Front. Comput. Sci., 2025, 19(6): 196404
https://doi.org/10.1007/s11704-024-40266-4

RESEARCH ARTICLE

Deterministic streaming algorithms for non-monotone submodular maximization

Xiaoming SUN[1,2], **Jialin ZHANG**[1,2], **Shuo ZHANG** (✉)[1,2]

1 State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
2 School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract Submodular maximization is a significant area of interest in combinatorial optimization. It has various real-world applications. In recent years, streaming algorithms for submodular maximization have gained attention, allowing real-time processing of large data sets by examining each piece of data only once. However, most of the current state-of-the-art algorithms are only applicable to monotone submodular maximization. There are still significant gaps in the approximation ratios between monotone and non-monotone objective functions.

In this paper, we propose a streaming algorithm framework for non-monotone submodular maximization and use this framework to design deterministic streaming algorithms for the d-knapsack constraint and the knapsack constraint. Our 1-pass streaming algorithm for the d-knapsack constraint has a $\frac{1}{4(d+1)}-\epsilon$ approximation ratio, using $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ memory, and $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ query time per element, where $\tilde{B}=\min(n,b)$ is the maximum number of elements that the knapsack can store. As a special case of the d-knapsack constraint, we have the 1-pass streaming algorithm with a $1/8-\epsilon$ approximation ratio to the knapsack constraint. To our knowledge, there is currently no streaming algorithm for this constraint when the objective function is non-monotone, even when $d=1$. In addition, we propose a multi-pass streaming algorithm with $1/6-\epsilon$ approximation, which stores $O(\tilde{B})$ elements.

Keywords submodular maximization, streaming algorithms, cardinality constraint, knapsack constraint

1 Introduction

Submodular maximization has become one of the central problems in optimization and combinatorics, with submodular functions embodying the essence of diminishing returns. The inherent trait of a submodular function —that the marginal value of an item decreases as one adds more items to the set—makes it a particularly intriguing function to maximize. The submodular maximization problem arises in a variety of applications including influence maximization [1], machine learning [2,3], sensor placement [4,5], feature selection [6,7], and information gathering [8].

Due to its remarkable significance, submodular maximization has been studied over the past forty years, especially in the offline model. For the monotone case, it cannot exceed $1-1/e$ under cardinality constraints [9,10], a tight approximation ratio met by the natural greedy algorithm [11]. When combined with enumeration techniques, the greedy algorithm also achieves a $1-1/e$ approximation ratio for knapsack constraints [12,13]. The continuous greedy algorithm [14], proposed by Vondrák in 2008, generalizes to address various constraints, including matroid, and d-knapsack and so on constraints with down-closed properties, necessitating randomization due to its sampling process [15,16]. Recently, Buchbinder and Feldman [17] proposed a deterministic algorithm with an approximation ratio of $1-1/e-\epsilon$. For the non-monotone case, where randomization is a tool, the continuous greedy technique facilitates a 0.401 approximation [18], trailing behind the 0.491 limit under cardinality constraints but being superior to the deterministic $1/e$ for the cardinality and 1/4 for the knapsack constraint [19,20].

However, as the digital era progresses, the real challenge arises when we have to deal with massive datasets that are constantly evolving. This is where streaming algorithms come into play. Traditional optimization algorithms that work on static, small-scale datasets fall short when applied to large streaming data, which is continuously fed and cannot be stored or processed in its entirety. Streaming algorithms, in contrast, allow for processing massive datasets in real-time, by seeing each piece of data only once or a few times, making only a fraction of the entire data available at any point. Badanidiyuru et al. [21] proposed a deterministic $1/2-\epsilon$ approximation algorithm for monotone submodular maximization with cardinality constraints in a streaming model with $O\left(\frac{k\log k}{\epsilon}\right)$ memory and sublinear time. Feldman et al. [22] showed that even under the cardinality constraint in the streaming model, the approximation ratio for this problem cannot be better than 1/2, regardless of whether the objective function is monotone

Received March 25, 2024; accepted September 3, 2024

E-mail: zhangshuo19z@ict.ac.cn

or not. For the non-monotone case, Alaluf et al. [23] proposed a 1-pass semi-streaming algorithm framework with a $\alpha/(1+\alpha)$-approximation ratio using an offline α-approximate algorithm. When using the offline continuous greedy algorithm [18], a 0.286-approximation random approximation algorithm can be obtained. When using the deterministic algorithm [19], a 0.268-approximation deterministic algorithm can be obtained. We can see that streaming algorithms are more difficult than offline algorithms because the information mastered is not the whole picture, but only a part of it.

Regarding the knapsack constraint, there are several related works for monotone objective functions. Huang et al. [24,25] proposed a 1-pass streaming algorithm that achieves a $1/3-\epsilon$ approximation, which was later improved to $2/5-\epsilon$ in their subsequent work. Huang and Kakimura [26] proposed a multi-pass streaming algorithm that achieves a $1/2-\epsilon$ approximation. Meanwhile, Yaroslavtsev et al. [27] proposed another multi-pass streaming algorithm that achieves the same approximation ratio and number of passes, but with less memory usage. For the d-knapsack constraint, deterministic streaming algorithms with a $\frac{1}{2d+1}-\epsilon$ approximation ratio for monotone submodular maximization have been proposed in [28,29]. When the function is non-monotone, to the best of our knowledge, there is currently no streaming algorithm for these two constraints.

The non-monotone submodular maximization problem, unlike its monotone counterpart, does not assume that adding elements to a set necessarily increases the value of the function. Many real-world applications require the handling of non-monotone objectives due to the complex interdependencies and trade-offs inherent in real systems. For example, in sensor placement, adding more sensors does not always lead to better coverage due to potential interferences, or in document summarization, adding sentences to a summary might make it less coherent or relevant. In today's era, data is being generated on an unprecedented scale. This vast influx of data necessitates algorithms capable of processing information in a streaming fashion, where data is observed sequentially and decisions must be made on-the-fly. Deterministic streaming algorithms address this need by enabling scalable and efficient processing of massive datasets without storing all elements in memory. In summary, the motivation for designing deterministic streaming algorithms for non-monotone submodular maximization lies in addressing the challenges presented by modern data-driven applications.

1.1 Our contribution

In this paper, we present several improved deterministic streaming algorithms for the submodular maximization problem subject to different constraints, with a focus on the **non-monotone** objective function. To address the submodular maximization problem with non-monotone objective functions, we propose a streaming algorithm framework based on an offline algorithm [30]. We modify this framework to create different versions of the streaming algorithms that are customized to different constraints.

We propose a 1-pass streaming algorithm for the d-knapsack constraint. As a result, we also have a 1-pass streaming algorithm for the knapsack constraint. Additionally, we propose a multi-pass streaming algorithm for knapsack constraint that achieves an improved approximation ratio. Our main results are as follows:

- For the d-knapsack constraint, we propose a deterministic streaming algorithm with $\frac{1}{4(d+1)}-\epsilon$ approximation. It does 1-pass over the data set, stores at most $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ elements, and makes at most $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ queries per element, where $\tilde{B}=\min(n,b)$ is the maximum number of elements the d-knapsack can store. When $d=1$, the d-knapsack constraint reduces to the knapsack constraint. As a special case of the d-knapsack constraint, we have the 1-pass streaming algorithm with a $1/8-\epsilon$ approximation ratio to the knapsack constraint.
- For the knapsack constraint, we propose a multi-pass streaming algorithm with $1/6-\epsilon$ approximation. The algorithm uses $O\left(\frac{\log b}{\epsilon}\right)$ passes, stores $O(\tilde{B})$ elements, and makes at most $O\left(\frac{\log b}{\epsilon}\right)$ queries per element.

As far as we know, there is currently no streaming algorithm subject to the d-knapsack constraint when the objective function is non-monotone, even when $d=1$. We have developed the first 1-pass streaming algorithm to solve this problem. Moreover, we have introduced another deterministic multi-pass streaming algorithm that can enhance the approximation ratio for the knapsack constraint.

We make a more detailed comparison between our and previous results in Table 1.

1.2 Related work

To better illustrate the improvement of our results, this subsection provides a list of results in the literature on *non-monotone* submodular maximization under a d-knapsack constraint and a knapsack constraint.

Table 1 Deterministic streaming algorithms for non-monotone submodular maximization under d-knapsack constraint and knapsack constraint. "Memory" refers to the number of elements in the register

Constraint	Reference	Ratio	Pass	Memory	Monotonicity
d-knapsack	[28]	$\frac{1}{2d+1}-\epsilon$	1	$O(\tilde{B}\epsilon^{-1}\log\tilde{B})$	Monotone
d-knapsack	Theorem 6	$\frac{1}{4(d+1)}-\epsilon$	1	$O(\tilde{B}\epsilon^{-1}\log\tilde{B})$	Non-monotone
Knapsack	[25]	$2/5-\epsilon$	1	$O(\tilde{B}\epsilon^{-4}\log^4\tilde{B})$	Monotone
Knapsack	[27]	$1/2-\epsilon$	$O(1/\epsilon)$	$O(\tilde{B}\epsilon^{-1}\log\tilde{B})$	Monotone
Knapsack	Theorem 6	$1/8-\epsilon$	1	$O(\tilde{B}\epsilon^{-1}\log\tilde{B})$	Non-monotone
Knapsack	Theorem 8	$1/6-\epsilon$	$O(\log b/\epsilon)$	$O(\tilde{B})$	Non-monotone

For the knapsack constraint, considering the maximum element and the greedy solution can achieve a $1/3-\epsilon$ approximation for monotone submodular maximization [24] by 1-pass. Huang and Kakimura [25] improved this approximation ratio to 2/5 through more detailed discussion. However, the current results of 1-pass cannot match the cardinality constraints whose approximation ratio is tight. Huang and Kakimura [26] proposed a multi-pass streaming algorithm, which achieved $1/2-\epsilon$ for monotonous issues. Meanwhile, Yaroslavtsev et al. [27] proposed another multi-pass streaming algorithm with a $1/2-\epsilon$ approximation ratio using the thresholding technique. For the non-monotone submodular maximization problem subject to the knapsack constraint, to the best of our knowledge, there is no streaming algorithm in the literature.

For the d-knapsack constraint, when d is constant or the width of the constraints is large, a constant approximation is possible [31–33]. Currently, the best-known algorithm is again based on the continuous greedy algorithm and has an approximation ratio of 0.401 [18]. Sun et al. [34] proposed a deterministic algorithm for the offline setting with $1/6$ approximation when the width of the knapsack is large enough. To the best of our knowledge, there is no streaming algorithm for the non-monotone submodular maximization problem in the literature. For the monotone case, Yu et al. [29] proposed a deterministic streaming algorithm for the d-knapsack constraint with $\frac{1}{2d+1}-\epsilon$ approximation ratio, whose memory cost is $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ and query time is $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ per element.

1.3 Organization

In Section 2, we formally introduce the problem of non-monotone submodular maximization under the cardinality constraint, the d-knapsack constraint, and the knapsack constraint. In Section 3, we propose a deterministic 1-pass streaming algorithm for the cardinality constraint as a warm-up with $\frac{1}{6}-\epsilon$ approximation ratio using $O\left(\frac{k\log k}{\epsilon}\right)$ memory. In Section 4, we propose a deterministic 1-pass streaming algorithm for the d-knapsack constraint with $\frac{1}{4(d+1)}-\epsilon$ approximation ratio using $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ memory, where $\tilde{B}=\min(n,b)$ is the maximum number of elements that the knapsack can store. In Section 5, we propose a deterministic multi-pass algorithm for the knapsack constraint with a $1/6-\epsilon$ approximation ratio using $O(\tilde{B})$ memory. In Section 6, we conclude the paper and list some future directions.

2 Preliminaries

In this section, we state the problems studied in this paper.

Definition 1 (Submodular function). Given a finite ground set N of n elements, a set function $f:2^N\mapsto\mathbb{R}$ is submodular if for all $S,T\subseteq N$,

$$f(S)+f(T)\geqslant f(S\cup T)+f(S\cap T).$$

Equivalently, f is submodular if for all $S\subseteq T\subseteq N$ and $u\in N\setminus T$,

$$f(S\cup\{u\})-f(S)\geqslant f(T\cup\{u\})-f(T).$$

For convenience, we use $f(u)$ to denote $f(\{u\})$, $f(S+u)$ to denote $f(S\cup\{u\})$, $f(u\,|\,T)$ to denote the marginal value $f(T+u)-f(T)$ of u with respect to T, and $f(S\,|\,T)$ to denote the marginal value $f(S\cup T)-f(T)$. The function f is *non-negative* if $f(S)\geqslant 0$ for all $S\subseteq N$. The function f is *monotone* if $f(S)\leqslant f(T)$ for all $S\subseteq T\subseteq N$.

Definition 2 (Cardinality). Given a finite ground set N and an integer k. For set $S\subseteq N$, the cardinality constraints can be defined as $\mathcal{I}=\{S\,|\,|S|\leqslant k\}$.

Definition 3 (Knapsack). Given a finite ground set N, assume there is a budget b, and each element $u\in N$ is associated with a cost $c(u)>0$. For set $S\subseteq N$, its cost $c(S)=\sum_{u\in S}c(u)$. We say S is feasible if $c(S)\leqslant b$. The knapsack can be written as $\mathcal{I}=\{S\,|\,c(S)\leqslant b\}$.

Without loss of generality, we can simply assume that all cost $c(u)\geqslant 1$.

If $c(u)=1$ for all $u\in N$ in the knapsack and let $b=k$, the knapsack reduces to the *cardinality constraint*.

Definition 4 (d-Knapsack). Given a finite ground set N a matrix $\mathbf{C}\in(0,\infty)^{d\times n}$, and a vector $\mathbf{b}\in(0,\infty)^d$. For set $S\subseteq N$, the d-knapsack constraints can be written as $\mathcal{I}=\{S\,|\,\mathbf{C}\mathbf{x}_\mathbf{S}\leqslant\mathbf{b}\}$, where $\mathbf{x}_\mathbf{S}$ stands for the characteristic vector of the set S.

Without loss of generality, for $1\leqslant i\leqslant d, e\in N$, we assume that $c_{i,e}\leqslant b_i$. For the sake of simplicity, we can standardize the constrained problem. We can simply assume that all cost $c_{i,e}\geqslant 1$. Let $b=\max_{1\leqslant i\leqslant d}b_i$. For $1\leqslant i\leqslant d, e\in N$, we replace each $c_{i,e}$ with $bc_{i,e}/b_i$ and b_i with b such that the d-knapsack has equal capacity b in all dimensions. The standardized problem has the same optimal solution. In the remainder of the paper, we will only consider the standardized version of the submodular maximization problem subject to a d-knapsack constraint.

Let $\tilde{B}=\min(|N|,b)$ which means the maximum number of elements the knapsack can store.

When $d=1$, the d-knapsack constraint reduces to the knapsack constraint.

Definition 5 (Constrained submodular maximization). The constrained submodular maximization problem has the form

$$\max\{f(S)\,|\,S\in\mathcal{I}\}.$$

In this paper, the constraint $\mathcal{I}$ is assumed to be the cardinality constraint or the d-knapsack constraint, respectively. The objective function f is assumed to be non-negative and submodular.

Besides, f is accessed by a value oracle that returns $f(S)$ when S is queried. As query access can be costly, the number of queries is usually considered to be a performance metric. Space complexity, on the other hand, refers to the amount of memory required to process and make decisions on the data stream. If the length of the stream is n, and the size of the parameters is k, the algorithm is generally constrained to using memory proportional to k or the logarithm of k, but independent of the length of the stream n. In most cases, the algorithm can only make a small constant number of passes over the stream, sometimes just one.

3 Warm-up: streaming algorithm for cardinality constraint

In this section, we use a cardinality-constrained streaming algorithm as a warm-up to fully demonstrate the basic version of our algorithm framework. Our framework proposes a streaming algorithm to solve the non-monotone submodular maximization problem, which can be applied to many constraints. To illustrate the framework, we present a 1-pass streaming algorithm for the cardinality constraint in this section. Although the approximation performance is slightly worse under cardinality constraints [23], it exhibits good adaptability to more complex constraints. Next, in the following section, we use this framework to design a 1-pass streaming algorithm for the d-knapsack constraint. Furthermore, we employ this framework to create a multi-pass streaming algorithm for the knapsack constant.

Our algorithm is obtained by modifying the technique from [30] to a streaming model, using the threshold method in the Sieve-Streaming algorithm [21]. First, we assume that we have some knowledge of OPT, and then remove this assumption by estimating OPT based on the max value of a single element. Finally, we remove all the assumptions and propose a 1-pass streaming algorithm subject to a cardinality constraint. Our algorithm framework is simpler than the state-of-the-art algorithm proposed by [23].

3.1 Streaming algorithm knowing OPT

Suppose that we have a value v such that $\alpha\text{OPT} \leqslant v \leqslant \text{OPT}$, for some $\alpha \in (0,1]$. We construct the algorithm to choose elements by the threshold according to the value of v. The resulting algorithm is depicted as Algorithm 1. It first invokes the thresholding progress to obtain a solution S_1. Then it runs the thresholding progress again over the remaining elements $N \setminus S_1$ to obtain another solution S_2. These two solutions still cannot guarantee any constant approximation ratios. To remedy this, the algorithm produces the third solution S_3 by solving the unconstrained submodular maximization problem over S_1.

Theorem 1 Assuming that the input v satisfies $\alpha\,\text{OPT} \leqslant v \leqslant \text{OPT}$, Algorithm 1 satisfies the following properties, where the unconstrained submodular maximization is solved by a deterministic 1/2-approximation algorithm proposed by [19].

Algorithm 1 Streaming repeat greedy for cardinality knowing OPT

1: **Input:** stream $e_1,\ldots,e_n$, size constraint k, non-negative submodular function f, v such that $\alpha\text{OPT} \leqslant v \leqslant \text{OPT}$, for some $\alpha \in (0,1]$.
2: $S_1, S_2, S_3 = \emptyset$.
3: $\tau = v/6$. ▷ τ is the threshold.
4: **for** each read item e **do**
5: **if** $f(e \mid S_1) \geqslant \frac{\tau}{k}$ and $|S_1| < k$ **then**
6: $S_1 = S_1 \cup \{e\}$.
7: **else if** $f(e \mid S_2) \geqslant \frac{\tau}{k}$ and $|S_2| < k$ **then**
8: $S_2 = S_2 \cup \{e\}$.
9: $S_3 = \textbf{Unconstrained}\ (S_1)$.
10: **return** $\text{argmax}\{f(S_1), f(S_2), f(S_3)\}$.

- It outputs a set S such that $|S| \leqslant k$ and $f(S) \geqslant \frac{\alpha}{6}\,\text{OPT}$.
- It does 1 pass over the data set, stores at most $2k$ elements, and has $O(1)$ query complexity per element.

Proof We prove this theorem by two cases subject to the size of two candidate sets S_1 and S_2 at the end of streaming. Let set O be the optimal set such that $f(O) = \text{OPT}$, and S_l^j be the candidate set at the end of the j-iteration, $j = 1,2,\ldots,n$, $S_l^0 = \emptyset$, and $S_l^n = S_l$, $l = 1,2$.

Case 1 At the end of the algorithm, at least one candidate set has k elements. Without loss of generality, let $|S_1| = k$, and for $1 \leqslant i \leqslant k$, let u_i be the ith element added into S_1. Then we have that $f(S) = \sum_{i=1}^{k}(f(\{u_1,u_2,\ldots,u_i\}) - f(\{u_1,u_2,\ldots,u_{i-1}\})) \geqslant k \cdot \frac{\tau}{k} = \frac{v}{6}$. Thus, $f(S) \geqslant f(S_1) \geqslant \frac{\alpha}{6}\,\text{OPT}$.

Case 2 Neither candidate set is full, that is, $|S_1| < k$ and $|S_2| < k$ at the end of the algorithm. For every element $u \in O \setminus S_1$, let it be rejected by S_1 (in line 5) in iteration j. Then we have $f(u \mid S_1) \leqslant f(u \mid S_1^j) \leqslant \frac{\tau}{k}$, which implies that $f(O \mid S_1) \leqslant \sum_{u \in O \setminus S_1} f(u \mid S_1) \leqslant \tau$, and further $f(S_1) \geqslant f(S_1 \cup O) - \tau$. Similarly, we have $f((O \setminus S_1) \mid S_2) \leqslant \tau$, which implies that $f(S_2) \geqslant f(S_2 \cup (O \setminus S_1)) - \tau$.

$$\begin{aligned} f(S) &\geqslant \frac{1}{2}(f(S_1) + f(S_2)) \\ &\geqslant \frac{1}{2}(f(S_1 \cup O) + f(S_2 \cup (O \setminus S_1)) - 2\tau) \\ &= \frac{1}{2}f(S_1 \cup O) + \frac{1}{2}f(S_2 \cup (O \setminus S_1)) \\ &\quad + \frac{1}{2}f(S_1 \cap O) - \frac{1}{2}f(S_1 \cap O) - \tau \\ &\geqslant \frac{1}{2}f(O) - \frac{1}{2}f(S_1 \cap O) - \tau. \end{aligned}$$

The last inequality is due to submodularity and non-negativity of f. Note that $f(S_1 \cup O) + f(S_2 \cup (O \setminus S_1)) + f(S_1 \cap O) \geqslant f(S_1 \cup O) + f(S_2 \cup O) \geqslant f(O) + f(S_1 \cup S_2 \cup O) \geqslant f(O)$. We use a $\frac{1}{2}$-approximation deterministic unconstrained submodular maximization algorithm to solve the part of $f(S_1 \cap O)$ [19]. That is, if $f(S_1 \cap O) \geqslant 2\tau$, then we have $f(S) \geqslant f(S_3) \geqslant \tau$. Otherwise, $f(S_1 \cap O) < 2\tau$, then

$$\begin{aligned} f(S) &\geqslant \frac{1}{2}f(O) - \frac{1}{2}f(S_1 \cap O) - \tau \\ &\geqslant \frac{1}{2}f(O) - 2\tau \\ &\geqslant 3\tau - 2\tau \\ &= \tau, \end{aligned}$$

where the last inequality is by $f(O) \geqslant v \geqslant 6\tau$. Then we have $f(S) \geqslant \frac{\alpha}{6}\,\text{OPT}$ in any case. □

3.2 Streaming algorithm knowing the max value

We use the maximum value element $m = \max_{u \in N} f(u)$ to estimate OPT. By submodularity, we have that $m \leqslant \text{OPT} \leqslant k \cdot m$. Consider the following set $Q = \{(1+\epsilon)^i \mid i \in \mathbb{Z}, m \leqslant (1+\epsilon)^i \leqslant k \cdot m\}$. At least one of the values $v \in Q$ should be a pretty good estimate of OPT, such that $\frac{1}{1+\epsilon}\,\text{OPT} \leqslant v \leqslant \text{OPT}$. We can run Algorithm 1 once for each value $v \in Q$ in parallel, producing one candidate set for each

$v \in Q$. As the final output, we return the best solution obtained.

Theorem 2 Assuming that the input m satisfies $m = \max_{u \in N} f(u)$, Algorithm 2 satisfies the following properties, where the unconstrained submodular maximization is solved by a deterministic 1/2-approximation algorithm proposed by [19].

- It outputs a set S such that $|S| \leqslant k$ and $f(S) \geqslant (\frac{1}{6} - \epsilon)\mathrm{OPT}$.
- It does 1 pass over the data set, stores at most $O(\frac{k \log k}{\epsilon})$ elements, and has $O\left(\frac{\log k}{\epsilon}\right)$ query complexity per element.

Proof Because at least one of the values $v \in Q$ should be a pretty good estimation of OPT such that $\frac{1}{1+\epsilon}\mathrm{OPT} \leqslant v \leqslant \mathrm{OPT}$, the approximation ratio of Algorithm 2 is the same as that of Algorithm 1, together with their proofs. We only analyze the memory and the query complexity of Algorithm 2. Note that $|Q| = O\left(\frac{\log k}{\epsilon}\right)$, we keep track of $O\left(\frac{\log k}{\epsilon}\right)$ many sets S_l^v of size at most k each, bounding the size of memory by $O\left(\frac{k \log k}{\epsilon}\right)$, $l = 1, 2, 3$. Moreover, the query time is $O\left(\frac{\log k}{\epsilon}\right)$ per element. □

3.3 1-pass streaming algorithm for cardinality constraint

In order to find the maximum value element $m = \max_{u \in N} f(u)$ during the streaming, we modify the set into $Q = \{(1+\epsilon)^i \mid i \in \mathbb{Z}, m \leqslant (1+\epsilon)^i \leqslant 6k \cdot m\}$. It can be seen that when a threshold v is instantiated from the set Q, every element with marginal value $\frac{v}{6k}$ to S^v will appear on or after v is instantiated.

Theorem 3 Algorithm 3 satisfies the following properties, where the unconstrained submodular maximization is solved by a deterministic 1/2-approximation algorithm proposed by [19].

- It outputs a set S such that $|S| \leqslant k$ and $f(S) \geqslant (\frac{1}{6} - \epsilon)\mathrm{OPT}$.

Algorithm 2 Streaming repeat greedy for cardinality knowing the max value

Input: stream $e_1, \ldots, e_n$, size constraint k, non-negative submodular function f, $m = \max_{e \in N} f(e)$.
$Q = \{(1+\epsilon)^i \mid i \in E, m \leqslant (1+\epsilon)^i \leqslant k \cdot m\}$.
for $v \in Q$ **do**
  $S_1^v, S_2^v, S_3^v = \emptyset$.
for each read item e **do**
  for $v \in Q$ **do**
    $\tau = v/6$. ▷ τ is the threshold.
    if $f(e \mid S_1^v) \geqslant \frac{\tau}{k}$ and $|S_1^v| < k$ **then**
      $S_1^v = S_1^v \cup \{e\}$.
    else if $f(e \mid S_2^v) \geqslant \frac{\tau}{k}$ and $|S_2^v| < k$ **then**
      $S_2^v = S_2^v \cup \{e\}$.
for $v \in Q$ **do**
  $S_3^v =$ **Unconstrained** (S_1^v).
return $\mathrm{argmax}_{v \in Q}\{f(S_1^v), f(S_2^v), f(S_3^v)\}$.

Algorithm 3 1-pass streaming repeat greedy for cardinality

Input: stream $e_1, \ldots, e_n$, size constraint k, non-negative submodular function f.
$P = \{(1+\epsilon)^i \mid i \in \mathbb{Z}\}$.
for $v \in P$ **do**
  $S_1^v, S_2^v, S_3^v = \emptyset$.
$m = 0$.
for each read item e **do**
  $m = \max(m, f(e))$.
  $Q = \{(1+\epsilon)^i \mid i \in \mathbb{Z}, m \leqslant (1+\epsilon)^i \leqslant 6k \cdot m\}$.
  for $v \in Q$ **do**
    $\tau = v/6$. ▷ τ is the threshold.
    if $f(e \mid S_1^v) \geqslant \frac{\tau}{k}$ and $|S_1^v| < k$ **then**
      $S_1^v = S_1^v \cup \{e\}$.
    else if $f(e \mid S_2^v) \geqslant \frac{\tau}{k}$ and $|S_2^v| < k$ **then**
      $S_2^v = S_2^v \cup \{e\}$.
for $v \in Q$ **do**
  $S_3^v =$ **Unconstrained** (S_1^v).
return $\mathrm{argmax}_{v \in Q}\{f(S_1^v), f(S_2^v), f(S_3^v)\}$.

- It does 1 pass over the data set, stores at most $O\left(\frac{k \log k}{\epsilon}\right)$ elements, and has $O\left(\frac{\log k}{\epsilon}\right)$ query complexity per element.

Proof For a specific threshold, we maintain two candidate sets. Thus, for a set of threshold ranges, we maintain two sets of candidate sets. When a threshold v is instantiated from the set Q, every element with marginal value $\frac{v}{6k}$ to S^v will appear on or after v is instantiated. To ensure this, we expand the range of the threshold in Algorithm 3 by a factor of $6k$. For those thresholds smaller than this range, we can directly discard them, because the marginal density increment there cannot guarantee the correct optimal solution threshold. The remaining proof is similar to Theorem 2. □

4 Streaming algorithm for *d*-knapsack constraint

In this section, we design a 1-pass streaming algorithm for d-knapsack constraint, whose algorithm framework is consistent with that shown in the warm-up section. When dealing with d-knapsack constraints, we encounter a challenge where we cannot use the enumerate technique (used in the offline setting) to solve the problem caused by insufficient capacity. To address this issue, we use a technique from [29] for the monotone case, which takes into account the large single element. Additionally, to handle the lack of monotonicity, we use the streaming algorithm framework which allows us to maintain two candidate sets. We also start describing the algorithm from the assumption that we know the value of OPT. Then, we remove this assumption by estimating OPT based on the maximum marginal unit value of all single elements. Finally, we remove all the assumptions and propose a 1-pass streaming algorithm subject to the d-knapsack constraint.

4.1 Streaming algorithm knowing OPT

Suppose we have a value v such that $\alpha\mathrm{OPT} \leqslant v \leqslant \mathrm{OPT}$, for some $\alpha \in (0, 1]$. Then we develop the algorithm to choose elements by the threshold according to the value of v.

Algorithm 4 Streaming repeat greedy for d-knapsack knowing OPT

1: **Input:** stream $e_1,\ldots,e_n$, d-knapsack capacity $\mathbf{b}$, cost function $\mathbf{C}$, non-negative submodular function f, v such that $\alpha\mathrm{OPT}\leqslant v\leqslant \mathrm{OPT}$, for some $\alpha\in(0,1]$.
2: $S_1,S_2,S_3=\emptyset$.
3: $\tau=\frac{v}{4(d+1)}$. ▷ τ is the threshold.
4: **for** each read item e **do**
5: **if** $c_{i,e}\geqslant\frac{b}{2}$ and $\frac{f(e)}{c_{i,e}}\geqslant\frac{2\tau}{b}$ for some $1\leqslant i\leqslant d$ **then**
6: $S=\{e\}$.
7: **return** S.
8: **if** $\frac{f(e|S_1)}{c_{i,e}}\geqslant\frac{2\tau}{b}$ and $\sum_{l\in S_1\cup\{e\}}c_{i,l}\leqslant b$ for all $1\leqslant i\leqslant d$ **then**
9: $S_1=S_1\cup\{e\}$.
10: **else if** $\frac{f(e|S_2)}{c_{i,e}}\geqslant\frac{2\tau}{b}$ and $\sum_{l\in S_2\cup\{e\}}c_{i,l}\leqslant b$ for all $1\leqslant i\leqslant d$ **then**
11: $S_2=S_2\cup\{e\}$.
12: $S_3=$ **Unconstrained** (S_1).
13: **return** $\mathrm{argmax}\{f(S_1),f(S_2),f(S_3)\}$.

Theorem 4 Assuming that the input v satisfies $\alpha\mathrm{OPT}\leqslant v\leqslant\mathrm{OPT}$, Algorithm 4 satisfies the following properties, where the unconstrained submodular maximization is solved by a deterministic 1/2-approximation algorithm proposed by [19].

- It outputs a set S such that $\mathbf{C}\mathbf{x}_\mathbf{S}\leqslant\mathbf{b}$ and $f(S)\geqslant\frac{\alpha}{4(d+1)}\mathrm{OPT}$.
- It does 1 pass over the data set, stores at most $2\tilde{B}$ elements, and has $O(d)$ query complexity per element.

Proof Similarly, we consider the size of the two candidate sets at the end of the algorithm. The algorithm will terminate when either we find an element $u^*\in N$ such that $c_{i,u^*}\geqslant\frac{b}{2}$ and $\frac{f(u^*)}{c_{i,u^*}}\geqslant\frac{2\tau}{b}$ for some $1\leqslant i\leqslant d$, or we finish one pass through the dataset. Here we define that an element $u^*\in N$ is a **big element** if it satisfies the condition in **line 5**.

Let set O be the optimal set such that $f(O)=\mathrm{OPT}$, and S_l^j be the candidate set at the end of the j-iteration, $j=1,2,\ldots,n$, $S_l^0=\emptyset$, and $S_l^n=S_l$, $l=1,2$.

1. We first prove that if we find a big element, the set $S=\{u^*\}$ will obtain a good approximate ratio. Assume N has at least one big element. Let u^* be the first big element that the algorithm finds. Then the algorithm outputs $S=\{u^*\}$ and terminates. Then by **line 5**, we have $f(S)=f(u^*)\geqslant\frac{2\tau}{b}\cdot\frac{b}{2}=\tau$. Thus, output S of Algorithm 4 satisfies $f(S)\geqslant\frac{v}{4(d+1)}\geqslant\frac{\alpha}{4(d+1)}\mathrm{OPT}$.
2. Otherwise, if N has no big element, we discuss the following two cases subject to the size of two candidate sets S_1 and S_2 at the end of streaming.

Case 1. At the end of the algorithm, at least one candidate set satisfies $\sum_{u\in S_l}c_{i,u}\geqslant\frac{b}{2}$ for some $1\leqslant i\leqslant d$, $l=\{1,2\}$.

Without loss of generality, let $l=1$. Assume that the elements in S_1 is selected in order $\{u_1,u_2,\ldots,u_{|S_1|}\}$. We have $f(S)\geqslant\sum_{j=1}^{|S_1|}(f(\{u_1,u_2,\ldots,u_j\})-f(\{u_1,u_2,\ldots,u_{j-1}\}))$. By the algorithm, we have that $f(\{u_1,u_2,\ldots,u_j\})-f(\{u_1,u_2,\ldots,u_{j-1}\})\geqslant\frac{2\tau}{b}c_{i,u_j}$, for all $1\leqslant i\leqslant d$. Then for such i that $\sum_{u_j\in S_1}c_{i,u_j}\geqslant\frac{b}{2}$, we have $f(S)\geqslant\sum_{j=1}^{|S_1|}\frac{2\tau}{b}c_{i,u_j}\geqslant\tau$.

Case 2. Both of the sizes of S_1 and S_2 have that $\sum_{u\in S_l}c_{i,u}<\frac{b}{2}$ for all $1\leqslant i\leqslant d$, at the end of the algorithm, $l=\{1,2\}$.

First, let's consider the case of S_1. For each element $e\in O\setminus S_1$, let it be recorded as e_j if rejected in iteration j. There exists an index $\mu(e_j)$, with $1\leqslant\mu(e_j)\leqslant d$ such that $\frac{f(e_j|S_1)}{c_{\mu(e_j),e_j}}\leqslant\frac{f(e_j|S_1^j)}{c_{\mu(e_j),e_j}}<\frac{2\tau}{b}$. By contradiction, if $\frac{f(e_j|S_1^j)}{c_{\mu(e_j),u_j}}\geqslant\frac{2\tau}{b}$, since e_j is not a big element and f is submodular, we have $c_{i,e_j}<b/2$, for $1\leqslant i\leqslant d$. Then e_j can be added into S_1, where a contradiction occurs.

Let Y_i be the set containing elements $e_j\in O\setminus S_1$ such that $\mu(e_j)=i$, for $1\leqslant i\leqslant d$. Then $O\setminus S_1=\cup_{1\leqslant i\leqslant d}Y_i$. We have that $f(Y_i\mid S_1)<\frac{2\tau}{b}\sum_{e_j\in Y_i}c_{\mu(e_j),e_j}<2\tau$, which implies that $f(O\mid S_1)\leqslant\sum_{i=1}^d f(Y_i\mid S_1)<2d\tau$ and further $f(S_1)\geqslant f(S_1\cup O)-2d\tau$.

Similarly, we have $f((O\setminus S_1)\mid S_2)\leqslant 2d\tau$, which implies that $f(S_2)\geqslant f(S_2\cup(O\setminus S_1))-2d\tau$.

$$\begin{aligned}f(S)&\geqslant\frac{1}{2}(f(S_1)+f(S_2))\\&\geqslant\frac{1}{2}f(S_1\cup O)+\frac{1}{2}f(S_2\cup(O\setminus S_1))-2d\tau\\&=\frac{1}{2}f(S_1\cup O)+\frac{1}{2}f(S_2\cup(O\setminus S_1))\\&\quad+\frac{1}{2}f(S_1\cap O)-\frac{1}{2}f(S_1\cap O)-2d\tau\\&\geqslant\frac{1}{2}f(O)-\frac{1}{2}f(S_1\cap O)-2d\tau,\end{aligned}$$

where the third inequality is due to submodularity and non-negativity of f. Note that $f(S_1\cup O)+f(S_2\cup(O\setminus S_1))+f(S_1\cap O)\geqslant f(S_1\cup O)+f(S_2\cup O)\geqslant f(O)+f(S_1\cup S_2\cup O)\geqslant f(O)$. We bound the value of $f(S_1\cap O)$ by two cases. That is, if $f(S_1\cap O)\geqslant\frac{v}{2}-2d\tau$, then we have $f(S)\geqslant f(S_3)\geqslant\frac{v}{4}-d\tau\geqslant\tau$ by the unconstrained submodular maximization algorithm [19]. Otherwise, $f(S_1\cap O)\leqslant\frac{v}{2}-2d\tau$, then

$$\begin{aligned}f(S)&\geqslant\frac{1}{2}f(O)-\frac{1}{2}f(S_1\cap O)-2d\tau\\&\geqslant\frac{1}{2}f(O)-\frac{v}{4}+d\tau-2d\tau\\&\geqslant 2(d+1)\tau-(d+1)\tau-d\tau\\&=\tau,\end{aligned}$$

where the last inequality is by $f(O)\geqslant v\geqslant 4(d+1)\tau$. Thus, we have $f(S)\geqslant\frac{\alpha}{4(d+1)}\mathrm{OPT}$ in any case. □

4.2 Streaming algorithm knowing the max density

We obtain an approximation of OPT by Algorithm 4 which requires OPT as an input. That is, we have to estimate OPT first. Like in the cardinality constraint that we estimate OPT using the *max value* of the element, we can estimate OPT by the *max density* of the element.

Lemma 1 Let $Q=\{(1+\epsilon)^i\mid i\in\mathbb{Z},m/(1+\epsilon)\leqslant(1+\epsilon)^i\leqslant\tilde{B}m\}$, where $\tilde{B}=\min\{n,b\}$ is the maximum number of elements that the d-knapsack can store. Then there is at least one $v\in Q$ such that $(1-\epsilon)\mathrm{OPT}\leqslant v\leqslant\mathrm{OPT}$.

Proof Without loss of generality, let $m = \max_{1\leqslant i'\leqslant d, 1\leqslant j'\leqslant n} \frac{f(u_{j'})}{c_{i',u_{j'}}}$. Since $c_{i',u_{j'}} \geqslant 1$, we have $\mathrm{OPT} \geqslant f(u_{j'}) = mc_{i',u_{j'}} \geqslant m$. On the other side, we have $\mathrm{OPT} = \sum_{i=1}^{|O|}(f(\{u_1,u_2,\ldots,u_i\}) - f(\{u_1,u_2,\ldots,u_{i-1}\})) \leqslant \sum_{i=1}^{|O|} f(u_i) \leqslant m\sum_{i=1}^{|O|} c_{1,u_i} \leqslant \tilde{B}m$.

Let $v = (1+\epsilon)^{\lfloor \log_{1+\epsilon}\mathrm{OPT}\rfloor}$. We obtain $\frac{m}{1+\epsilon} \leqslant (1-\epsilon)\,\mathrm{OPT} \leqslant v \leqslant \mathrm{OPT} \leqslant \tilde{B}m$. □

By Lemma 1, we design the following algorithm requiring the *max density* as an input.

Theorem 5 Assuming that the input m satisfies $m = \max_{1\leqslant i\leqslant d, 1\leqslant j\leqslant n} f(u_j)/c_{i,u_j}$, Algorithm 5 satisfies the following properties, where the unconstrained submodular maximization is solved by a deterministic 1/2-approximation algorithm proposed by [19].

- It outputs a set S such that $\mathbf{Cx_S} \leqslant \mathbf{b}$ and $f(S) \geqslant \left(\frac{1}{4(d+1)} - \epsilon\right)\mathrm{OPT}$.
- It does 1 pass over the data set, stores at most $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ elements, and has $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ query complexity per element.

Proof By Lemma 1, we choose $v \in Q$ such that $(1-\epsilon)\,\mathrm{OPT} \leqslant v \leqslant \mathrm{OPT}$. Then by Theorem 4, the output set S satisfies $f(S) \geqslant \left(\frac{1}{4(d+1)} - \epsilon\right)\mathrm{OPT}$.

Notice that there are $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ elements in Q, and for each v there are at most $\tilde{B}$ elements in S_k^v. Thus, Algorithm 5 stores at most $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ elements and has $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ query complexity per element. □

4.3 1-pass streaming algorithm for d-knapsack constraint

We modify the estimation candidate set Q into $Q = \{(1+\epsilon)^k \mid k \in \mathbb{Z}, m/(1+\epsilon) \leqslant (1+\epsilon)^k \leqslant 2(d+1)bm\}$. Let m be the current maximum marginal value per weight of all single elements. The streaming algorithm will perform a parallel threshold algorithm while updating m and the estimation candidate set Q. It can be seen that when a threshold v is instantiated from the set Q, every element with marginal value $\frac{v}{4(d+1)}$ to S^v will appear on or after v is instantiated. Finally, we develop the final 1-pass streaming algorithm and establish the following theorem, whose proof follows the same lines as the proof of Theorem 5.

Theorem 6 Algorithm 6 satisfies the following properties, where the unconstrained submodular maximization is solved by a deterministic 1/2-approximation algorithm proposed by [19].

- It outputs a set S such that $\mathbf{Cx_S} \leqslant \mathbf{b}$ and $f(S) \geqslant \left(\frac{1}{4(d+1)} - \epsilon\right)\mathrm{OPT}$.
- It does 1 pass over the data set, stores at most $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ elements and has $O\left(\frac{\log\tilde{B}}{\epsilon}\right)$ query complexity per element.

Proof When a threshold v is instantiated from the set Q, every element with marginal value $\frac{v}{4(d+1)}$ to S^v will appear on or after v is instantiated. The remaining proof is similar to Theorem 5. In the setting of the streaming algorithm, we consider d to be much smaller than b and treat it as a constant when considering memory space. □

One immediate corollary is that the Algorithm 6 can achieve a $\frac{1}{8} - \epsilon$ approximation when $d = 1$, which means the single

Algorithm 5 Streaming repeat greedy for d-knapsack knowing m

Input: stream $e_1,\ldots,e_n$, d-knapsack capacity $\mathbf{b}$, cost function $\mathbf{C}$, non-negative submodular function f, the max density of the element $m = \max_{1\leqslant i'\leqslant d, 1\leqslant j'\leqslant n} \frac{f(u_{j'})}{c_{i',u_{j'}}}$.
$Q = \{(1+\epsilon)^i \mid i \in \mathbb{Z}, m/(1+\epsilon) \leqslant (1+\epsilon)^i \leqslant bm\}$.
for $v \in Q$ **do**
  $S_1^v, S_2^v, S_3^v, S^v = \emptyset$.
for each read item e **do**
  for $v \in Q$ **do**
    $\tau = \frac{v}{4(d+1)}$. ▷ τ is the threshold.
    if $c_{i,e} \geqslant \frac{b}{2}$ and $\frac{f(e)}{c_{i,e}} \geqslant \frac{2\tau}{b}$ for some $1 \leqslant i \leqslant d$ **then**
      $S^v = \{e\}$.
    else if $\frac{f(e|S_1^v)}{c_{i,e}} \geqslant \frac{2\tau}{b}$ and $\sum_{l\in S_1^v\cup\{e\}} c_{i,l} \leqslant b$ for all $1 \leqslant i \leqslant d$ **then**
      $S_1^v = S_1^v \cup \{e\}$.
    else if $\frac{f(e|S_2^v)}{c_{i,e}} \geqslant \frac{2\tau}{b}$ and $\sum_{l\in S_2^v\cup\{e\}} c_{i,l} \leqslant b$ for all $1 \leqslant i \leqslant d$ **then**
      $S_2^v = S_2^v \cup \{e\}$.
for $v \in Q$ **do**
  S_3^v = **Unconstrained** (S_1^v).
return $\mathrm{argmax}_{v\in Q}\{f(S_1^v), f(S_2^v), f(S_3^v), f(S^v)\}$.

Algorithm 6 1-Pass streaming repeat greedy for d-knapsack

Input: stream $e_1,\ldots,e_n$, d-knapsack capacity $\mathbf{b}$, cost function $\mathbf{C}$, non-negative submodular function f.
$P = \{(1+\epsilon)^k \mid k \in \mathbb{Z}\}$.
for $v \in P$ **do**
  $S_1^v, S_2^v, S_3^v, S^v = \emptyset$.
$m = 0$.
for each read item e **do**
  for $i = 1$ to d **do**
    $m = \max\{m, \frac{f(e)}{c_{i,e}}\}$.
  $Q = \{(1+\epsilon)^k \mid k \in \mathbb{Z}, m/(1+\epsilon) \leqslant (1+\epsilon)^k \leqslant 2(d+1)bm\}$.
  for $v \in Q$ **do**
    $\tau = \frac{v}{4(d+1)}$. ▷ τ is the threshold.
    if $c_{i,e} \geqslant \frac{b}{2}$ and $\frac{f(e)}{c_{i,e}} \geqslant \frac{2\tau}{b}$ for some $1 \leqslant i \leqslant d$ **then**
      $S^v = \{e\}$.
    else if $\frac{f(e|S_1^v)}{c_{i,e}} \geqslant \frac{2\tau}{b}$ and $\sum_{l\in S_1^v\cup\{e\}} c_{i,l} \leqslant b$ for all $1 \leqslant i \leqslant d$ **then**
      $S_1^v = S_1^v \cup \{e\}$.
    else if $\frac{f(e|S_2^v)}{c_{i,e}} \geqslant \frac{2\tau}{b}$ and $\sum_{l\in S_2^v\cup\{e\}} c_{i,l} \leqslant b$ for all $1 \leqslant i \leqslant d$ **then**
      $S_2^v = S_2^v \cup \{e\}$.
for $v \in Q$ **do**
  S_3^v = **Unconstrained** (S_1^v).
return $\mathrm{argmax}_{v\in Q}\{f(S_1^v), f(S_2^v), f(S_3^v), f(S^v)\}$.

knapsack constraint.

Corollary 1 Let $\tilde{B} = \min(n, B)$. There exists a 1-pass streaming algorithm (Algorithm 6) that uses $O\left(\frac{\tilde{B}\log\tilde{B}}{\epsilon}\right)$ space and outputs a $\left(\frac{1}{8}-\epsilon\right)$-approximation to the submodular maximization problem under a knapsack constraint, with $O\left(\frac{\tilde{B}}{\epsilon}\right)$ query per element.

5 Multi-pass streaming algorithm for knapsack

In this section, we develop a multi-pass streaming algorithm for the knapsack constraint using the same algorithmic framework. When $d = 1$, the d-knapsack constraint reduces to the knapsack constraint. As a special case of d-knapsack, we can apply additional techniques, such as multi-pass techniques, to enhance algorithmic performance. Initially, we outline our offline algorithm, which serves as a foundation for subsequent adaptation to a streaming algorithm. As mentioned in the previous section, we also face the challenge of not being able to use enumeration technology to solve the problem of insufficient capacity. Our algorithm modifies the technique from the Greedy+Max algorithm [27] for the monotone submodular maximization subject to the knapsack constraint. Moreover, we adapt and refine methods from [30] to tackle the lack of monotonicity which allows us to maintain two candidate sets.

We begin the algorithm by assuming that we know the maximum value of the elements. We later remove this assumption by reading one more pass of data.

5.1 Offline repeat Greedy+Max algorithm

To begin with, we introduce our adjustments to the Greedy+Max algorithm when the objective function is non-monotone in the offline algorithm. The Greedy algorithm starts with an empty set called G. In each iteration, it selects an item u that has the highest marginal density and can fit into the knapsack. Different from the monotone case, in the non-monotone case, we need to require that the marginal density of u is greater than 0. It works by adding an item with the highest marginal value (instead of density) to each partial solution that is constructed by Greedy. Then, the best solution is chosen from these augmentations. For the needs of subsequent algorithms, we also output the greedy result G at the same time. You can find the implementation in Algorithm 7.

Algorithm 7 Offline Greedy+Max algorithm

Input: Set of elements N, knapsack capacity b, cost function $c(\cdot)$, non-negative non-monotone submodular function f.
$G, A = \emptyset, E = N$.
while $E \neq \emptyset$ **do**
  Remove all elements $e \in E$ with $c(e) > b$.
  $a = \text{argmax}_{e\in E} f(e \mid G)$.
  if $f(A) < f(G \cup a)$ **then**
    $A = G \cup a$.
  Remove all elements $e \in E$ with $f(e \mid G) \leqslant 0$.
  $u = \text{argmax}_{e\in E} f(e \mid G)/c(e)$.
  $G = G \cup u$.
  $b = b - c(u)$.
return A, G

Lemma 2 For each i, let G_i be the first i selected in the construction of G. For the feasible set A returned by Algorithm 7, there exists a partial greedy solution G_{i-1} satisfies $f(A) \geqslant \frac{1}{2}f(G_{i-1} \cup Y)$ for any feasible set Y satisfying $c(Y) \leqslant b$.

Proof First, we declare some symbols that will be used in the proof. Let u_i be the i-th element selected into G. Let a_i be the element with the maximum marginal value to G_{i-1}. Let A_i be the augmentation solution generated by G_{i-1}. Let y_1 be the item with the largest cost in Y. We consider the last item added by the greedy solution before its cost exceeds $b - c(y_1)$.

If the greedy algorithm stops before the cost meets this point, then there exists a j such that $f(G_j \cup Y) \leqslant f(G_j) + \sum_{e\in Y\setminus G_j} f(e \mid G_j) \leqslant f(G_j)$.

Otherwise, we define $c^* \in [0, b - c(y_1)]$ so that $b - c(y_1) - c^*$ is the cost of the greedy solution before this item is taken. Let G_{i-1} be the greedy solution at this point, so that $c(G_{i-1}) \leqslant b - c(y_1) < c(G_i)$.

$$\begin{aligned} &f(G_{i-1} \cup Y) \\ &\quad = f((G_{i-1} \cup y_1) \cup (Y \setminus y_1)) \\ &\quad = f(G_{i-1} \cup y_1) + f(Y \setminus (y_1 \cup G_{i-1}) \mid G_{i-1} \cup y_1) \\ &\quad \leqslant f(G_{i-1} \cup a_i) + \sum_{e\in Y\setminus(y_1\cup G_{i-1})} f(e \mid G_{i-1} \cup y_1) \\ &\quad = f(A_i) + \sum_{e\in Y\setminus(y_1\cup G_{i-1})} c(e) \cdot \frac{f(e \mid G_{i-1} \cup y_1)}{c(e)}, \end{aligned}$$

where the inequality is by submodularity and the last equality is by the definition of marginal density. Since all items in $Y \setminus (y_1 \cup G_{i-1})$ still fit for G_{i-1}, as y_1 is the largest cost item in Y. Since the greedy algorithm always selects the item with the largest marginal density, then

$$\begin{aligned} \max_{e\in Y\setminus(y_1\cup G_{i-1})} \frac{f(e \mid G_{i-1} \cup y_1)}{c(e)} &\leqslant \max_{e\in Y\setminus(y_1\cup G_{i-1})} \frac{f(e \mid G_{i-1})}{c(e)} \\ &\leqslant \frac{f(u_i \mid G_{i-1})}{c(u_i)}. \end{aligned}$$

Hence,

$$f(G_{i-1} \cup Y) \leqslant f(A_i) + \frac{f(u_i \mid G_{i-1})}{c(u_i)} \cdot \sum_{e\in Y\setminus(y_1\cup G_{i-1})} c(e).$$

If $f(A_i) \geqslant \frac{1}{2}f(G_{i-1} \cup Y)$, then we finish the proof because $f(A_i)$ is a lower bound on the value of the augmented solution when the cost of the greedy part is $b - c(y_1) - c^*$. Otherwise,

$$\begin{aligned} \frac{1}{2}f(G_{i-1} \cup Y) &\leqslant \frac{f(u_i \mid G_{i-1})}{c(u_i)} \cdot \sum_{e\in Y\setminus(y_1\cup G_{i-1})} c(e) \\ &\leqslant \frac{f(u_i \mid G_{i-1})}{c(u_i)} \cdot (b - c(y_1)) \\ &\leqslant \sum_{j=1}^{i} f(u_j \mid G_{j-1}) \\ &\leqslant f(G_i), \end{aligned}$$

where the second inequality follows from the fact that $y_1 \in Y$, and the third inequality follows from the greedy procedure in the algorithm. □

The Greedy+Max algorithm itself can not solve with a constant approximation ratio, due to the lack of monotonicity. To address this issue, we use a technique from [30]. The resulting algorithm is depicted as Algorithm 8. It first invokes the Greedy+Max algorithm to obtain a solution S_1. Then it runs the Greedy+Max algorithm again over the remaining elements $N \setminus G$ to obtain another solution S_2. Note that, unlike the offline algorithm framework, because the algorithm may output an augmented solution in the process, we need to deduct the final greedy solution. These two solutions still can not guarantee any constant approximation ratio. To remedy this, the algorithm produces the third solution S_3 by solving the unconstrained submodular maximization problem over G.

Our algorithm finally returns the maximum solution among S_1, S_2, and S_3. We will show that this achieves a 1/6 approximation ratio.

Theorem 7 Algorithm 8 achieves a 1/6 approximation ratio and uses $O(n^2)$ queries.

Proof Let set O be the optimal set such that $f(O) = \text{OPT}$, set G be the first greedy solution output by the Offline Greedy+Max phase, and G_i be the partial greedy solution of G containing the first i elements. For the convenience of writing the proof, we record the greedy result in the second Greedy+Max phase as T and the partial greedy solution as T_j. By Lemma 2, there exist i, j such that

$$f(S_1) \geqslant \frac{1}{2} f(G_{i-1} \cup O),$$
$$f(S_2) \geqslant \frac{1}{2} f(T_{j-1} \cup (O \setminus G)).$$

If $f(G \cap O) \geqslant \delta f(O)$, then by [19] we have $f(S_3) \geqslant \frac{1}{2} f(G \cap O) \geqslant \frac{\delta}{2} f(O)$. If $f(G \cap O) \leqslant \delta f(O)$, then

$$f(S_1) \geqslant \frac{1}{2}(f(G_{i-1} \cup O) + f(G \cap O) - \delta f(O)).$$

Thus, we have that

$$\begin{aligned}
&\max\{f(S_1), f(S_2)\} \\
&\quad \geqslant \frac{1}{2}(f(S_1) + f(S_2)) \\
&\quad \geqslant \frac{1}{4} f(G_{i-1} \cup O) + \frac{1}{4} f(G \cap O) \\
&\qquad + \frac{1}{4} f(T_{j-1} \cup (O \setminus G)) - \frac{\delta}{4} f(O) \\
&\quad \geqslant \frac{1}{4} f(G_{i-1} \cup T_{j-1} \cup O) + \frac{1}{4} f(O \setminus G) \\
&\qquad + \frac{1}{4} f(G \cap O) - \frac{\delta}{4} f(O) \\
&\quad \geqslant \frac{1}{4} f(G_{i-1} \cup T_{j-1} \cup O) + \frac{1}{4} f(O) - \frac{\delta}{4} f(O) \\
&\quad \geqslant \frac{1-\delta}{4} f(O).
\end{aligned}$$

The third and fourth inequalities hold due to submodularity.

Algorithm 8 Offline repeat Greedy+Max algorithm

1: **Input:** Set of elements N, knapsack capacity b, cost function $c(\cdot)$, non-negative non-monotone submodular function f.
2: $S_1, S_2, S_3, G, T = \emptyset$.
3: (S_1, G) =**Offline Greedy+Max**(N, b, c, f).
4: (S_2, T) =**Offline Greedy+Max**$(N \setminus G, b, c, f)$.
5: S_3 = **Unconstrained**(G, f).
6: **return** argmax$\{f(S_1), f(S_2), f(S_3)\}$.

The last inequality holds due to non-negativity. Therefore, the approximation ratio of the returned set is at least $\min\{\delta/2, (1-\delta)/4\}$. Let $\delta = 1/3$. We get that the approximation ratio is at least $1/6$.

Finally, since both the algorithm from [19] using $O(|G|^2)$ queries and Algorithm 7 make $O(n^2)$ queries, Algorithm 8 also makes $O(n^2)$ queries in total. □

5.2 Multi-pass streaming algorithm repeat Greedy+Max

We are introducing the multi-pass streaming algorithm, which is represented as Algorithm 9. We first give the algorithm under the assumption that it is given a parameter m, which is the max value of the element. We can remove this assumption using another pass before the algorithm. During the algorithm, we use a multi-pass thresholding technique to simulate the greedy process. Then we use 1-pass to obtain the augmented solution for the partial greedy solution. The final output will be the best among all augmented solutions. For the needs of subsequent algorithms, we also output the greedy result G at the same time. As discussed in the description of our techniques use $O(\log b/\epsilon)$ passes over the data to simulate the execution of offline Greedy+Max approximately.

Lemma 3 For the feasible set A returned by Algorithm 9, there is partial greedy solution G_{i-1} (defined in line 9) satisfies $f(A) \geqslant (\frac{1}{2} - \epsilon) f(G_{i-1} \cup Y)$ for any feasible set Y satisfying $c(Y) \leqslant b$.

Algorithm 9 Multi-pass streaming algorithm

1: **Input**: Stream $e_1, \ldots, e_n$, knapsack capacity b, cost function $c(\cdot)$, non-negative submodular function f, the max value of the element $m = \max_{e \in N} f(e)$, $\epsilon > 0$.
2: $G, A = \emptyset$, $\tau = m$.
3: **while** $\tau > \frac{m}{b}$ **do** ▷ Thresholding stage
4: Take a new pass over the stream.
5: **for** each read item e **do**
6: **if** $f(e \mid G)/c(e) \geqslant \tau$ and $c(e \cup G) \leqslant b$ **then**
7: $G = G \cup \{e\}$.
8: $\tau = \tau/(1+\epsilon)$.
9: For each i, let G_i be the first i items selected in the construction of G (line 7) above and let $a_i = \emptyset$.
10: Take a pass over the stream.
11: **for** each read item e **do** ▷ Augmentation stage
12: $l = \max\{i \mid c(G_i) + c(e) \leqslant b\}$.
13: **for** $i = 1$ to $l + 1$ **do**
14: **if** $f(G_{i-1} \cup a_i) < f(G_{i-1} \cup e)$ **then**
15: $a_i = \{e\}$.
16: $A = \text{argmax}\, f(G_{i-1} \cup a_i)$.
17: **return** A, G

Proof Like the previous proof in Lemma 2, we first state some definitions. Let u_i be the i-th element selected. Let a_i be the element with the maximum marginal value to G_{i-1} and $A_i = G_{i-1} \cup a_i$ be the augmentation solution generated by G_{i-1}. Let set O be the optimal set such that $f(O) = \text{OPT}$. Let y_1 be the item of the largest cost in Y. Consider the last item added by the greedy solution in the thresholding stage before the cost of this solution exceeds $b - c(y_1)$.

If the thresholding stage stops before the cost meets this point, by the thresholding there exists a j such that

$$\begin{aligned} f(G_j \cup Y) &\leqslant f(G_j) + \sum_{e \in Y \setminus G_j} f(e \mid G_j) \\ &\leqslant f(G_j) + (1+\epsilon) \cdot \frac{m}{b} \cdot c(Y \setminus G_j) \\ &\leqslant f(G_j) + (1+\epsilon) \cdot m, \end{aligned}$$

where the first inequality is by submodularity and the second inequality is by the threshold at termination. Thus, we have $f(G_j \cup Y) \leqslant f(G_j) + (1+\epsilon) f(A_1)$ and $f(A) \geqslant (\frac{1}{2} - \epsilon) f(G_j \cup Y)$.

Otherwise, we define c^* so that $b - c(y_1) - c^*$ is the cost of the greedy solution before this item is taken. Let G_{i-1} be the greedy solution at this point, so that $c(G_{i-1}) \leqslant b - c(y_1) < c(G_i)$.

$$\begin{aligned} & f(G_{i-1} \cup Y) \\ &= f((G_{i-1} \cup y_1) \cup (Y \setminus y_1)) \\ &= f(G_{i-1} \cup y_1) + f(Y \setminus (y_1 \cup G_{i-1}) \mid G_{i-1} \cup y_1) \\ &\leqslant f(G_{i-1} \cup a_i) + \sum_{e \in Y \setminus (y_1 \cup G_{i-1})} f(e \mid G_{i-1} \cup y_1) \\ &= f(A_i) + \sum_{e \in Y \setminus (y_1 \cup G_{i-1})} c(e) \cdot \frac{f(e \mid G_{i-1} \cup y_1)}{c(e)}, \end{aligned}$$

where the second inequality is by submodularity and the last equality is by the definition of marginal density. Since all items in $Y \setminus (y_1 \cup G_{i-1})$ still fit, as y_1 is the largest item in Y. In all passes, the thresholding algorithm always selects an element that gives $\frac{1}{1+\epsilon}$-approximation of the highest possible marginal density. Since the greedy algorithm always selects the item with the largest marginal density, then

$$\begin{aligned} & \max_{e \in Y \setminus (y_1 \cup G_{i-1})} \frac{f(e \mid G_{i-1} \cup y_1)}{c(e)} \\ \leqslant & \max_{e \in Y \setminus (y_1 \cup G_{i-1})} \frac{f(e \mid G_{i-1})}{c(e)} \\ \leqslant & (1+\epsilon) \cdot \frac{f(u_i \mid G_{i-1})}{c(u_i)}. \end{aligned}$$

Hence,

$$\begin{aligned} & f(G_{i-1} \cup Y) \\ & \leqslant f(A_{i-1}) + (1+\epsilon) \frac{f(u_i \mid G_{i-1})}{c(u_i)} \cdot \sum_{e \in Y \setminus (y_1 \cup G_{i-1})} c(e). \end{aligned}$$

If $f(A_{i-1}) \geqslant \frac{1}{2} f(G_{i-1} \cup Y)$, then we have $f(A) \geqslant f(A_{i-1}) \geqslant \frac{1}{2} f(G_{i-1} \cup Y)$. Otherwise,

$$\begin{aligned} & \frac{1}{2} f(G_{i-1} \cup Y) \\ & \leqslant (1+\epsilon) \cdot \frac{f(u_i \mid G_{i-1})}{c(u_i)} \cdot \sum_{e \in Y \setminus (y_1 \cup G_{i-1})} c(e) \\ & \leqslant (1+\epsilon) \cdot \frac{f(u_i \mid G_{i-1})}{c(u_i)} \cdot (b - c(y_1)) \\ & \leqslant (1+\epsilon)^2 \sum_{j=1}^{i} f(u_j \mid G_{j-1}) \\ & \leqslant (1+\epsilon)^2 f(G_i), \end{aligned}$$

where the second inequality follows from the fact that $y_1 \in Y$ and the third inequality follows from the greedy procedure in the algorithm. Note that $\epsilon > 0$, then we finish the proof of the lemma by $f(A) \geqslant f(G_i) \geqslant (\frac{1}{2} - \epsilon) f(G_{i-1} \cup Y)$. □

We present the complete multi-pass streaming algorithm. First, we need to run 1-pass to find the maximum value of the elements $m = \max_{e \in N} f(e)$. Then we apply the streaming algorithm framework for non-monotone submodular maximization, and we need to deduct the final greedy solution. The resulting algorithm is depicted as Algorithm 10. Our algorithm finally returns the maximum solution among S_1, S_2, and S_3. We will show that this achieves a $1/6 - \epsilon$ approximation ratio using $O(\log b/\epsilon)$ passes.

Theorem 8 For any fixed $\epsilon > 0$, Algorithm 10 achieves a $1/6 - \epsilon$ approximation ratio and uses $O(\log b/\epsilon)$ passes, stores at most $2\tilde{B}$ elements, and makes at most $O(\log b/\epsilon)$ queries per elements.

Proof Let $O \in \text{argmax}\{f(S) \mid c(S) \leqslant b\}$. Set G be the first greedy solution output by the Offline Greedy+Max phase, and G_i be the partial greedy solution of G containing the first i elements. For the convenience of writing the proof, we record the greedy result in the second Greedy+Max phase as T and the partial greedy solution as T_j. By Lemma 3, there exist i, j such that

Algorithm 10 Multi-pass streaming repeat Greedy+Max algorithm

Input: stream $N = e_1, \ldots, e_n$, knapsack capacity b, cost function $c(\cdot)$, non-negative submodular function f, $\epsilon > 0$.
$S_1, S_2, S_3, G, T = \emptyset$.
$m_1, m_2 = 0$.
Take a new pass over the stream N.
for each read item e **do**
    $m_1 = \max\{m_1, f(e)\}$.
(S_1, G) = **Multi-Pass Streaming Algorithm**(streaming(N), b, c, f, m_1, ϵ).
Take a new pass over the stream $N \setminus G$.
for each read item e **do**
    $m_2 = \max\{m_2, f(e)\}$.
(S_2, T) = **Multi-Pass Streaming Algorithm**(streaming($N \setminus G$), b, c, f, m_2, ϵ).
S_3 = **Unconstrained**(G, f).
return argmax$\{f(S_1), f(S_2), f(S_3)\}$.

$$f(S_1) \geqslant (\frac{1}{2}-\epsilon)f(G_{i-1}\cup O),$$
$$f(S_2) \geqslant (\frac{1}{2}-\epsilon)f(T_{j-1}\cup (O\setminus G)).$$

If $f(G\cap O)\geqslant \delta f(O)$, then $f(S_3)\geqslant \frac{1}{2}f(G\cap O)\geqslant \frac{\delta}{2}f(O)$ by [19]. If $f(G\cap O)\leqslant \delta f(O)$, then

$$f(S_1) \geqslant (\frac{1}{2}-\epsilon)(f(G_{i-1}\cup O)+f(G\cap O)-\delta f(O)).$$

Thus, we have that

$$\begin{aligned}
&\max\{f(S_1),f(S_2)\}\\
&\quad\geqslant \frac{1}{2}(f(S_1)+f(S_2))\\
&\quad\geqslant \frac{1-\epsilon}{4}f(G_{i-1}\cup O)+\frac{1-\epsilon}{4}f(G\cap O)\\
&\qquad+\frac{1-\epsilon}{4}f(T_{j-1}\cup (O\setminus G))-\frac{(1-\epsilon)\delta}{4}f(O)\\
&\quad\geqslant \frac{1-\epsilon}{4}f(G_{i-1}\cup T_{j-1}\cup O)+\frac{1-\epsilon}{4}f(O\setminus G)\\
&\qquad+\frac{1-\epsilon}{4}f(G\cap O)-\frac{(1-\epsilon)\delta}{4}f(O)\\
&\quad\geqslant \frac{1-\epsilon}{4}\left(f(G_{i-1}\cup T_{j-1}\cup O)+f(O)-\delta f(O)\right)\\
&\quad\geqslant \frac{(1-\epsilon)(1-\delta)}{4}f(O).
\end{aligned}$$

The third and fourth inequalities hold due to submodularity. The last inequality holds due to non-negativity. Therefore, the approximation ratio of the returned set is at least $\min\{\delta/2,(1-\epsilon)(1-\delta)/4\}$. Let $\delta=(1-\epsilon)/(3-\epsilon)$. We get that the approximation ratio is at least $(1-\epsilon)/6$.

Finally, since the algorithm from Algorithm 9 make $O(\log b/\epsilon)$ passes, Algorithm 10 also makes $O(\log b/\epsilon)$ passes in total. At most $O(\tilde{B})$ elements are stored by the multi-pass streaming algorithm. The multi-pass streaming algorithm makes at most $O(\log b/\epsilon)$ queries per element. □

6 Conclusions and future work

We have created a streaming algorithm framework for submodular maximization with a general (possibly non-monotone) objective function. This framework enables us to propose 1-pass streaming algorithms that provide improved approximation ratios for non-monotone submodular maximization under a *d*-knapsack constraint or a knapsack constraint. Additionally, we have developed a multi-pass streaming algorithm for knapsack constraints. All of these algorithms are efficient and deterministic.

For the *d*-knapsack constraint, a question that we are more concerned about next is whether the approximation ratio of the streaming algorithm can be further improved. The current analysis of the cost of the *d*-knapsack is still a bit rough. We believe that through more detailed discussion, better results can be achieved. We also believe that if we can find a way to balance the cost of each dimension, the approximation ratio of this problem can be improved through a multi-pass streaming algorithm.

As for the single knapsack constraint, there is an unresolved issue on how to enhance the approximation of 1-pass streaming algorithms. Another future research direction is whether increasing the number of feasible solutions of candidate sets will improve the approximation ratio. In theory, the knapsack constraint can yield the same approximation ratio as the cardinality constraint, but this has not been achieved even in the monotone case.

When the objective function is non-monotone, though our algorithms improve the best-known deterministic algorithms, their approximation ratios are still worse than the best randomized algorithms. It is very interesting to fill these gaps.

Acknowledgements All author contributed equally to this paper and the order of author's names is alphabetical. This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62325210 and 62272441).

Competing interests Xiaoming Sun is an Editorial Board member of the journal and a co-author of this article. To minimize bias, they were excluded from all editorial decision-making related to the acceptance of this article for publication. The remaining authors declare no conflict of interest.

References

1. Kempe D, Kleinberg J, Tardos É. Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003, 137−146
2. Das A, Kempe D. Algorithms for subset selection in linear regression. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing. 2008, 45−54
3. Khanna R, Elenberg E R, Dimakis A G, Negahban S N, Ghosh J. Scalable greedy feature selection via weak submodularity. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. 2017, 1560−1568
4. Iyer R, Bilmes J. Algorithms for approximate minimization of the difference between submodular functions, with applications. In: Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence. 2012, 407−417
5. Iyer R, Bilmes J. Submodular optimization with submodular cover and submodular knapsack constraints. In: Proceedings of the 26th International Conference on Neural Information Processing Systems. 2013, 2436−2444
6. Jin L B, Zhang L, Zhao L. Max-difference maximization criterion: a feature selection method for text categorization. Frontiers of Computer Science, 2023, 17(1): 171337
7. Liang J, Zhang Y, Chen K, Qu B, Yu K, Yue C, Suganthan P N. An evolutionary multiobjective method based on dominance and decomposition for feature selection in classification. Science China Information Sciences, 2024, 67(2): 120101
8. Krause A, Guestrin C. Submodularity and its applications in optimized information gathering. ACM Transactions on Intelligent Systems and Technology, 2011, 2(4): 32
9. Nemhauser G L, Wolsey L A. Best algorithms for approximating the maximum of a submodular set function. Mathematics of Operations Research, 1978, 3(3): 177−188
10. Feige U. A threshold of ln *n* for approximating set cover. Journal of the ACM, 1998, 45(4): 634−652
11. Nemhauser G L, Wolsey L A, Fisher M L. An analysis of approximations for maximizing submodular set functions—I. Mathematical Programming, 1978, 14(1): 265−294
12. Khuller S, Moss A, Naor J. The budgeted maximum coverage problem. Information Processing Letters, 1999, 70(1): 39−45
13. Sviridenko M. A note on maximizing a submodular set function subject

to a knapsack constraint. Operations Research Letters, 2004, 32(1): 41−43
14. Vondrak J. Optimal approximation for the submodular welfare problem in the value oracle model. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing. 2008, 67−74
15. Kulik A, Shachnai H, Tamir T. Maximizing submodular set functions subject to multiple linear constraints. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms. 2009, 545−554
16. Feldman M, Naor J, Schwartz R. A unified continuous greedy algorithm for submodular maximization. In: Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science. 2011, 570−579
17. Buchbinder N, Feldman M. Deterministic algorithm and faster algorithm for submodular maximization subject to a matroid constraint. In: Proceedings of the 65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024. 2024
18. Buchbinder N, Feldman M. Constrained submodular maximization via new bounds for DR-submodular functions. In: Proceedings of the 56th Annual ACM Symposium on Theory of Computing. 2024, 1820−1831
19. Buchbinder N, Feldman M. Deterministic algorithms for submodular maximization problems. ACM Transactions on Algorithms, 2018, 14(3): 32
20. Sun X, Zhang J, Zhang S, Zhang Z. Improved deterministic algorithms for non-monotone submodular maximization. In: Proceedings of Computing and Combinatorics: 28th International Conference. 2022, 496−507
21. Badanidiyuru A, Mirzasoleiman B, Karbasi A, Krause A. 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, 671−680
22. Feldman M, Norouzi-Fard A, Svensson O, Zenklusen R. 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, 1363−1374
23. Alaluf N, Ene A, Feldman M, Nguyen H L, Suh A. An optimal streaming algorithm for submodular maximization with a cardinality constraint. Mathematics of Operations Research, 2022, 47(4): 2667−2690
24. Huang C C, Kakimura N, Yoshida Y. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. Algorithmica, 2020, 82(4): 1006−1032
25. Huang C C, Kakimura N. Improved streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. Algorithmica, 2021, 83(3): 879−902
26. Huang C C, Kakimura N. Multi-pass streaming algorithms for monotone submodular function maximization. Theory of Computing Systems, 2022, 66(1): 354−394
27. Yaroslavtsev G, Zhou S, Avdiukhin D. “Bring your own greedy”+max: near-optimal 1/2-approximations for submodular knapsack. In: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics. 2020, 3263−3274
28. Kumar R, Moseley B, Vassilvitskii S, Vattani A. Fast greedy algorithms in mapReduce and streaming. ACM Transactions on Parallel Computing, 2015, 2(3): 14
29. Yu Q, Xu E L, Cui S. Submodular maximization with multi-knapsack constraints and its applications in scientific literature recommendations. In: Proceedings of 2016 IEEE Global Conference on Signal and Information Processing. 2016, 1295−1299
30. Gupta A, Roth A, Schoenebeck G, Talwar K. Constrained non-monotone submodular maximization: offline and secretary algorithms. In: Proceedings of the 6th International Workshop on Internet and Network Economics. 2010, 246−257
31. Lee J, Mirrokni V S, Nagarajan V, Sviridenko M. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. SIAM Journal on Discrete Mathematics, 2010, 23(4): 2053−2078
32. Fadaei S, Fazli M A, Safari M A. Maximizing non-monotone submodular set functions subject to different constraints: combined algorithms. Operations Research Letters, 2011, 39(6): 447−451
33. Buchbinder N, Feldman M. Constrained submodular maximization via a nonsymmetric technique. Mathematics of Operations Research, 2019, 44(3): 988−1005
34. Sun X, Zhang J, Zhang S, Zhang Z. Improved deterministic algorithms for non-monotone submodular maximization. Theoretical Computer Science, 2024, 984: 114293

Xiaoming Sun is a researcher at the Institute of Computing Technology, Chinese Academy of Sciences, China, leading the Laboratory for Quantum Computation and Theoretical Computer Science. His research interests include approximation algorithms, computational complexity, and quantum computing.

Jialin Zhang is a researcher and a doctoral supervisor at the Institute of Computing Technology, Chinese Academy of Sciences, China. Her research topics include submodular optimization, approximation algorithms, online algorithms, quantum computing, and algorithmic game theory.

Shuo Zhang is a doctoral student at the Institute of Computing Technology, Chinese Academy of Sciences, China, supervised by Professor Jialin Zhang. She mainly works on theoretical computers, algorithm design and optimization, particularly on the topic of submodular optimization.