Deletion-Robust Submodular Maximization with Knapsack Constraints

Shuang Cui[1], **Kai Han**[2*], **He Huang**[2]

[1] School of Computer Science and Technology / Suzhou Institute for Advanced Research, University of Science and Technology of China

[2] School of Computer Science and Technology, Soochow University

lakers@mail.ustc.edu.cn, hankai@suda.edu.cn, huangh@suda.edu.cn

Abstract

Submodular maximization algorithms have found wide applications in various fields such as data summarization, recommendation systems, and active learning. In recent years, deletion-robust submodular maximization algorithms have garnered attention due to their significant implications in scenarios where some data points may be removed due to user preferences or privacy concerns, such as in recommendation systems and influence maximization. In this paper, we study the fundamental problem of submodular maximization with knapsack constraints and propose a robust streaming algorithm for it. To the best of our knowledge, our algorithm is the first to solve this problem for non-monotone submodular functions and can achieve an approximation ratio of $1/(6.82 + 2.63d) - \epsilon$ under a near-optimal summary size of $\tilde{\mathcal{O}}(k + r)$, where k denotes the maximum cardinality of any feasible solution, d denotes the number of the knapsack constraints and r is the robustness parameter. For monotone submodular functions, our algorithm can achieve an approximation ratio of $1/(2 + 2d) - \epsilon$ under a near-optimal summary size of $\tilde{\mathcal{O}}(k+r)$, significantly improving upon the best-known ratio of $\Omega\big((1/d-\epsilon)^2\big)$. The empirical performance of our algorithm is extensively evaluated in several applications including influence maximization and recommendation systems, and the experimental results demonstrate the effectiveness of our algorithm.

Introduction

Submodular maximization algorithms, which have played a critical role in advancing the field of artificial intelligence, encompass a broad range of applications, including data summarization (Balkanski, Breuer, and Singer 2018; Amanatidis et al. 2022; Cui et al. 2021, 2023b, 2022, 2023a), object detection (Angelova and Zhu 2013; Zhu, Jiang, and Shao 2014), text classification (Lei et al. 2019), and active learning (Golovin and Krause 2010; Wei, Iyer, and Bilmes 2015; Golovin and Krause 2011). As a result of their widespread applicability, submodular maximization problems have been extensively studied under various constraints, such as cardinality, knapsack, matroid, and independence system constraints. Among the various NP-hard problems studied in this area, submodular maximization with knapsack constraints (abbreviated as the **SK** problem) stands out as one of the most fundamental problems as knapsack constraints can capture real-world constraints such as budget, time, and size. As a result, the SK problem has attracted tremendous attention since the 1980s (Wolsey 1982). In recent years, with the increase in the amount of data, there has been a growing focus on the design of low-memory streaming algorithms for submodular maximization problems (Kazemi et al. 2019; Mitrovic et al. 2017; Badanidiyuru et al. 2014; Mirzasoleiman, Jegelka, and Krause 2018; Haba et al. 2020; El Halabi et al. 2020; Chekuri, Gupta, and Quanrud 2015), which is one of the primary focuses of our paper.

In this paper, we investigate submodular maximization algorithms that are robust to deletions, which have significant implications in scenarios where some data points may be removed by users. For instance, Article 17 of the European "General Data Protection Regulation" (Voigt and Von dem Bussche 2017) outlines obligations for service providers to comply with individuals "Right to be forgotten" allowing them to request the deletion of their personal data or impose restrictions on its use. Another application of these algorithms is in movie recommendation systems, where the goal is to select a subset of movies for a user that maximizes the submodular utility function. However, in the event that a user has already viewed some of the recommended movies, the remaining recommendations may not provide a satisfactory user experience. In such cases, the user may desire for the recommendation system to remove these movies and promptly update the list of recommendations. Nevertheless, rerunning the algorithm on the dataset that excludes movies removed by the user can be both expensive and time-consuming. As such, a more effective approach to handling user deletions would be to enhance the robustness of the algorithm itself.

Motivated by the aforementioned practical needs, several studies such as (Kazemi, Zadimoghaddam, and Karbasi 2018; Dütting et al. 2022b; Zhang, Tatti, and Gionis 2022) have modeled the robustness to deletion as a two-stage game against an adversary and presented robust submodular maximization algorithms. In their model, the robust algorithm generates a summary of the groundset according to the robustness parameter r in the first stage. Subsequently, in the second stage, after revealing the elements removed by the

*Corresponding author: Kai Han

adversary, the algorithm generates the final solution set.[1] They assumed that while the adversary may have knowledge of the algorithm, they do not have access to its random bits and are therefore oblivious to the contents of the summary generated by the algorithm. This is a natural assumption in many real-world applications and is also adopted in this paper. For example, in the movie recommendation application, whether a movie has already been watched by the user or is deemed inappropriate is independent of its inclusion or exclusion in the summary.

Among the existing works on robust algorithms, only (Avdiukhin et al. 2019) has successfully proposed an algorithm for the problem of deletion-robust submodular maximization with knapsack constraints (abbreviated as the **RSK** problem). While they have investigated a more general robust setting where the adversary has access to the random bits utilized by the algorithm, their algorithm is limited to the special scenario in which the submodular function is monotone. More importantly, it has been shown to have a very poor approximation ratio of $\big((1-\frac{1}{e})\frac{3/d+8}{128(1+2d)}-\epsilon\big)^2 = \Omega\big((\frac{1}{d}-\epsilon)^2\big)$, particularly when compared to the best approximation ratio of $\frac{1}{1+2d}-\epsilon$ achieved by (Yu, Xu, and Cui 2018) for this problem in the non-robust scenario (i.e, the monotone SK problem). Although the approximation ratio of (Avdiukhin et al. 2019)'s algorithm can be slightly improved to $\big(\frac{2(1-1/e)(1-1/(2\log B))}{32(1-1/(2\log B))+3}-\epsilon\big)^2$ when there is only a single knapsack constraint (i.e., $d=1$), it is still far from optimal. It remains an open problem whether there exists a robust algorithm that can solve the general (non-monotone) RSK problem in either an offline or streaming setting, and whether it is possible to achieve a good approximation ratio of $\Omega(\frac{1}{d}-\epsilon)$.

Contribution

In this paper, we provide confirmative answers to all of the open problems mentioned above by presenting a novel streaming algorithm dubbed RSK-Streaming for the RSK problem. Our algorithm fundamentally differs from (Avdiukhin et al. 2019)'s in terms of algorithmic design and analysis technique, which are detailedly explained in the fourth section. The major contributions of our paper can be summarized as follows:

- For the non-monotone RSK problem, our RSK-Streaming algorithm can achieve an approximation ratio of $\frac{1}{6.82+2.63d}-\epsilon$ under a near-optimal summary size of $\mathcal{O}(\frac{k\log dB}{\epsilon}+\frac{r\log dB}{\epsilon^2})$, where k denotes the maximum cardinality of any feasible solution and B denotes the budget in the problem. To the best of our knowledge, RSK-Streaming is the first algorithm with provable performance bounds for the non-monotone RSK problem, whether in offline or streaming settings.
- For the monotone RSK problem, our RSK-Streaming algorithm can achieve an approximation ratio of $\frac{1}{2+2d}-\epsilon$ under a near-optimal summary size of $\mathcal{O}(\frac{k\log dB}{\epsilon}+\frac{r\log dB}{\epsilon^2})$, which significantly improves the approximation ratio of $\big((1-\frac{1}{e})\frac{3/d+8}{128(1+2d)}-\epsilon\big)^2$ achieved by the state-of-the-art work (Avdiukhin et al. 2019).
- We conduct extensive experiments using several real-world applications including influence maximization and movie recommendation, and the experimental results strongly demonstrate the effectiveness of our algorithm.

In order to facilitate smooth reading and adhere to page limitations, we have included detailed proofs of most lemmas and theorems in the appendix, while only providing the intuition and underlying ideas of the proofs in the main text.

Related Work

In this section, we provide a review of the two algorithms that are most closely related to our research topic.

Algorithms for Submodular Maximization With Knapsack Constraints

The traditional SK problem has a long history of research, with the initial proposal of an algorithm by (Wolsey 1982) that achieved an approximation ratio of 0.357 for the special case where $d=1$ and the objective function is monotone. Subsequent work, such as that by (Lee et al. 2010; Gupta et al. 2010; Fadaei, Fazli, and Safari 2011; Kulik, Shachnai, and Tamir 2013; Khuller, Moss, and Naor 1999; Badanidiyuru and Vondrák 2014; Ene and Nguyen 2019; Amanatidis et al. 2020), focused on the development of offline algorithms to address both monotone and non-monotone submodular functions. In the general case where the submodular function is non-monotone, the optimal approximation ratio of 0.385 was achieved by (Buchbinder and Feldman 2019). In recent years, there has been significant interest in the streaming algorithm of the SK problem, with studies such as (Han et al. 2021; Yaroslavtsev, Zhou, and Avdiukhin 2020; Huang and Kakimura 2018; Huang, Kakimura, and Yoshida 2020; Huang and Kakimura 2021; Yu, Xu, and Cui 2018) examining the case where the submodular function is monotone and achieving an approximation ratio of $\frac{1}{1+2d}-\epsilon$. For the case where the submodular function is non-monotone, (Han et al. 2021) and (Cui et al. 2022) have proposed approximation algorithms for it, and the optimal approximation ratio is $\frac{1}{3.6+2.1d}$.

Algorithms for Deletion-Robust Submodular Maximizaton

The study of deletion-robust submodular maximization was first introduced by (Krause et al. 2008), who considered a setting that required the algorithm to construct a solution without allowing it to update the output after deletion, which is different from the one we studied. Subsequent work including (Bogunovic et al. 2017) and (Orlin, Schulz, and Udwani 2018) also adopted this setting. The robust setting we consider in this paper is first proposed by (Mitrovic et al. 2017) and has been adopted by subsequent work including (Kazemi, Zadimoghaddam, and Karbasi 2018; Mirzasoleiman, Karbasi, and Krause 2017; Avdiukhin et al. 2019; Zhang, Tatti, and Gionis 2022; Dütting et al. 2022a),

[1] Formal problem definitions are provided in the third section.

where the algorithms proposed by (Mitrovic et al. 2017) and (Avdiukhin et al. 2019) are capable of handling stronger cases in which the adversary can select removed elements adaptively with respect to the summary produced by the algorithm. Among the studies mentioned above, (Kazemi, Zadimoghaddam, and Karbasi 2018) stand out as a significant contribution to the problem of deletion-robust submodular maximization with a simple cardinality constraint. Their algorithm employs random sampling to pick elements from a pool of candidate elements, an idea that is later adopted by (Zhang, Tatti, and Gionis 2022; Dütting et al. 2022b,a) and provids some inspiration for enhancing the robustness of our algorithm. For the RSK problem we focus on in this paper, only (Avdiukhin et al. 2019) have successfully designed a robust algorithm, based on the partitioning approach proposed by (Mitrovic et al. 2017). However, their algorithm only works for the special case where the submodular function is monotone and yield a poor approximation ratio of $\left((1-\frac{1}{e})\frac{3/d+8}{128(1+2d)}-\epsilon\right)^2$, which is far from optimal.

Problem Statement

We begin by providing some fundamental definitions:

Definition 1 (Submodular Function)**.** *A set function $f\colon 2^{\mathcal{N}} \mapsto \mathbb{R}_{\geq 0}$ defined on a finite ground set $\mathcal{N}$ with $|\mathcal{N}| = n$ is considered submodular if it satisfies the following condition for all sets: $f(X)+f(Y) \geq f(X \cup Y)+f(X \cap Y)$.*

In this paper, we allow $f(\cdot)$ to be non-monotone, i.e., $\exists X \subseteq Y \subseteq \mathcal{N}\colon f(X) > f(Y)$. Following existing work such as (Dütting et al. 2022b; El Halabi et al. 2020; Mitrovic et al. 2017), we assume that $f(\cdot)$ is normalized, i.e., $f(\emptyset)=0$. We assume that each element $u \in \mathcal{N}$ has d associated weights $c_1(u), c_2(u), \ldots, c_d(u)$ and define the function $c_z(S)=\sum_{u\in S} c_z(u)$ for any $S \subseteq \mathcal{N}$. To facilitate comparison with existing work, we follow (Yaroslavtsev, Zhou, and Avdiukhin 2020; Han et al. 2021; Avdiukhin et al. 2019; Cui et al. 2022) to assume that the weights of elements in $\mathcal{N}$ are normalized such that $\forall u \in \mathcal{N}, \forall z \in [d]\colon c_z(u) \geq 1$.

Based on the above definitions, the traditional submodular maximization with knapsack constraints (i.e., the SK problem) is defined as: $\max\{f(S)\colon S \subseteq \mathcal{N}$ and $\forall\, z \in [d], c_z(S) \leq B\}$. In this paper, we study the deletion-robust version of the SK problem (i.e., the RSK problem), which involves solving the SK problem for each possible deletion instance of $R \subseteq \mathcal{N}$, defined as follows:$\max\{f(S)\colon S \subseteq \mathcal{N}\setminus R$ and $\forall\, z \in [d], c_z(S) \leq B\}$, where R with size of $|R| \leq r$ is unknown a priori and selected by an adversary that is unaware of the random bits used by the algorithm. Following (Kazemi, Zadimoghaddam, and Karbasi 2018; Dütting et al. 2022b; Zhang, Tatti, and Gionis 2022), we model the procedure for finding a solution that is robust to deletions as a two-stage game against an adversary. In the first stage, the algorithm is provided with a robustness parameter r and selects a subset of $\mathcal{N}$ as a summary of the entire dataset $\mathcal{N}$. Simultaneously, an adversary, who may know the algorithm but has no access to its random bits, selects a deletion set R. In the second stage, the adversary reveals R and the algorithm determines a feasible solution from the summary after removing R. In this model, the algorithm's performance is evaluated based on its approximation ratio and the size of its summary. We investigate the RSK problem within the context of a streaming setting, where elements in $\mathcal{N}$ arrive sequentially in an arbitrary order. Our objective is to develop a single-pass streaming algorithm that utilizes a small amount of memory, with the memory usage being nearly linear in relation to the sum of k and r.

For convenience, we define $f(u)=f(\{u\})$ for any element $u \in \mathcal{N}$, and the marginal gain of u with respect to S as $f(u \mid S)=f(S\cup\{u\})-f(S)$. We denote the set $\{1,\ldots,i\}$ as $[i]$ for any natural number i, and the maximum cardinality of any feasible solution in the problem as k. Throughout this paper, we use O to denote an optimal solution for the RSK problem. Without loss of generality, we assume that $\forall z \in [d]\colon c_z(u) \leq B$ for every $u \in \mathcal{N}$, as any element violating the assumption can be removed from the ground set without affecting any feasible solution.

Algorithms

To the best of our knowledge, only (Avdiukhin et al. 2019) have proposed an algorithm for the RSK problem. Their algorithm is limited to solving the special case where the submodular function is monotone, whereas our algorithm does not have such limitations. Furthermore, our algorithm fundamentally differs from theirs in terms of algorithmic design, as mainly reflected in the following key aspects. (i) Drawing on the partitions-and-buckets approach in (Bogunovic et al. 2017; Mitrovic et al. 2017), the algorithm proposed by (Avdiukhin et al. 2019) creates multiple partitions, each of which corresponds to a specific threshold of marginal density, as well as multiple buckets that are dynamically added as needed. These partitions are characterized by exponentially decreasing thresholds, while the bucket capacity exhibits exponential growth. Each incoming element in the stream is added to the first bucket that satisfies its threshold and capacity, which ensures that the product of cost and marginal density of each element in the bucket is nearly identical, thereby enhancing the robustness of the summary. In contrast, our algorithm employs a single threshold $\gamma^* \in \left[\frac{f(O)}{(1+\epsilon)\alpha B}, \frac{f(O)}{\alpha B}\right]$, and a fixed number of candidate solutions to control the quality of elements added to the summary. Each candidate solution has a capacity of at most the budget B and includes a "warehouse" for temporarily storing high-quality elements. As new elements arrive, the algorithm greedily selects the candidate solution with the maximum marginal density and adds the element to its warehouse if its marginal density satisfies the threshold. Once the number of elements in a warehouse exceeds a certain number, we sample an element from the warehouse and add it to the corresponding candidate solution if it satisfies the budget constraint. The above process ensures both the quality and robustness of the summary, which will be explained in detail later. (ii) Their algorithm necessitates reordering the initial output summary based on the cost of elements and invoking the algorithm again to ensure that the final summary meets the size requirements, while our algorithm can directly generate a summary with satisfactory size. (iii) Unlike their al-

Algorithm 1: RSK-Streaming-Summary

Input: budget B, $\epsilon \in (0,1)$, deletion parameter r, positive number $\alpha > d$, and integer h;

1 $e_{[r]} \leftarrow$NULL, $V_{[r]} \leftarrow$ the first r elements to arrive;
2 **while** there is an incoming element e **do**
3 $\quad V'_{[r]} \leftarrow$ the $r+1$ elements with the highest function values among the arrived elements;
4 $\quad$ **if** $\frac{f(e)}{\sum_{z\in[d]} c_z(e)} > \min_{a\in V_{[r]}} \frac{f(a)}{\sum_{z\in[d]} c_z(a)}$ **then**
5 $\quad\quad$ Add e to $V_{[r]}$ and then pop the element u with the smallest value of $\frac{f(u)}{\sum_{z\in[d]} c_z(u)}$;
6 $\quad$ **else** $u \leftarrow e$;
7 $\quad$ **if** $\frac{f(u)}{\sum_{z\in[d]} c_z(u)} > \frac{f(e_{[r]})}{\sum_{z\in[d]} c_z(e_{[r]})}$ **then** $e_{[r]} \leftarrow u$;
8 $\quad \Gamma \leftarrow \big\{(1+\epsilon)^i : i \in \mathbb{Z} \wedge \frac{f(e_{[r]})}{(1+\epsilon)\alpha B \sum_{z\in[d]} c_z(e_{[r]})} \le (1+\epsilon)^i \le \frac{(1+\epsilon) f(e_{[r]})}{\sum_{z\in[d]} c_z(e_{[r]})}\big\}$;
9 $\quad$ Remove all S_t^γ and W_t^γ where $\gamma \notin \Gamma$;
10 $\quad$ **foreach** $t \in [h]$ and $\gamma \in \Gamma$ **do**
11 $\quad\quad$ **if** Neither S_t^γ nor W_t^γ exist **then**
12 $\quad\quad\quad S_t^\gamma \leftarrow \emptyset$; $W_t^\gamma \leftarrow \emptyset$
13 $\quad$ **foreach** $\gamma \in \Gamma$ **do**
14 $\quad\quad x \leftarrow \arg\max_{t\in[h]} \frac{f(u|S_t^\gamma)}{\sum_{z\in[d]} c_z(u)}$;
15 $\quad\quad$ **if** $\frac{f(u|S_x^\gamma)}{\sum_{z\in[d]} c_z(u)} \ge \gamma$ **then** Add u to W_x^γ;
16 $\quad\quad$ **while** $\exists y \in [h] : |W_y^\gamma| \ge \frac{r}{\epsilon}$ **do**
17 $\quad\quad\quad$ Randomly select and remove an element v^* from W_y^γ with a selection probability of $1/\big(f(v \mid S_y^\gamma) \cdot \sum_{w\in W_y^\gamma} f(w \mid S_y^\gamma)^{-1}\big)$ for each $v \in W_y^\gamma$;
18 $\quad\quad\quad$ **if** $\forall z \in [d] : c_z(S_y^\gamma \cup \{v^*\}) \le B$ **then**
19 $\quad\quad\quad\quad S_y^\gamma \leftarrow S_y^\gamma \cup \{v^*\}$, and reassign all elements in W_y^γ to appropriate sets based on the rules specified in Lines 14-15;
20 $W \leftarrow \bigcup_{t\in[h],\gamma\in\Gamma} W_t^\gamma \cup V_{[r]} \cup V'_{[r]} \cup e_{[r]}$;
21 **return** $\{S_t^\gamma : \gamma \in \Gamma, t \in [h]\}, W$

Algorithm 2: RSK-Streaming-Mining

Input: The set R of removed elements, and the summary returned by RSK-Streaming-Summary including $\{S_t^\gamma : \gamma \in \Gamma, t \in [h]\}, W$

1 $e_{max} \leftarrow \arg\max_{a\in W\setminus R} \frac{f(a)}{\sum_{z\in[d]} c_z(a)}$;
2 $\Gamma' \leftarrow \big\{(1+\epsilon)^i : i \in \mathbb{Z} \wedge \frac{f(e_{max})}{(1+\epsilon)\alpha B \sum_{z\in[d]} c_z(e_{max})} \le (1+\epsilon)^i \le \frac{(1+\epsilon) f(e_{max})}{\sum_{z\in[d]} c_z(e_{max})}\big\}$;
3 Remove all S_t^γ where $\gamma \notin \Gamma'$;
4 **foreach** $t \in [h]$ and $\gamma \in \Gamma'$ **do**
5 $\quad$ **if** S_t^γ does not exist **then** $S_t^\gamma \leftarrow \emptyset$;
6 $U^* \leftarrow \mathsf{Offline}\big((\bigcup_{\gamma\in\Gamma', t\in[h]} S_t^\gamma \cup W) \setminus R\big)$;
7 $S^* \leftarrow \arg\max_{A\in\{S_t^\gamma \setminus R : \gamma\in\Gamma', t\in[h]\}} f(A)$;
8 $e^* \leftarrow \arg\max_{a\in W\setminus R} f(a)$;
9 **if** $f(\cdot)$ is monotone **then**
10 $\quad$ **for** $\gamma \in \Gamma'$ **do**
11 $\quad\quad T^\gamma \leftarrow S_1^\gamma \setminus R$;
12 $\quad\quad$ **for** $e \in W \setminus R$ **do**
13 $\quad\quad\quad$ **if** $\frac{f(e|T^\gamma)}{\sum_{z\in[d]} c_z(e)} \ge \gamma$ **then**
14 $\quad\quad\quad\quad$ **if** $\forall z \in [d] : c_z(T^\gamma \cup \{e\}) \le B$ **then**
15 $\quad\quad\quad\quad\quad T^\gamma \leftarrow T^\gamma \cup \{e\}$
16 $\quad T^* \leftarrow \arg\max_{A\in\{T^\gamma : \gamma\in\Gamma'\}} f(A)$;
17 **return** $X^* \leftarrow \arg\max_{A\in\{S^*,\, T^*,\, U^*,\, \{e^*\}\}} f(A)$

gorithm, which calls the offline algorithm on the summary to obtain the final solution, our algorithm not only uses the summary as the groundset to call the offline algorithm to obtain a candidate solution, but also attempts to further improve the approximation ratio of the algorithm by adding elements that meet the threshold and budget requirements to each existing candidate solution when the objective function is monotone. In a nutshell, our algorithm is simple yet effective, which can handle the general RSK problem efficiently.

Algorithm Design

In this section, we present our two-stage algorithm for the RSK problem, as shown by Algorithms 1-2. Overall, given that the value of $f(O)$ is unknown, Algorithm 1 employs the element $e_{[r]}$, which possesses the $(r+1)$-th highest density among the arrived elements, to guess the ideal threshold $\gamma^* \in \left[\frac{f(O)}{(1+\epsilon)\alpha B}, \frac{f(O)}{\alpha B}\right]$ used for filtering high-quality elements (Lines 1-12). Furthermore, in order to ensure that Algorithm 1 consistently utilizes a minimal amount of memory, unnecessary thresholds and their corresponding sets are removed in a timely manner (Line 9). In addition to this, Algorithm 1 maintains two sets, $V_{[r]}$ and $V'_{[r]}$, which store the r elements with the highest density (i.e., function value per cost) and the $r+1$ elements with the highest function value in the stream (Lines 1-5), respectively. These two sets ultimately become part of the summary returned by the algorithm. For each threshold $\gamma \in \Gamma$, we maintain h disjoint candidate solutions, denoted as $S_1^\gamma, \cdots, S_h^\gamma$, and each of them has an associated warehouse W_i^γ that stores high-quality elements for S_i^γ. When a new element u needs to be processed, Algorithm 1 greedily finds the candidate solution $S_x^\gamma \in \{S_1^\gamma, \cdots, S_h^\gamma\}$ that maximizes the marginal density $\frac{f(u|S_x^\gamma)}{\sum_{z\in[d]} c_z(u)}$ and adds the element to the warehouse W_x^γ associated with the selected candidate solution if the marginal density satisfies the threshold γ (Lines 14-15). Once a warehouse W_y^γ with sufficient high-quality elements is identified, Algorithm 1 randomly selects an element from the warehouse and adds the element to the candidate solution S_y^γ without violating the knapsack constraints and then reconstruct all warehouses, where the probability of each element

being selected is inversely proportional to its marginal gain (Lines 17-19). The initution behind the above approach to managing elements can be elucidated as follows:

- Since the marginal density of all elements in the warehouse exceeds the threshold, selecting any one of them would yield sufficient utility. In addition, our selection process follows the principle that elements with smaller marginal gain are more likely to be selected, so as to ensure that the marginal gain of each element in the candidate solution is similar in expectation, which means that the loss caused by removing an element in the candidate solution is equal to the gain of an element retained in the candidate solution, preventing unbounded losses. Concurrently, by randomly selecting elements from r/ϵ elements in the warehouse, we effectively reduce the probability of any element in the candidate solution being deleted by the adversary. All the above conditions ensure the high quality and robustness of the summary returned by the algorithm.
- The algorithm maintains h candidate solutions, wherein an element can only be added to the candidate solution with the maximum marginal density in order to ensure that these solutions are disjoint. This is done to establish a relationship between the candidate solutions and the optimal solution when the objective function is non-monotone, as detailed in Lemma 4.

Algorithm 1 returns $V_{[r]}$, $V'_{[r]}$, all candidate solutions and warehouses as a summary at the end. This summary is received by Algorithm 2 after the adversary reveals its removed elements set R, and Algorithm 2 then starts generating the final solution from the summary after removing R. Specifically, Algorithm 2 guesses the optimal threshold using the latest information and removes unnecessary sets (Lines 1-5). Then, Algorithm 2 obtains a candidate solution by invoking an offline algorithm using the summary after removing elements in R as the input set, where the offline algorithm can achieve an approximation ratio of β for the SK problem (Line 6). In cases where the objective function is monotone, the algorithm additionally performs a simple operation to enhance the performance of existing candidate solutions by adding the elements in the summary that meet the threshold and budget requirements to each candidate solution (Lines 9-16). Finally, Algorithm 2 returns the set with the highest objective function value among all candidate solutions and singleton element sets as the final solution.

Theoretical Analysis

Before everything starts, it is imperative that we establish the efficacy of our threshold guessing method in identifying the ideal threshold $\gamma^* \in \left[\frac{f(O)}{(1+\epsilon)\alpha B}, \frac{f(O)}{\alpha B}\right]$. As shown by Lemma 1, in Algorithm 1, the generation of candidate solutions corresponding to the ideal threshold must precede the arrival of elements in the data stream whose marginal density satisfies the threshold requirement. In the event that the number of elements meeting the ideal threshold requirement is limited (i.e., no more than r), these elements can be directly incorporated into the summary and subsequently processed in the algorithm of the second stage (Algorithm 2). As shown by Lemma 2, given that Algorithm 2 is aware of the set R comprising the removed elements, we are able to find the ideal threshold and subsequently create the corresponding candidate solutions.

Lemma 1. *Upon termination of RSK-Streaming-Summary, a series of candidate solutions $\{S_t^{\gamma^*} : t \in [h]\}$ is created; otherwise, all elements with a density greater than γ^* belonging to the set $V_{[r]}$.*

Proof. According to Lines 8-12 of RSK-Streaming-Summary, the algorithm creates a set S_t^{γ} with a γ value that satisfies the inequality

$$\frac{f(e_{[r]})}{(1+\epsilon)\alpha B \sum_{z\in[d]} c_z(e_{[r]})} \le \gamma \le \frac{(1+\epsilon) f(e_{[r]})}{\sum_{z\in[d]} c_z(e_{[r]})}. \quad (1)$$

Therefore, we attempt to prove this lemma by demonstrating that γ^* satisfies this inequality when RSK-Streaming-Summary terminates.

Let $e^* = \arg\max_{u\in\mathcal{N}\setminus R} \frac{f(u)}{\sum_{z\in[d]} c_z(u)}$. Then we can get $f(O) \ge f(e^*) \ge \frac{f(e^*)}{\sum_{z\in[d]} c_z(e^*)} \ge \frac{f(e_{[r]})}{\sum_{z\in[d]} c_z(e_{[r]})}$ due to the fact that $e_{[r]}$ has the $(r+1)$-th largest density in $\mathcal{N}$, which proves γ^* satisfies the left-hand side of Eqn. (1). Subsequently, we discuss the right-hand side of Eqn. (1), which necessitates dividing our analysis into two cases:

- $\frac{f(e_{[r]})}{\sum_{z\in[d]} c_z(e_{[r]})} \ge \frac{f(O)}{(1+\epsilon)\alpha B}$. In this case, we denote v^* as the first element not in $V_{[r]}$ but with a density greater than $\frac{f(O)}{(1+\epsilon)\alpha B}$. Then we have $\frac{f(O)}{\alpha B} \le \frac{(1+\epsilon) f(v^*)}{\sum_{z\in[d]} c_z(v^*)}$, which implies γ^* must satisfy the right-hand side of Eqn. (1), and hence the candidate solutions $\{S_t^{\gamma^*} : t \in [h]\}$ will be created when v^* arrives according to Lines 8-12 of Algorithm 1.
- $\frac{f(e_{[r]})}{\sum_{z\in[d]} c_z(e_{[r]})} < \frac{f(O)}{(1+\epsilon)\alpha B}$. In this case, according to the definition of $e_{[r]}$, all elements with density greater than γ^* must be stored in $V_{[r]}$.

Combining the above cases, the lemma follows. □

Lemma 2. *After executing Lines 4-5 of RSK-Streaming-Mining, a series of candidate solutions $\{S_t^{\gamma^*} : t \in [h]\}$ must have been created.*

Proof. According to Line 1 of RSK-Streaming-Mining, $f(e_{max}) = \max_{u\in\mathcal{N}\setminus R} \frac{f(u)}{\sum_{z\in[d]} c_z(u)}$, so we must have $\frac{f(e_{max})}{\max_{z\in[d]} c(e_{max})} \ge \frac{f(O)}{\alpha B}$, else we can get

$$f(O) \le \sum_{u\in O} f(u) = \sum_{u\in O} \Big(\frac{f(u)}{\sum_{z\in[d]} c(u)} \cdot \sum_{z\in[d]} c(u) \Big)$$
$$< \sum_{u\in O} \Big(\frac{f(O)}{\alpha B} \cdot \sum_{z\in[d]} c(u) \Big) \le \frac{d \cdot f(O)}{\alpha},$$

which means $d > \alpha$, a contradiction. Combining the facts that $f(O) \ge f(e_{max}) \ge \frac{f(e_{max})}{\sum_{z\in[d]} c_z(e_{max})}$, we complete the proof. □

Based on the ideal threshold γ^* and the sets associated with it, we proceed with our analysis. In accordance with the knapsack constraints, the analysis can be categorized into two distinct cases, based on whether or not an element with a marginal density of at least γ^* is discarded by Algorithm 1 due to a budget violation (Line 18). Our first focus is on the case described in Lemma 3, wherein a candidate solution with a sufficiently large cost has been identified and each element in the solution possesses a sufficiently large density (i.e., the ratio of its marginal gain to cost is no less than γ^*). Consequently, this candidate solution can be utilized as a reliable upper bound of the optimal solution O due to the fact that $\gamma^* \geq \frac{f(O)}{(1+\epsilon)\alpha B}$, which can help us to obtain an approximation ratio for the algorithms in this case.

Lemma 3. *If the judgment in Line 18 of RSK-Streaming-Summary has failed for a candidate solution $S_q^{\gamma^*}(q \in [h])$ and an element $e \in \mathcal{N} \backslash R$, the solution X^* returned by RSK-Streaming-Mining satisfies $f(S_q^{\gamma^*}) + f(X^*) \geq \frac{f(O)}{(1+\epsilon)\alpha}$.*

Then we analyse the approximation ratio for the case described by Lemma 4, by establishing an upper bound on the utility loss caused by elements present in the optimal solution but absent from the candidate solutions maintained by our algorithm. We first consider the elements that belong to the optimal solution but are not present in the candidate solutions $S_1^{\gamma^*}, \cdots, S_h^{\gamma^*}$ and their corresponding warehouses $W_1^{\gamma^*}, \cdots, W_h^{\gamma^*}$. The exclusion of these elements is not due to the knapsack constraints in this case, which implies that these elements must have a marginal density lower than γ^*, and thus the utility loss caused by these elements can be bounded by γ^*. Furthermore, high-quality elements are distributed among different candidate solutions rather than being concentrated in a single solution, resulting in some loss of utility, and unused high-quality elements in the warehouse also contribute to utility loss. The above two types of utility loss can be bounded by the offline algorithm called by Algorithm 2 (Line 6). Based on the above analysis, we can get an upper bound on the function value of the optimal solution in this case, as shown by Lemma 4.

Lemma 4. *If the situation described by Lemma 3 never happens, the solution X^* returned by RSK-Streaming-Mining satisfies $\beta \cdot h^{-1} \sum_{t\in[h]} f(S_t^{\gamma^*}) + f(X^*) \geq \beta(1-\frac{1}{h}-\frac{d}{\alpha})f(O)$.*

With Lemmas 3-4, we are only a few steps away from obtaining the approximation ratio of the algorithm, that is, bounding the function value of $S_t^{\gamma^*}$ $(t \in [h])$ with the solution X^* returned by our algorithm, where $S_t^{\gamma^*}$ is the candidate solution that has not had elements removed by the adversary. Note that the candidate solution after the removal of elements selected by the adversary (i.e., $S_t^{\gamma^*} \setminus R$) remains comparable to the original solution $S_t^{\gamma^*}$, which can be explained as follows. Any element in $S_t^{\gamma^*}$ is selected at random from r/ϵ elements, resulting in a low probability that an element will be removed by the adversary. Furthermore, our selection process adheres to the principle that elements with smaller marginal gain are more likely to be selected, which ensures that the marginal gain of each element in the candidate solution is similar in expectation. Consequently, the loss incurred by removing an element from the candidate solution can be bounded by the gain of an element retained within the candidate solution. Therefore, we can establish an upper bound for $f(S_t^{\gamma^*})$ using $f(S_t^{\gamma^*} \setminus R)$, as shown by Lemma 5.

Lemma 5. *For any $t \in [h]$, we have $\mathbb{E}[f(S_t^{\gamma^*} \setminus R)] \geq (1-\epsilon)\mathbb{E}[f(S_t^{\gamma^*})]$.*

By synthesizing the results of Lemmas 3-5, we derive the approximation ratio for our algorithm, as demonstrated in Theorem 1. Additionally, the summary size of our approach is presented in Theorem 2.

Theorem 1. *By setting $\alpha = 11(1/\beta + d)/10, h = 11$, Algorithms 1-2 achieve an approximation ratio of $\frac{1}{1.1(1+2/\beta+(2+\beta)d)} - \epsilon$ for the non-monotone RSK problem, where β is the approximation ratio of any offline algorithm for the non-monotone SK problem.*

Theorem 2. *The size of the summary returned by RSK-Streaming-Summary is $\mathcal{O}(\frac{k \log dB}{\epsilon} + \frac{r \log dB}{\epsilon^2})$.*

By utilizing the algorithm (Buchbinder and Feldman 2019) that achieves a 0.385-approximation for the non-monotone SK problem as the offline sub-algorithm for RSK-Streaming, we are able to derive the following corollary.

Corollary 1. *There exists an algorithm that can achieve an approximation ratio of $\frac{1}{6.82+2.63d} - \epsilon$ for the non-monotone RSK problem.*

Improved Approximation Ratio for the Monotone RSK Problem

By leveraging the monotonicity of the objective function in the monotone RSK problem, we can obtain stronger versions of Lemma 3 and Lemma 4, as shown by Lemma 6 and Lemma 7. These results lead to an improved approximation ratio for our algorithm, as shown by Theorem 3.

Lemma 6. *If the judgment in Line 18 of RSK-Streaming-Summary or Line 14 of RSK-Streaming-Mining has failed, the candidate solutions created by the two algorithms satisfy $f(S_1^{\gamma^*} \cup T^{\gamma^*}) + f(X^*) \geq \frac{f(O)}{(1+\epsilon)\alpha}$.*

Lemma 7. *If the situation described by Lemma 6 never happens, the solution X^* returned by RSK-Streaming-Mining satisfies $f(S_1^{\gamma^*} \cup T^{\gamma^*}) + f(X^*) \geq (1 - \frac{d}{\alpha})f(O)$.*

Theorem 3. *By setting $\alpha = 1+d$ and $h = 1$, Algorithms 1-2 achieve an approximation ratio of $\frac{1}{2+2d} - \epsilon$ for the monotone RSK problem.*

Experiments

In this section, we compare our streaming algorithm RSK-Streaming with state-of-the-art streaming algorithms for solving the RSK-Streaming problem, on the real-world application. Specifically, we implement three algorithms: (1) Our RSK-Streaming algorithm (Algorithm 1-2), abbreviated as "RSKS"; (2) AlgMult (Avdiukhin et al. 2019), which to our knowledge is the only existing robust algorithm capable of solving the (monotone) RSK problem

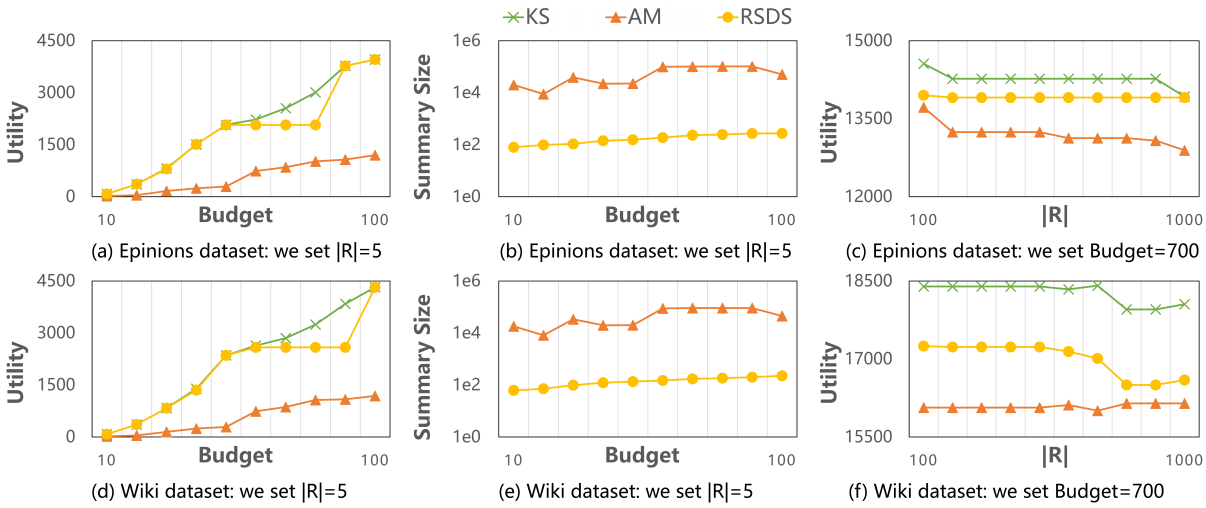

Figure 1: Experiment results on the influence maximization application.

with a provable approximation ratio and is abbreviated as "AM"; and (3) KnapsackStreaming (Cui et al. 2022), which is the state-of-the-art algorithm for solving the SK problem and is abbreviated as "KS". For the purposes of comparison, we assume that KS is omniscient and is thus aware of the removed element in advance, which allows KS to effectively solve the RSK problem and serve as a strong comparison for our algorithm. In our experiments, we utilize the objective function value (i.e., utility) as the primary metric, and also evaluate the summary size of our algorithm against that of another robust algorithm (i.e., AlgMult). Following the approach of previous work on robust submodular optimization (Dütting et al. 2022b; Kazemi, Zadimoghaddam, and Karbasi 2018; Zhang, Tatti, and Gionis 2022), the state-of-the-art practical algorithm for the SK problem (i.e., the SMMK (Han et al. 2021) algorithm) is utilized to generate the set R of removed elements and serves as the sub-algorithm called by any algorithm that requires an offline sub-algorithm, and we set $\epsilon = 0.5$ to avoid large summary. Each randomized algorithm is executed independently five times and the average result is reported, and all experiments are conducted on a Linux server equipped with an Intel Xeon Gold 6126 @ 2.60GHz CPU and 256GB memory. The implemented algorithms are evaluated in two real-world applications, including robust influence maximization and interactive movie recommendation. The experimental settings and results pertaining to the former are presented in the following, while those corresponding to the latter are relegated to appendix due to constraints on page length.

Robust Influence Maximization

The application is also consider in several work on robust submodular maximization (Dütting et al. 2022b; Zhang, Tatti, and Gionis 2022; Avdiukhin et al. 2019). Given a network $G = (\mathcal{N}, E)$, our goal is to identify a subset of seed nodes $S \subseteq \mathcal{N}$ that can influence a large number of users within a given budget B. This objective can be formally expressed:

$$\max\{f(S) = |\cup_{u \in S} N(u)| : c(S) \leq B\},$$

where $N(u) = \{v : (u, v) \in E\}$ denotes the neighbors of u. It has been indicated in previous research (Dütting et al. 2022b; Zhang, Tatti, and Gionis 2022; Avdiukhin et al. 2019) that $f(\cdot)$ is a monotone submodular fucntion. Following (Kazemi et al. 2021; Harshaw et al. 2019), each node $u \in \mathcal{N}$ is associated with a non-negative cost $c(\{u\}) = 1 + \sqrt{d(u)}$, where $d(u)$ represents the out-degree of u. For our experiments, we utilize two network datasets: (1) epinions (Leskovec and Krevl 2014) with 131,828 nodes and 841,372 edges; and (2) wiki (Rossi and Ahmed 2015) with 138,592 nodes and 740,397 edges.

Experimental Results

Figure 1 (a)-(b) and Figure 1 (d)-(e) show the results when the number of removed elements (i.e., $|R|$) is 5 and budget varies. It can be observed from Figure 1 (a) and Figure 1 (d) that our RSKS algorithm achieves significantly better utility than AM, with performance gains of up to 350% (and 299% on average). Notably, AM is the only existing robust algorithm for the RSK problem. Furthermore, our RSKS algorithm can achieve comparable utility to the omniscient algorithm SMMK, which knows the removed elements in advance, further demonstrating the effectiveness of our algorithm on utility. Meanwhile, as shown in Figure 1 (b) and Figure 1 (e), the summary generated by RSKS stores fewer elements than that generated by AM in more than two orders of magnitude, which demonstrates the superiority of our algorithm in terms of efficiency and memory consumption. In Figure 1 (c) and Figure 1 (f), we scale the number of removed elements (i.e., $|R|$) to compare the utility of the implemented algorithms, which shows that the utility achieved by our RSKS algorithm decreases little with the increase of $|R|$ (up to a maximum of 6%), demonstrating the strong robustness of our algorithm.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62172384, Grant No. 62332013, and Grant No. U20A20182. The work of Shuang Cui is done under the guidance of his supervisor: Kai Han.

References

Amanatidis, G.; Fusco, F.; Lazos, P.; Leonardi, S.; and Reiffenhäuser, R. 2020. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Amanatidis, G.; Fusco, F.; Lazos, P.; Leonardi, S.; and Reiffenhäuser, R. 2022. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. *Journal of Artificial Intelligence Research (JAIR)*, 74: 661–690.

Angelova, A.; and Zhu, S. 2013. Efficient object detection and segmentation for fine-grained recognition. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 811–818.

Avdiukhin, D.; Mitrović, S.; Yaroslavtsev, G.; and Zhou, S. 2019. Adversarially robust submodular maximization under knapsack constraints. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 148–156.

Badanidiyuru, A.; Mirzasoleiman, B.; Karbasi, A.; and Krause, A. 2014. Streaming submodular maximization: Massive data summarization on the fly. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 671–680.

Badanidiyuru, A.; and Vondrák, J. 2014. Fast algorithms for maximizing submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1497–1514.

Balkanski, E.; Breuer, A.; and Singer, Y. 2018. Non-monotone submodular maximization in exponentially fewer iterations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2359–2370.

Bogunovic, I.; Mitrović, S.; Scarlett, J.; and Cevher, V. 2017. Robust submodular maximization: A non-uniform partitioning approach. In *International Conference on Machine Learning (ICML)*, 508–516.

Buchbinder, N.; and Feldman, M. 2019. Constrained submodular maximization via a nonsymmetric technique. *Mathematics of Operations Research*, 44(3): 988–1005.

Chekuri, C.; Gupta, S.; and Quanrud, K. 2015. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 318–330.

Cui, S.; Han, K.; Tang, J.; and Huang, H. 2023a. Constrained Subset Selection from Data Streams for Profit Maximization. In *International World Wide Web Conferences (WWW)*, 1822–1831.

Cui, S.; Han, K.; Tang, J.; Huang, H.; Li, X.; and Li, Z. 2022. Streaming Algorithms for Constrained Submodular Maximization. In *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*.

Cui, S.; Han, K.; Tang, J.; Huang, H.; Li, X.; and Zhiyuli, A. 2023b. Practical parallel algorithms for submodular maximization subject to a knapsack constraint with nearly optimal adaptivity. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Cui, S.; Han, K.; Zhu, T.; Tang, J.; Wu, B.; and Huang, H. 2021. Randomized Algorithms for Submodular Function Maximization with a k-System Constraint. In *International Conference on Machine Learning (ICML)*, 2222–2232.

Dütting, P.; Fusco, F.; Lattanzi, S.; Norouzi-Fard, A.; and Zadimoghaddam, M. 2022a. Deletion robust non-monotone submodular maximization over matroids. *arXiv preprint arXiv:2208.07582*.

Dütting, P.; Fusco, F.; Lattanzi, S.; Norouzi-Fard, A.; and Zadimoghaddam, M. 2022b. Deletion robust submodular maximization over matroids. In *International Conference on Machine Learning (ICML)*, 5671–5693.

El Halabi, M.; Mitrović, S.; Norouzi-Fard, A.; Tardos, J.; and Tarnawski, J. M. 2020. Fairness in streaming submodular maximization: Algorithms and hardness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ene, A.; and Nguyen, H. L. 2019. A Nearly-Linear Time Algorithm for Submodular Maximization with a Knapsack Constraint. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 132, 53.

Fadaei, S.; Fazli, M.; and Safari, M. 2011. Maximizing non-monotone submodular set functions subject to different constraints: Combined algorithms. *Operations Research Letters*, 39(6): 447–451.

Golovin, D.; and Krause, A. 2010. Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization. In *Annual Conference on Computational Learning Theory (COLT)*, 333–345.

Golovin, D.; and Krause, A. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research (JAIR)*, 42: 427–486.

Gupta, A.; Roth, A.; Schoenebeck, G.; and Talwar, K. 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics (WINE)*, 246–257.

Haba, R.; Kazemi, E.; Feldman, M.; and Karbasi, A. 2020. Streaming Submodular Maximization under a k-Set System Constraint. In *International Conference on Machine Learning (ICML)*.

Han, K.; Cui, S.; Zhu, T.; Zhang, E.; Wu, B.; Yin, Z.; Xu, T.; Tang, S.; and Huang, H. 2021. Approximation Algorithms for Submodular Data Summarization with a Knapsack Constraint. In *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*.

Harshaw, C.; Feldman, M.; Ward, J.; and Karbasi, A. 2019. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *International Conference on Machine Learning (ICML)*, 2634–2643.

Huang, C.; Kakimura, N.; and Yoshida, Y. 2020. Streaming Algorithms for Maximizing Monotone Submodular Functions Under a Knapsack Constraint. *Algorithmica*, 82: 1006–1032.

Huang, C.-C.; and Kakimura, N. 2018. Multi-pass streaming algorithms for monotone submodular function maximization. *arXiv preprint arXiv:1802.06212*.

Huang, C.-C.; and Kakimura, N. 2021. Improved streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 83(3): 879–902.

Kazemi, E.; Minaee, S.; Feldman, M.; and Karbasi, A. 2021. Regularized submodular maximization at scale. In *International Conference on Machine Learning (ICML)*, 5356–5366.

Kazemi, E.; Mitrovic, M.; Zadimoghaddam, M.; Lattanzi, S.; and Karbasi, A. 2019. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning (ICML)*, 3311–3320.

Kazemi, E.; Zadimoghaddam, M.; and Karbasi, A. 2018. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In *International Conference on Machine Learning (ICML)*, 2544–2553.

Khuller, S.; Moss, A.; and Naor, J. S. 1999. The budgeted maximum coverage problem. *Information processing letters (IPL)*, 70(1): 39–45.

Krause, A.; McMahan, H. B.; Guestrin, C.; and Gupta, A. 2008. Robust Submodular Observation Selection. *Journal of Machine Learning Research (JMLR)*, 9(12).

Kulik, A.; Shachnai, H.; and Tamir, T. 2013. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematics of Operations Research*, 38(4): 729–739.

Lee, J.; Mirrokni, V. S.; Nagarajan, V.; and Sviridenko, M. 2010. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics (SIDMA)*, 23(4): 2053–2078.

Lei, Q.; Wu, L.; Chen, P.-Y.; Dimakis, A.; Dhillon, I. S.; and Witbrock, M. J. 2019. Discrete adversarial attacks and submodular optimization with applications to text classification. *Systems and Machine Learning (SysML)*, 1: 146–165.

Leskovec, J.; and Krevl, A. 2014. SNAP Datasets: Stanford Large Network Dataset Collection.

Mirzasoleiman, B.; Jegelka, S.; and Krause, A. 2018. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *AAAI Conference on Artificial Intelligence (AAAI)*, 1379–1386.

Mirzasoleiman, B.; Karbasi, A.; and Krause, A. 2017. Deletion-robust submodular maximization: Data summarization with "the right to be forgotten". In *International Conference on Machine Learning (ICML)*, 2449–2458.

Mitrovic, S.; Bogunovic, I.; Norouzi-Fard, A.; Tarnawski, J. M.; and Cevher, V. 2017. Streaming robust submodular maximization: A partitioned thresholding approach. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Orlin, J. B.; Schulz, A. S.; and Udwani, R. 2018. Robust monotone submodular function maximization. *Mathematical Programming*, 172: 505–537.

Rossi, R. A.; and Ahmed, N. K. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Voigt, P.; and Von dem Bussche, A. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676): 10–5555.

Wei, K.; Iyer, R.; and Bilmes, J. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning (ICML)*, 1954–1963.

Wolsey, L. A. 1982. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3): 410–425.

Yaroslavtsev, G.; Zhou, S.; and Avdiukhin, D. 2020. "Bring Your Own Greedy"+ Max: Near-Optimal 1/2-Approximations for Submodular Knapsack. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 3263–3274.

Yu, Q.; Xu, L.; and Cui, S. 2018. Streaming algorithms for news and scientific literature recommendation: Monotone submodular maximization with a d-knapsack constraint. *IEEE Access*, 6: 53736–53747.

Zhang, G.; Tatti, N.; and Gionis, A. 2022. Coresets remembered and items forgotten: submodular maximization with deletions. In *IEEE International Conference on Data Mining (ICDM)*.

Zhu, F.; Jiang, Z.; and Shao, L. 2014. Submodular object recognition. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2457–2464.