Streaming Algorithms for Maximizing Non-submodular Functions on the Integer Lattice

Bin Liu[1(✉)], Zihan Chen[1], Huijuan Wang[2], and Weili Wu[3]

[1] School of Mathematical Sciences, Ocean University of China, Qingdao, China
binliu@ouc.edu.cn
[2] School of Mathematics and Statistics, Qingdao University, Qingdao, China
[3] Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA

Abstract. Submodular functions play a key role in combinatorial optimization field. The problem of maximizing submodular and non-submodular functions on the integer lattice has received a lot of recent attention. In this paper, we study streaming algorithms for the problem of maximizing a monotone non-submodular functions with cardinality constraint on the integer lattice. For a monotone non-submodular function $f : \mathbf{Z}_+^n \to \mathbf{R}_+$ defined on the integer lattice with diminishing-return (DR) ratio γ, we present a one pass streaming algorithm that gives a $(1 - \frac{1}{2^\gamma} - \epsilon)$-approximation, requires at most $O(k\epsilon^{-1} \log k/\gamma)$ space and $O(\epsilon^{-1} \log k/\gamma \cdot \log \|\mathbf{B}\|_\infty)$ update time per element. To the best of our knowledge, this is the first streaming algorithm on the integer lattice for this constrained maximization problem.

Keywords: Streaming algorithm · Cardinality constraint · Non-submodular maximization · Integer lattice

1 Introduction

A set function $f : 2^E \to \mathbf{R}$ with a ground set E is called *submodular* if for any $A, B \subseteq E$, it holds that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. There is an equivalent definition of submodularity which called diminishing marginal return property, i.e. for any $S \subseteq T \subseteq E$ and $e \in E \setminus T$, we have $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$. We say a set function f is *monotone* if for any $S \subseteq T \subseteq E$, it holds $f(S) \leq f(T)$. Submodular functions play a key role in combinatorial optimization, as they capture many instances such as rank functions of matroids, cuts functions of graphs and covering functions [1,8]. The few decades have seen a proliferation of works on submodular maximization. In particular, there are many algorithms for maximizing a submodular function subject to various

This work was supported in part by the National Natural Science Foundation of China (11971447, 11871442), and the Fundamental Research Funds for the Central Universities.

D. Mohaisen and R. Jin (Eds.): CSoNet 2021, LNCS 13116, pp. 3–14, 2021.
https://doi.org/10.1007/978-3-030-91434-9_1

constraints such as greedy algorithms, random sampling algorithms and local search algorithms which achieve constant factor approximation guarantees.

Due to the phenomenon of big data, constrained submodular maximization has found many new applications, including data summarization [2,3], generalized assignment [4] and influence maximization in social networks [5–7]. In some of the above applications, the amount of input data is much larger than the data that the individual computers can store. This issue motivates us to use streaming computation approach to process the data which uses only a small amount of memory and only a single pass over the data ideally. That is, when each item in the ground set $E = \{e_1, ..., e_n\}$ arrives, the streaming algorithm must decide whether to keep the current item before the arrival of the next item. Generally, there are four indicators to measure the streaming algorithm, which are approximation ratio, query complexity, memory complexity and the number of passes to scan all data. Up to now, much work has been done on submodular maximization in the streaming model [9–12].

Set functions are powerful tools to describe the problem of elements selection. However, in practice, we sometimes face situations that cannot be solved with set functions such as problems that allow multiple choices of an element in the ground set. Thus it is nature for us to generalize a submodular function from set E to integer lattice $\mathbf{Z}^E$. Recently, much work has studied the generalization of submodular functions on bounded integer lattice [13–15]. But in instances, functions with many problems are non-submodular [16,17]. To solve this problem, some parameters were proposed to describe the closeness of non-submodular function and submodular function, such as diminishing-return ratio, submodularity ratio and generic submodularity ratio [16,21,22]. In this paper, we describe a streaming algorithm for non-submodular maximization with a cardinality constraint on the integer lattice. Let $\mathbf{B} \in \mathbf{Z}_+^n$ be an integer vector and $[\mathbf{B}] = \{\boldsymbol{x} \in \mathbf{Z}_+^n | 0 \leq x(i) \leq B(i), \forall 1 \leq i \leq n\}$ be an integer lattice domain, where $x(i)$ denotes the i-th component of vector $\boldsymbol{x}$. Our problem is described as follows

$$\begin{aligned} &\max f(\boldsymbol{x}) \\ &\text{s. t. } \|\boldsymbol{x}\|_1 \leq k, \\ &\quad \boldsymbol{x} \leq \mathbf{B}. \end{aligned} \tag{1}$$

where $f : \mathbf{Z}_+^E \to \mathbf{R}$ is a non-negative monotone non-submodular function with $f(\mathbf{0}) = 0$, and k is a positive integer.

Our Contribution. In this paper, we focus on maximizing non-submodular functions subject to a cardinality constraint on the integer lattice. Inspired by the Sieve-Streaming algorithm introduced by Badanidiyuru et al. [9], we propose a one pass streaming algorithm. For each arriving item with its copies, we employ a modified binary search algorithm (cf. Sect. 2) to determine the amount of the current item that should be kept. Finally, we give a $(1 - \frac{1}{2^\gamma} - \epsilon)$-approximation algorithm with memory $O(k\epsilon^{-1} \log k/\gamma)$ and update time $O(\epsilon^{-1} \log k/\gamma \log l_{max})$ per element, where γ is the diminishing return (DR) ratio (cf. Definition 1) for non-submodular functions on the integer lattice. To the best of our knowledge, this is the first streaming algorithm on the integer lattice for this constrained maximization problem.

1.1 Additional Related Work

The research on submodular and non-submodular optimization is too extensive to give a comprehensive description. In the following, we only describe the related work of this paper.

Integer Lattice. As a generalization of submodular set functions, a function is called *lattice submodular function* if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{Z}^E$, we have $f(\boldsymbol{x}) + f(\boldsymbol{y}) \geq f(\boldsymbol{x} \vee \boldsymbol{y}) + f(\boldsymbol{x} \wedge \boldsymbol{y})$, where $\vee$ and $\wedge$ are coordinate-wise max and min. However, unlike set functions, submodularity on integer lattice is not equivalent to diminishing returns property. A function $f : \mathbf{Z}^E \rightarrow \mathbf{R}$ satisfying $f(\boldsymbol{x} + \mathcal{X}_e) - f(\boldsymbol{x}) \geq f(\boldsymbol{y} + \mathcal{X}_e) - f(\boldsymbol{y})$ is called *diminishing-return(DR) submodular function* for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{Z}^E$ with $\boldsymbol{x} \leq \boldsymbol{y}$ and $e \in E$, where $\mathcal{X}_e$ denotes the unit vector with coordinate e being 1 and other components are 0. Note that lattice submodularity is weaker than DR-submodularity in general [14]. For maximizing a monotone submodular function subject to a knapsack constraint on the integer lattice, Soma et al. [14] proposed a pseudo-polynomial-time algorithm with approximation ratio $1 - 1/e$. Later, the running time is significantly improved. Soma et al. [13] proposed polynomial-time algorithms which use threshold greedy technique to achieve an arbitrarily close to $1 - 1/e$ approximation for both lattice and DR-submodular maximization under a cardinality constraint, DR-submodular maximization under a polymatroid constraint and a knapsack constraint.

Streaming Model. For monotone submodular maximization subject to a cardinality constraint, Badanidiyuru et al. [9] presented the first one pass 1/2-approximation streaming algorithm named Sieve-Streaming with memory $O(k \log k/\epsilon)$ and update time $O(\log k/\epsilon)$ per element. In contrast, Buchbinder et al. [10] designed a streaming algorithm with a lower ratio of 1/4 but an improved memory complexity $O(k)$. For this problem, Norouzi-Fard et al. [11] proved that with memory $O(n/k)$, the best approximation ratio of the one pass streaming algorithm for this problem is 1/2. Kazemi et al. [12] described a streaming algorithm called Sieve Streaming++, which obtained a approximation of 1/2 with memory $O(k/\epsilon)$. Following this vain, [17] studied the non-submodular function with cardinality constraint and give a $(1 - \frac{1}{2^\gamma} - \epsilon)$-approximation streaming algorithm with memory $O(k \log(k/\gamma)/\epsilon)$ and update time $O(\log(k/\gamma)/\epsilon)$ per element.

In the streaming setting, besides the set functions we mentioned above, there are also works considering submodular maximization on the integer lattice. Zhang et al. [19,20] gave $(1/2 - \epsilon)$ algorithms for DR-submodular and lattice submodular maximization with cardinality constraint, respectively. For DR-submodular maximization subject to knapsack constraint, Tan et al. [18] proposed a $(1/3 - \epsilon)$-approximation algorithm with a single pass.

Non-submodular Functions. Das and Kempe [21] gave the concept of *submodularity ratio* γ_s to describe how close a function is from being submodular. Kuhnle et al. [16] proposed the *diminishing-return (DR) ratio* γ_d on the integer lattice, and generalized the definition of submodularity ratio γ_s in [21] from set

functions to the integer lattice. Later, Nong et al. [22] defined the *generic submodularity ratio* γ to measure the diminishing return property of a set function. DR ratio defined in [16] is the extension of generic submodularity ratio defined in [22] from set to the integer lattice. For maximizing a monotone non-submodular function subject to cardinality constraint, Nong et al. [22] proved that standard greedy algorithm achieves a $(1 - e^{-\gamma})$-approximation with query complexity $O(nk)$. Wang et al. [17] introduced a streaming algorithm with approximation $1 - \frac{1}{2^{\gamma}} - \epsilon$. In addition, Kuhnle et al. [16] utilized threshold greedy technique to obtain a algorithm for maximizing monotone non-submodular functions on the integer lattice whose approximation ratio arbitrarily approaching $(1 - e^{-\gamma_s \gamma_d})$.

The rest of this paper is organized as follows. The necessary notations and subproblems that our algorithms need to solve are introduced in Sect. 2. In Sect. 3 we first propose a streaming algorithm with known optimal value. Then in Sect. 4 we introduce a streaming algorithm with known value of the unit standard vector. Finally, we present the one pass streaming algorithm for the non-submodular functions in Sect. 5.

2 Preliminaries

In this section, we will define the DR ratio and introduce some notations and properties about non-submodular function on a bounded integer lattice.

Notations. Denote $\mathbf{Z}_+$ and $\mathbf{R}_+$ be the non-negative integers and non-negative reals, respectively. Let E represent a finite ground set of size n. For each $e \in E$, we use $x(e)$ to denote the component of a vector $\boldsymbol{x} \in Z_+^E$ corresponding to element e. Let $\mathbf{B} \in \mathbf{Z}_+^n$ be an integer vector and $[\mathbf{B}] = \{\boldsymbol{x} \in \mathbf{Z}_+^n | 0 \leq x(i) \leq B(i), \forall 1 \leq i \leq n\}$ be an integer lattice domain. Specially, $[k] = \{1, 2, ..., k\}$ for any integer $k \in \mathbf{Z}_+$. For any $e_i \in E$, let $\mathcal{X}_i$ denote the unit vector in which the i-th component is 1 and the other components are 0. The zero vector is represented by $\mathbf{0}$. For vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{Z}_+^E$, we define $f_{\boldsymbol{y}}(\boldsymbol{x}) = f(\boldsymbol{y} + \boldsymbol{x}) - f(\boldsymbol{y})$. For a vector $\boldsymbol{x} \in \mathbf{Z}_+^E$, let $supp^+(\boldsymbol{x})$ be the set $\{e \in E | x(e) > 0\}$, and $\{\boldsymbol{x}\}$ be the multiset corresponding to vector $\boldsymbol{x}$. Finally, we define $\boldsymbol{x} \vee \boldsymbol{y}$ to be the vector whose i-th coordinate is $\max\{\boldsymbol{x}(i), \boldsymbol{y}(i)\}$, and $\boldsymbol{x} \wedge \boldsymbol{y}$ to be the vector whose i-th coordinate is $\min\{\boldsymbol{x}(i), \boldsymbol{y}(i)\}$.

Using this notations we can now define DR ratio as follows.

Definition 1 (DR Ratio [16]). *Let function $f : \boldsymbol{Z}_+^E \to \boldsymbol{R}$, the diminishing-return (DR) ratio of f, γ, is the maximum value in $[0, 1]$ such that for any $e \in E$, and for all $\boldsymbol{x} \leq \boldsymbol{y}$, such that $\boldsymbol{y} + \mathcal{X}_e \leq \boldsymbol{B}$,*

$$\gamma f_{\boldsymbol{y}}(\mathcal{X}_e) \leq f_{\boldsymbol{x}}(\mathcal{X}_e).$$

Next, we describe a subproblem of our algorithm need to solve, namely, binary search pivot subproblem.

BinarySearchPivot. Integer lattice can be represented as a multiset, where elements can be contained repeatedly. In a streaming algorithm on the integer lattice, when element e and its copies arrive, any l satisfying $(1) f_{\boldsymbol{y}}(l\mathcal{X}_e) \geq$

$l\tau; (2) f_{\boldsymbol{y}}((l+1)\mathcal{X}_e) < (l+1)\tau$ is named as a *pivot* with respect to $\boldsymbol{y}, e, \tau$. The subproblem is described as follows: given a threshold τ, determine a valid pivot which satisfies both (1) and (2). In the literature, Soma et al. [13] studied the maximization of DR-submodular functions f on the integer lattice, and they used the standard Binary Search algorithm to obtain a valid pivot. Meanwhile, when f is non-submodular, BinarySearchPivot algorithm of Kuhnle et al. [16] ensures the average marginal contribution of elements added exceeds τ. In this paper, we focus on non-submodular functions, thus we analyze our algorithms using the BinarySearchPivot as a subroutine. The full algorithm for BinarySearchPivot is given in Appendix A. Next, we use the following lemma of Kuhnle et al. [16].

Lemma 1 ([16]). *BinarySearchPivot finds a valid pivot* $l \in \{0, .., l_{max}\}$ *in* $O(\log l_{max})$ *queries of* f*, where* $l_{max} = \min\{B(e) - y(e), k - \|\boldsymbol{y}\|_1\}$.

3 Streaming Algorithm with Known *OPT*

In this section, we assume that the optimal value of the problem (1) is known. Then we present an algorithm that obtains a constant ratio with polynomial query complexity. Procedure for BinarySearchPivot is desired; see section 2 for an analysis of this subproblem.

Overview of Algorithm. Inspired by [9], we present a threshold greedy algorithm in the streaming model. Suppose that a parameter v with $\lambda OPT \leq v \leq OPT$ is known, where $\lambda \in [0, 1]$. When element e_i and its $B(e_i)$ copies arrive, we utilize the BinarySearchPivot algorithm with threshold $\tau = \frac{\gamma v}{2\gamma k}$ to obtain a valid pivot l_i. That is, when algorithm runs to element e_i, we employ Algorithm BinarySearchPivot to obtain a appropriate l_i which satisfies

$$\frac{f_{\boldsymbol{y}}(l_i\mathcal{X}_i)}{l_i} \geq \frac{\gamma v}{2\gamma k}.$$

and

$$\frac{f_{\boldsymbol{y}}((l_i+1)\mathcal{X}_i)}{l_i+1} < \frac{\gamma v}{2\gamma k}.$$

Finally, we add $l_i\mathcal{X}_i$ to the current solution $\boldsymbol{y}$.

Algorithm 1. Streaming-Know-OPT

Require: function f, cardinality k, ground set E, $\mathbf{B}$ and v such that $\lambda OPT \leq v \leq OPT$.

$\boldsymbol{y} \leftarrow \mathbf{0}$
for $i = 1, 2, ..., n$ **do**
  if $\|\boldsymbol{y}\|_1 \leq k$ **then**
    $l_i \leftarrow$BinarySearchPivot$(f, \boldsymbol{y}, B(e_i), e_i, k, \frac{\gamma v}{2\gamma k})$
    $\boldsymbol{y} \leftarrow \boldsymbol{y} + l_i\mathcal{X}_i$
  end if
end for
return $\boldsymbol{y}$

Lemma 2. *For the i-th iteration, Algorithm BinarySearchPivot considers the element e_i and its copies $B(e_i)$. Let $\boldsymbol{y}_i$ be the current solution $\boldsymbol{y}$ at the end of iteration i of Algorithm 1. Then for each vector $\boldsymbol{y}_i \leq \boldsymbol{v} \leq \boldsymbol{w} \leq \boldsymbol{B}$, where $i = 1, 2, ..., n$, we have*

$$f(\boldsymbol{w}) - f(\boldsymbol{v}) \leq \frac{1}{\gamma} \sum\nolimits_{e_i \in supp^+\{\boldsymbol{w}-\boldsymbol{v}\}} (w(e_i) - v(e_i)) f_{\boldsymbol{y}_i}(\mathcal{X}_i).$$

The above lemma is proven in Appendix B. From Lemma 2, it is clearly to see that for any vector $\boldsymbol{y} \leq \boldsymbol{v} \leq \boldsymbol{w} \leq \boldsymbol{B}$, we also have

$$f(\boldsymbol{w}) - f(\boldsymbol{v}) \leq \frac{1}{\gamma} \sum_{e_i \in \{\boldsymbol{w}-\boldsymbol{v}\}} f_{\boldsymbol{y}}(\mathcal{X}_i). \tag{2}$$

Lemma 3. *Let $\boldsymbol{y}_i$ be the vector $\boldsymbol{y}$ following the i-th update of Algorithm 1, then it satisfies*

$$f(\boldsymbol{y}_i) \geq \frac{\gamma v \|\boldsymbol{y}_i\|_1}{2\gamma k}. \tag{3}$$

Proof. The proof is by induction. First, since f is normalized, we have $f(\mathbf{0}) = 0$. Next, we suppose that for vector $\boldsymbol{y}_{i-1}$ the result holds, that is, $f(\boldsymbol{y}_{i-1}) \geq \frac{\gamma v \|\boldsymbol{y}_{i-1}\|_1}{2\gamma k}$. Let l_i be the valid pivot returned by Algorithm BinarySearchPivot during the i-th iteration of Algorithm 1, then we have $\boldsymbol{y}_i = \boldsymbol{y}_{i-1} + l_i \mathcal{X}_i$. We now show inequation 3 holds.

By Algorithm BinarySearchPivot we know

$$f(\boldsymbol{y}_{i-1} + l_i \mathcal{X}_i) - f(\boldsymbol{y}_{i-1}) \geq l_i \frac{\gamma v}{2\gamma k}. \tag{4}$$

Using the above assumption and the equality 4, we have

$$\begin{aligned} f(\boldsymbol{y}_{i-1} + l_i \mathcal{X}_i) &\geq f(\boldsymbol{y}_{i-1}) + l_i \frac{\gamma v}{2\gamma k} \\ &\geq \frac{\gamma v \|\boldsymbol{y}_{i-1}\|_1}{2\gamma k} + l_i \frac{\gamma v}{2\gamma k} \\ &= \frac{\gamma v (\|\boldsymbol{y}_{i-1}\|_1 + l_i)}{2\gamma k} \\ &= \frac{\gamma v \|\boldsymbol{y}_i\|_1}{2\gamma k}. \end{aligned}$$

Thus the lemma follows.

Theorem 1. *Let f be a non-submodular function with DR ratio $\gamma \in [0, 1]$. For any given $\lambda \in [0, 1]$, denote $\boldsymbol{y}$ be the output of Algorithm 1. Then we have*

- *$f(\boldsymbol{y}) \geq \min\{\lambda\gamma/2^\gamma, 1 - 1/2^\gamma\} OPT$, where OPT is the optimal value.*
- *Algorithm 1 requires one pass, at most k space and $O(\log \|\boldsymbol{B}\|_\infty)$ update time per element.*

Proof. The proof splits into two cases depending on the cardinality of solution $\boldsymbol{y}$ returned by Algorithm 1.

1. At the end of the algorithm we have $\|\boldsymbol{y}\|_1 = k$, then by Lemma 3 we get

$$f(\boldsymbol{y}) \geq \frac{\gamma v \|\boldsymbol{y}\|_1}{2^\gamma k} = \frac{\gamma v}{2^\gamma} \geq \frac{\lambda\gamma}{2^\gamma} \cdot OPT.$$

2. We consider the case $\|\boldsymbol{y}\|_1 < k$. Suppose that $\boldsymbol{y}^*$ is the optimal solution of Algorithm 1, then it is easy to see that $\|\boldsymbol{y}^*\|_1 = k$. Let $\boldsymbol{x} = (\boldsymbol{y}^* - \boldsymbol{y}) \vee \mathbf{0}$. Denote $\boldsymbol{y}_i$ be the vector at the end of i-th iteration of Algorithm 1, $\boldsymbol{y}_i = l_1\mathcal{X}_1 + ... + l_i\mathcal{X}_i$. Then by Lemma 1 and Algorithm BinarySearchPivot, we have

$$f_{\boldsymbol{y}_i}(\mathcal{X}_i) < \frac{\gamma v}{2^\gamma k}. \tag{5}$$

Next, from 5, the monotonicity of f, Lemma 2 and the choice of v, we have that

$$\begin{aligned} OPT - f(\boldsymbol{y}) &\leq f(\boldsymbol{y}^* \vee \boldsymbol{y}) - f(\boldsymbol{y}) \\ &\leq f(\boldsymbol{y} + \boldsymbol{x}) - f(\boldsymbol{y}) \\ &\leq \frac{1}{\gamma} \sum_{e_i \in supp^+\{\boldsymbol{x}\}} x(e_i) f_{\boldsymbol{y}_i}(\mathcal{X}_i) \\ &\leq \frac{1}{\gamma} \sum_{e_i \in supp^+\{\boldsymbol{x}\}} x(e_i) \frac{\gamma v}{2^\gamma k} \\ &\leq \frac{v}{2^\gamma} \\ &\leq \frac{1}{2^\gamma} OPT. \end{aligned}$$

Rearranging the above inequality, we get

$$f(\boldsymbol{y}) \geq (1 - \frac{1}{2^\gamma}) OPT.$$

From the above two cases, we have

$$f(\boldsymbol{y}) \geq \min\{\frac{\lambda\gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma}\} \cdot OPT.$$

4 Streaming Algorithm with Known $f(\mathcal{X}_e)$

The premise of Algorithm 1 is that we know the value of the optimal solution OPT, which is unrealistic. To address this issue, we present the next algorithm, which is instantiated with different guesses of OPT. Suppose that we know the maximum values $\alpha = \max_{e \in E} f(\mathcal{X}_e)$. Combining with the DR ratio of f, it is easy to see the guesses $v \in \mathcal{V}_\epsilon$ of OPT are increasing from α to $k\alpha/\gamma$. For each guesses v, Algorithm 2 outputs a feasible solution $\boldsymbol{y}^v$. Finally, the best of these candidate vectors is returned.

Algorithm 2. Streaming-Know-MAX VAL

Require: function f, cardinality k, ground set E, $\mathbf{B}$, $\alpha = \max_{e \in E} f(\mathcal{X}_e)$.
1: $\mathcal{V}_\epsilon = \{(1+\epsilon)^i | i \in \mathbf{Z}, \alpha/(1+\epsilon) \leq (1+\epsilon)^i \leq k\alpha/\gamma\}$
2: for each $v \in \mathcal{V}_\epsilon$, set $\boldsymbol{y}^v \leftarrow \mathbf{0}$
3: **for** $i = 1, 2, ..., n$ **do**
4: **for** $v \in \mathcal{V}_\epsilon$ **do**
5: **if** $\|\boldsymbol{y}^v\|_1 \leq k$ **then**
6: $l_i \leftarrow$BinarySearchPivot$(f, \boldsymbol{y}^v, B(e_i), e_i, k, \frac{\gamma v}{2\gamma k})$
7: $\boldsymbol{y}^v \leftarrow \boldsymbol{y}^v + l_i \mathcal{X}_i$
8: **end if**
9: **end for**
10: **end for**
11: return $\arg\max_{v \in \mathcal{V}_\epsilon} f(\boldsymbol{y}^v)$

The next lemma is proven in Appendix C.

Lemma 4. *There is a guesses $v \in \mathcal{V}_\epsilon$ such that $(1-\epsilon) \cdot OPT \leq v \leq OPT$.*

Theorem 2. *Let f be a non-submodular function with DR ratio $\gamma \in [0,1]$. For any given $\epsilon \in [0,1]$, denote $\boldsymbol{y}$ be the output of Algorithm 2. Then we have*

- *$f(\boldsymbol{y}) \geq (1 - 1/2^\gamma - \epsilon)OPT$, where OPT is the optimal value.*
- *Algorithm 2 requires one pass, at most $O(k\epsilon^{-1}\log k/\gamma)$ space and $O(\epsilon^{-1}\log k/\gamma \log \|\boldsymbol{B}\|_\infty)$ update time per element.*

Proof. By the result of Lemma 4, we know that there must be a v_0 such that $(1-\epsilon)OPT \leq v_0 \leq OPT$. Let $\boldsymbol{y}_0$ denote the solution of Algorithm 2 output corresponding to v_0. For the rest of the proof, we suppose that the results of Lemma 1, Lemma 2 and Lemma 3 all hold for the Algorithm 2 with value v_0. Thus, in the same way as Theorem 1, we consider the following two cases

1. When $\|\boldsymbol{y}_0\|_1 = k$, by applying Lemma 3, we have

$$f(\boldsymbol{y}_0) \geq \frac{\gamma v_0 \|\boldsymbol{y}_0\|_1}{2\gamma k} = \frac{\gamma v_0}{2^\gamma} \geq \frac{(1-\epsilon)\gamma}{2^\gamma} \cdot OPT.$$

2. When $\|\boldsymbol{y}_0\|_1 < k$, we have

$$f(\mathbf{y}_0) \geq (1 - \frac{1}{2^\gamma}) \cdot OPT.$$

Obviously, we can get the following inequality

$$\frac{(1-\epsilon)\gamma}{2^\gamma} \geq 1 - \frac{1}{2^\gamma} - \epsilon, \forall \epsilon \in (0,1), \gamma \in [0,1].$$

From the above two cases, we have

$$\begin{aligned} f(\boldsymbol{y}_0) &\geq \min\{\frac{(1-\epsilon)\gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma}\} \cdot OPT \\ &\geq \min\{\frac{(1-\epsilon)\gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma} - \epsilon\} \cdot OPT \\ &= (1 - \frac{1}{2^\gamma} - \epsilon) \cdot OPT. \end{aligned}$$

Memory and Query Complexity. We first argue that the amount of $\mathcal{V}_\epsilon$ is $O(\epsilon^{-1}\log k/\gamma)$. For each arriving element e, we need to consider all the parameters in $\mathcal{V}_\epsilon$ to get $O(\epsilon^{-1}\log k/\gamma)$ different solutions. This implies that the memory of Algorithm 2 is $O(k\epsilon^{-1}\log k/\gamma)$. Further, suppose that the element e is fixed, then for each $v \in \mathcal{V}_\epsilon$, we have to call to Algorithm BinarySearchPivot to get a valid pivot. Combine with Lemma 1, we conclude that the query complexity is $O(\epsilon^{-1}\log\|\mathbf{B}\|_\infty \log k/\gamma)$.

5 The One Pass Streaming Algorithm

Algorithm 3. Streaming Algorithm

Require: function f, cardinality k, ground set E, $\mathbf{B}$ and $\epsilon \in (0,1)$.
1: $\mathcal{V}_\epsilon = \{(1+\epsilon)^i | i \in \mathbf{Z}^+\}$
2: for each $v \in \mathcal{V}_\epsilon$, set $\boldsymbol{y}^v \leftarrow \mathbf{0}$
3: $\alpha \leftarrow 0, \beta \leftarrow 0$
4: **for** $i = 1, 2, ..., n$ **do**
5: $\quad\alpha \leftarrow \max\{\alpha, f(\mathcal{X}_i)\}, \beta \leftarrow \{\beta, f(B(e_i)\mathcal{X}_i)\}$
6: $\quad\mathcal{V}_\epsilon^i = \{(1+\epsilon)^s | s \in \mathbf{Z}, \alpha/(1+\epsilon) \le (1+\epsilon)^s \le 2^\gamma k\beta/\gamma\}$
7: $\quad$Delete all $\boldsymbol{y}^v$, where $v \notin \mathcal{V}_\epsilon^i$
8: $\quad$**for** $v \in \mathcal{V}_\epsilon^i$ **do**
9: $\quad\quad$**if** $\|\boldsymbol{y}_v\|_1 < k$ **then**
10: $\quad\quad\quad l_i \leftarrow$BinarySearchPivot$(f, \boldsymbol{y}^v, B(e_i), e_i, k, \frac{\gamma v}{2^\gamma k})$
11: $\quad\quad\quad\boldsymbol{y}^v \leftarrow \boldsymbol{y}^v + l_i\mathcal{X}_i$
12: $\quad\quad$**end if**
13: $\quad$**end for**
14: **end for**
15: return $\arg\max_{v\in\mathcal{V}_\epsilon^n} f(\boldsymbol{y}^v)$

Both Algorithm 1 and Algorithm 3 are idealized versions. This is due to the fact that we do not know the exact value of OPT and α defined in Algorithm 2. To solve this problem, we present a new one-pass algorithm in this section, in which we estimate the values of α and a new parameter β. For each element e, denote α and β be the current maximum $f(\mathcal{X}_e)$ and $f(B(e)\mathcal{X}_e)$. This implies that when element e_i and its copies arrive, their corresponding set $\mathcal{V}_\epsilon^i$ will be updated, which is recorded as $\mathcal{V}_\epsilon^i = \{(1+\epsilon)^s | s \in \mathbf{Z}, \alpha/(1+\epsilon) \le (1+\epsilon)^s \le 2^\gamma k\beta/\gamma\}$. When the parameter v first appears in $\mathcal{V}_\epsilon^i$, the value of the vector $\boldsymbol{y}^v$ is $\mathbf{0}$. When the parameter v is not in $\mathcal{V}_\epsilon^i$, we delete the vector $\boldsymbol{y}^v$ to save memory. Finally, the best of $\boldsymbol{y}^v$ is returned.

Lemma 5. *For any $v \in \cup_{i=1}^n \mathcal{V}_\epsilon^i$, denote $\boldsymbol{y}^v$ be the final solution of Algorithm 3 and $\boldsymbol{x}^v = (\boldsymbol{y}^* - \boldsymbol{y}^v) \vee \mathbf{0}$. Suppose that v first appears in $\mathcal{V}_\epsilon^m$. Then for each $e_i, i \in [m-1]$, it always satisfies*

$$f(\boldsymbol{x}^v(e_i)\mathcal{X}_i) < \boldsymbol{x}^v(e_i) \cdot \frac{\gamma v}{2^\gamma k}. \tag{6}$$

Proof. The proof is by contradiction. Suppose that there exists a $i_0 \in [m-1]$ such that

$$f(\boldsymbol{x}^v(e_{i_0})\mathcal{X}_{i_0}) \geq \boldsymbol{x}^v(e_{i_0})\frac{\gamma v}{2^\gamma k}. \tag{7}$$

Let $\alpha_i = \max_{j\in[i]} f(\mathcal{X}_j)$ and $\beta_i = \max_{j\in[i]} f(B(e_j)\mathcal{X}_j)$. Then combine the monotonicity of f and (6), we have

$$v \leq \frac{f(\boldsymbol{x}^v(i_0)\mathcal{X}_{i_0})2^\gamma k}{\boldsymbol{x}^v(e_{i_0})\gamma} \leq \frac{2^\gamma k}{\gamma} f(B(e_{i_0})\mathcal{X}_{i_0}) \leq \frac{2^\gamma k \beta_{i_0}}{\gamma}.$$

Since $v \in \mathcal{V}_\epsilon^m$, we get

$$v \geq \frac{\alpha_m}{1+\epsilon} \geq \frac{\alpha_{i_0}}{1+\epsilon}.$$

Using the above two inequalities, we have $\frac{\alpha_{i_0}}{1+\epsilon} \leq v \leq \frac{2^\gamma k \beta_{i_0}}{\gamma}$. That implies that $v \in \mathcal{V}_{i_0}$, which contradicts the fact v first appears in $\mathcal{V}_\epsilon^m$. Thus we get the desired result.

Theorem 3. *Let f be a non-submodular consider with DR ratio $\gamma \in [0,1]$. For any given $\epsilon \in [0,1]$, denote $\boldsymbol{y}$ be the output of Algorithm 3. Then we have*

- *$f(\boldsymbol{y}) \geq (1 - 1/2^\gamma - \epsilon)OPT$, where OPT is the optimal value.*
- *Algorithm 3 requires one pass, at most $O(k\epsilon^{-1}\log k/\gamma)$ space and $O(\epsilon^{-1}\log k/\gamma \log \|\boldsymbol{B}\|_\infty)$ update time per element.*

Proof. By the result of Lemma 4, we know that there must be a v_0 such that $(1-\epsilon)OPT \leq v_0 \leq OPT$. From Lemma 5, we can assume that v_0 is considered when the first element arriving. For the rest of the proof, we consider the quality of $\boldsymbol{y}^{v_0}$. The analysis is similar as in Theorem 2, thus we have

$$f(\boldsymbol{y}^{v_0}) \geq (1 - 1/2^\gamma - \epsilon)OPT.$$

Since $\boldsymbol{y}$ is the maximum of all solutions, we have

$$f(\boldsymbol{y}) \geq f(\boldsymbol{y}^{v_0}) \geq (1 - 1/2^\gamma - \epsilon)OPT.$$

Thus, we complete the proof.

References

1. Goemans, M.-X., Williamson, D.-P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
2. Lin, H., Bilmes, J.: A class of submodular functions for document summarization. In: 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon, pp. 510–520. Association for Computational Linguistics (2011)
3. Sipos, R., Swaminathan, A., Shivaswamy, P., Joachims, T.: Temporal corpus summarization using submodular word coverage. In: 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, pp. 754–763. Association for Computing Machinery (2012)

4. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a submodular set function subject to a matroid constraint (extended abstract). In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 182–196. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72792-7_15
5. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 199–208. Association for Computing Machinery (2009)
6. Seeman, L., Singer, Y.: Adaptive seeding in social networks. In: 54th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, pp. 459–468. Institute of Electrical and Electronic Engineers (2013)
7. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 1029–1038. Association for Computing Machinery (2010)
8. Ageev, A.-A., Sviridenko, M.-I.: An 0.828-approximation algorithm for the uncapacitated facility location problem. Discret. Appl. Math. **93**(2–3), 149–156 (1999)
9. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: Streaming submodular maximization: massive data summarization on the fly. In: 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 671–680. Association for Computing Machinery (2014)
10. Buchbinder, N., Feldman, M., Schwartz, R.: Online submodular maximization with preemption. In: 26th ACM-SIAM Symposium on Discrete Algorithms, Cambridge, Massachusetts, USA, pp. 1202–1216. Society for Industrial and Applied Mathematics (2014)
11. Norouzi-Fard, A., Tarnawski, J., Mitrovic, S., Zandieh, A., Mousavifar, A., Svensson, O.: Beyond 1/2-approximation for submodular maximization on massive data streams. In: 35th International Conference on Machine Learning, Stockholm, Sweden, pp. 3829–3838. International Machine Learning Society (2018)
12. Kazemi, E., Mitrovic, M., Zadimoghaddam, M., Lattanzi, S., Karbasi, A.: Submodular streaming in all its glory: tight approximation, minimum memory and low adaptive complexity. In: 36th International Conference on Machine Learning, Long Beach, California, pp. 3311–3320. International Machine Learning Society (2019)
13. Soma, T., Yoshida, Y.: Maximizing monotone submodular functions over the integer lattice. Math. Program. 539–563 (2018). https://doi.org/10.1007/s10107-018-1324-y
14. Soma, T., Kakimura, N., Inaba, K., Kawarabayashi, K.-I.: Optimal budget allocation: theoretical guarantee and efficient algorithm. In: 31th International Conference on Machine Learning, Beijing, China, pp. 351–359. International Machine Learning Society (2014)
15. Nong, Q., Fang, J., Gong, S., Du, D., Feng, Y., Qu, X.: A 1/2-approximation algorithm for maximizing a non-monotone weak-submodular function on a bounded integer lattice. J. Comb. Optim. **39**(4), 1208–1220 (2020). https://doi.org/10.1007/s10878-020-00558-4
16. Kuhnle, A., Smith, J.-D., Crawford, V., Thai, M.: Fast maximization of non-submodular, monotonic functions on the integer lattice. In: 35th International Conference on Machine Learning, Stockholm, Sweden, pp. 2786–2795. International Machine Learning Society (2018)

17. Wang, Y., Xu, D., Wang, Y., Zhang, D.: Non-submodular maximization on massive data streams. J. Glob. Optim. **76**(4), 729–743 (2019). https://doi.org/10.1007/s10898-019-00840-8
18. Tan, J., Zhang, D., Zhang, H., Zhang, Z.: Streaming algorithms for monotone DR-submodular maximization under a knapsack constraint on the integer lattice. In: Ning, L., Chau, V., Lau, F. (eds.) PAAP 2020. CCIS, vol. 1362, pp. 58–67. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-0010-4_6
19. Zhang, Z., Guo, L., Wang, Y., Xu, D., Zhang, D.: Streaming algorithms for maximizing monotone DR-submodular functions with a cardinality constraint on the integer lattice. Asia-Pac. J. Oper. Res. 2140004 (2021)
20. Zhang, Z., Guo, L., Wang, L., Zou, J.: A streaming model for monotone lattice submodular maximization with a cardinality constraint. In: Zhang, Y., Xu, Y., Tian, H. (eds.) PDCAT 2020. LNCS, vol. 12606, pp. 362–370. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-69244-5_32
21. Das, A., Kempe, D.: Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In: 28th International Conference on Machine Learning, Bellevue, WA, USA, pp. 1057–1064. International Machine Learning Society (2011)
22. Nong, Q., Sun, T., Gong, S., Fang, Q., Du, D., Shao, X.: Maximize a monotone function with a generic submodularity ratio. In: Du, D.-Z., Li, L., Sun, X., Zhang, J. (eds.) AAIM 2019. LNCS, vol. 11640, pp. 249–260. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27195-4_23