(\mathrm{OPT})/2\} \\ &\geq (1-\epsilon)f(\mathrm{OPT})/2. \end{aligned}$$

Lemma 12 states that

$$\begin{aligned}\mathsf{E}[f(\mathrm{ALG})] &\geq \mathsf{E}[f(G')] \geq (1-\epsilon)\mathsf{E}[f(G)] \\ &\geq \frac{(1-\epsilon)^2}{2} f(\mathrm{OPT}) \geq \frac{1-2\epsilon}{2} f(\mathrm{OPT}),\end{aligned}$$

proving the approximation.

The bound on the coreset size is proved in the same way as in Theorem 7. □

D. Further details of experiments

Every algorithm returns the best between its solution and an additional greedy selection over the coreset after deletions. The greedy selection possesses better quality most of the time.

In Figure 1, the error bar at each point is by three random runs.

In Section VI-A, A universe set of 9 724 movies is expanded into a larger set of size 22 046, by considering all movie-genre tuples.