TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/14 pp888–895
DOI: 10.26599/TST.2022.9010031
Volume 28, Number 5, October 2023

Streaming Algorithms for Non-Submodular Maximization on the Integer Lattice

Jingjing Tan, Yue Sun, Yicheng Xu, and Juan Zou*

Abstract: Many practical problems emphasize the importance of not only knowing whether an element is selected but also deciding to what extent it is selected, which imposes a challenge on submodule optimization. In this study, we consider the monotone, nondecreasing, and non-submodular maximization on the integer lattice with a cardinality constraint. We first design a two-pass streaming algorithm by refining the estimation interval of the optimal value. For each element, the algorithm not only decides whether to save the element but also gives the number of reservations. Then, we introduce the binary search as a subroutine to reduce the time complexity. Next, we obtain a one-pass streaming algorithm by dynamically updating the estimation interval of optimal value. Finally, we improve the memory complexity of this algorithm.

Key words: integer lattice; non-submodular; streaming algorithm; cardinality constraint

1 Introduction

Submodular functions defined on a set are those that take a subset of a set as input and return a real value as output. For a set submodular function, its submodular property is equivalent to the diminishing returns property; that is, if the same item is added to a large set and a small set, the marginal gain of the latter is greater than that of the former. Many problems in computer science, artificial intelligence, deep learning, and other areas can be characterized by monotone set submodular functions, such as data summarization[1], influence maximization[2], maximum entropy sampling[3], active learning[4], and online advertising[5]. With the extensive application in practical applications, the submodular optimization problem has attracted the attention of scholars making it one of the hot research issues in combinatorial optimization and computer science[6–9]. The extensive applications of submodular function in practical problems have also promoted the development of submodular theory[10, 11]. Numerous studies on submodular functions have paid close attention to the case of set functions[12–16]. It is known that we can use a simple greedy algorithm to approximately maximize a submodular set function in polynomial time with approximation guarantees. The extensive applications of submodular function in practical problems have also promoted the development of submodular theory. Numerous studies on submodular functions have paid close attention to the case of set functions. It is known that we can use a simple greedy algorithm to approximately maximize a submodular set function in polynomial time with approximation guarantees. Nemhauser et al.[17] first proposed a greedy algorithm for the problem of maximizing a submodular function with the cardinality constraint. The approximation is $(1 - 1/e)$. For this problem, Feige[18]

• Jingjing Tan is with School of Mathematics and Information Science, Weifang University, Weifang 261061, China. E-mail: tanjingjing1108@163.com.
• Yue Sun is with Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing 100124, China. E-mail: suny39@emails.bjut.edu.cn.
• Yicheng Xu is with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: yc.xu@siat.ac.cn.
• Juan Zou is with School of Mathematical Sciences, Qufu Normal University, Qufu 273165, China. E-mail: zoujuanjn@163.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-04-04; revised: 2022-06-18; accepted: 2022-08-08

reported that the tightest approximation is $(1 - 1/\mathrm{e})$ under the condition $P = NP$. The knapsack constraint introduced to this problem with the first $(1 - 1/\mathrm{e})$-approximation algorithm is proposed by Sviridenko[19]. Then, Calinescu et al.[20] obtained the same approximation ratio of $(1 - 1/\mathrm{e})$ for the same problem with a matroid constraint by using the continuous greedy approach.

The non-submodular optimization problem has also attracted a great deal of research interest due to its application to practical problems, such as viral marketing and machine learning[21–29]. Bian et al.[30] designed a standard greedy algorithm for solving the problem of non-submodular maximization with cardinality constraint by characterizing the non-submodular functions with the help of parameter weak submodular rate, thus proving that this algorithm has a tight approximation ratio. Bogunovic et al.[31] introduced the DR ratio γ_f to study the robust maximization of a set function. Kuhnle et al.[32] extended the DR ratio and weak DR ratio to lattice functions and proved that $\gamma_f \leqslant \gamma_f^w$. They considerd the non-submodular maximization on the integer lattice with cardinality constraint, and proposed two threshold greedy algorithms that generalize a prior work on this topic. More excellent surveys on the non-submodule optimization research can be seen in Refs. [33–38]. Moreover the rapid development of computer science technology and Internet has ushered in the era of big data and led to the processing of massive data. Due to the limited computer storage capacity and the high speed of data arrival, it is very important to deal with the data in a streaming manner. A well designed streaming algorithm can solve the problem of massive data effectively.

1.1 Problem

We begin by defining the problem definition of the monotone nondecreasing Non-submodular Maximization with a Cardinality Constraint (NMCC). Let $[k]$ be the set of all the positive integers from 1 to k for any $k \in \mathbf{N}^+$. We suppose that the items in set $G = \{e_1, e_2, \ldots, e_n\}$ arrive one by one. In the streaming algorithm, we must make a decision for the coming item immediately before the next item arrives. At the same time, we should consider not only the running time to measure the performance of the algorithms, but also the complexity of storage and the number of passes for reading the data. Let $\boldsymbol{s} \in \mathbf{N}^G$ with the component of coordinate $e_i \in G$ be $\boldsymbol{s}(e_i)$. Let χ_{e_i} denote the standard unit vector; that is, all the components have a value of 0, except for the i-th component, which has a value of 1. Denote $\boldsymbol{s}(S)$ as the sum of $\boldsymbol{s}(e_i)$, where $e_i \in S$. For any $\boldsymbol{s} \in \mathbf{N}^G$, $supp^+(\boldsymbol{s}) = \{e \in G | \boldsymbol{s}(e) > 0\}$ is the supporting set of $\boldsymbol{s}$. Let $\{\boldsymbol{s}\}$ be the multi-set in which the number of occurrences of e is $\boldsymbol{s}(e)$ and $|\{\boldsymbol{s}\}| := \boldsymbol{s}(G)$; $\boldsymbol{c}$ is a box in $\{N \cup \{\infty\}\}^G$; $D_{\boldsymbol{c}} = \{\boldsymbol{s} \in \mathbf{N}^G : \boldsymbol{s} \leqslant \boldsymbol{c}\}$. $f()$ is a non-submodular function defined on $D_{\boldsymbol{c}}$; and $f(\mathbf{0}) = 0$. The problem can be described as follows:

$$\max_{\boldsymbol{s} \leqslant \boldsymbol{c}, \boldsymbol{s}(G) \leqslant k} f(\boldsymbol{s}) \tag{1}$$

where $\boldsymbol{s}(G) \leqslant k$ is the cardinality constraint and

$$\boldsymbol{s}(G) = \sum_{e \in G} \boldsymbol{s}(e).$$

1.2 Preliminary

Here, we introduce some definitions and basic facts on the submodular functions in this subsection.

For each element $e \in G$, we denote $(\boldsymbol{s} \wedge \boldsymbol{t})(e)$ as the minimum value of $\boldsymbol{s}(e)$ and $\boldsymbol{t}(e)$, and $(\boldsymbol{s} \vee \boldsymbol{t})(e)$ is the maximum value of $\boldsymbol{s}(e)$ and $\boldsymbol{t}(e)$. For any $\{\boldsymbol{s}\}$ and $\{\boldsymbol{t}\}$, we denote $\{\boldsymbol{s}\} \setminus \{\boldsymbol{t}\}$ as the coordinate wise maximum of $(\boldsymbol{s}(e) — \boldsymbol{t}(e))$ and $\mathbf{0}$. The monotone nondecreasing of $f : \mathbf{N}^G \to \mathbf{R}_+$ and $f(\boldsymbol{s}) \leqslant f(\boldsymbol{t})$ holds for any $\boldsymbol{s} \leqslant \boldsymbol{t}$. A nonnegative and normalized function f means that $f(\boldsymbol{s}) \geqslant 0$ for any $\boldsymbol{s} \in \mathbf{N}^G$ and $f(\mathbf{0}) = 0$. The following two definitions illustrate the mathematical descriptions of the DR-submodular and lattice submodular.

Definition 1 f is called DR-submodular if it holds for any $\boldsymbol{s}, \boldsymbol{t} \in \mathbf{N}^G$ with $\boldsymbol{s} \leqslant \boldsymbol{t}$ and $e \in G$, such that

$$f(\boldsymbol{t} + \chi_e) - f(\boldsymbol{t}) \leqslant f(\boldsymbol{s} + \chi_e) + f(\boldsymbol{s}).$$

Definition 2 f is called a lattice submodular if it holds for all $\boldsymbol{s}, \boldsymbol{t} \in \mathbf{N}^G$ that

$$f(\boldsymbol{s} \vee \boldsymbol{t}) + f(\boldsymbol{s} \wedge \boldsymbol{t}) \leqslant f(\boldsymbol{s}) + f(\boldsymbol{t}).$$

Let $F_{\boldsymbol{c}}$ be the set of nonnegative monotone DR-submodular functions. For $f \in F_{\boldsymbol{c}}$, and vectors $\boldsymbol{s}, \boldsymbol{t} \in \mathbf{N}^G$, let $f(\boldsymbol{t}|\boldsymbol{s})$ be the marginal increment of a vector $\boldsymbol{s}$ with $\boldsymbol{t}$, that is,

$$f(\boldsymbol{t}|\boldsymbol{s}) = f(\boldsymbol{s} + \boldsymbol{t}) - f(\boldsymbol{s}).$$

Definition 3[32] The DR ratio of a function f in $F_{\boldsymbol{c}}$ is the maximum scalar $\gamma_f(f)$, such that for any $\boldsymbol{s}, \boldsymbol{t} \in D_{\boldsymbol{c}}$ with $\boldsymbol{s} \leqslant \boldsymbol{t}$ and $e \in G$,

$$\gamma_f(f) f(\chi_e|\boldsymbol{t}) \leqslant f(\chi_e|\boldsymbol{s}),$$

where $\boldsymbol{t} + \chi_e \in D_{\boldsymbol{c}}$.

Definition 4[32] The weak DR ratio of a function f in $F_{\boldsymbol{c}}$ is the maximum scalar $\gamma_f^w(f)$, such that for any $\boldsymbol{s}, \boldsymbol{t} \in D_{\boldsymbol{c}}$ with $\boldsymbol{s} \leqslant \boldsymbol{t}$,

$$\gamma_f^w(f)(f(\boldsymbol{t})-f(\boldsymbol{s}))\leqslant\sum_{e\in\{t\}\setminus\{s\}}f(\chi_e|\boldsymbol{s}).$$

Remark 1 From the definition of DR ratio $\gamma_f(f)$ and weak DR ratio $\gamma_f^w(f)$, we can easily have $\gamma_f(f)\in[0,1]$, $\gamma_f^w(f)\in[0,1]$, and $\gamma_f(f)\leqslant\gamma_f^w(f)$ for any $f\in F_{\boldsymbol{c}}$.

Remark 2 If $\gamma_f(f)$ and $\gamma_f^w(f)$ are equal to 1, the function f is a DR-submodular function on the lattice, which means that DR ratio and weak DR ratio generalize the concept of DR-submodular. This non-submodular function captures many practical problems, such as the optimal budget allocation problem in the advertisement.

In this paper, we denote by $F_{\boldsymbol{c}}^{\gamma_f,\gamma_f^w}$ for the set of all the functions $f\in F_{\boldsymbol{c}}$, in which the DR ratio of f is γ_f and the weak DR ratio of f is γ_f^w. We denote the optimal solution vector and optimal value as $\boldsymbol{s}^*$ and OPT, respectively. We consider the streaming algorithms for NMCC on the integer lattice.

The remainder of this paper is organized as follows. In Section 2, we first design three streaming algorithms for NMCC, after which we analyze the performance of the three algorithms. In Section 3, we summarize our work.

2 Main Result

In this section, we design a streaming algorithm with two-pass and two online streaming algorithms with one-pass for the NMCC. We extend the problem studied in Ref. [37] to the integer lattice.

2.1 Two-pass streaming algorithm

In the streaming model, we cannot decide immediately whether the value of the arriving element exceeds the maximum marginal value over each iteration. A natural idea is to compare the marginal gain produced by the arriving element with OPT in a certain way. To determine whether or not the arriving item e is selected, a specified threshold of $\frac{\gamma_f v/2^{\gamma_f}-f(\boldsymbol{s})}{k-\boldsymbol{s}(G)}$ is used. We combine the exponential growth method for estimating the OPT with binary search to give the $\min\{(1-\varepsilon)\gamma_f/2^{\gamma_f},(1-1/\gamma_f^w 2^{\gamma_f})\}$-approximation algorithm for NMCC. The detailed description refers to Algorithms 1 and 2 below.

Algorithm 1 Binary search

Input: $f():\mathbf{N}^G\to\mathbf{R}^+$, stream of data G, $e\in G$, $\boldsymbol{s},\boldsymbol{c}\in\mathbf{N}^G$, $k\in\mathbf{N}$, and $\tau\in\mathbf{R}^+$.
Output: $\alpha\in\mathbf{R}^+$.
$\alpha_t\leftarrow\min\{\boldsymbol{c}(e)-\boldsymbol{s}(e),k-\boldsymbol{s}(G)\}$;
$\alpha_s\leftarrow 1$;
if $\frac{f(\alpha_t\chi_e|\boldsymbol{s})}{\alpha_t}\geqslant\tau$ **then**
  return α_t
end if
if $f(\chi_e|\boldsymbol{s})<\tau$ **then**
  return 0
end if
while $\alpha_t>\alpha_s+1$ **do**
  $\rho=\lfloor\frac{\alpha_t+\alpha_s}{2}\rfloor$;
  if $f(\rho\chi_e|\boldsymbol{s})\geqslant\tau$ **then**
    $\alpha_s=\rho$,
  else
    $\alpha_t=\rho$;
  end if
end while
return α_t.

Algorithm 2 Streaming algorithm

Input: $f\in F_{\boldsymbol{c}}$, G, cardinality constraints k, $\varepsilon\in(0,1)$.
Output: Vector $\boldsymbol{s}\in\mathbf{N}^G$.
$m\leftarrow\max_{e\in G}f(\chi_e)$;
$V_\varepsilon=\{(1+\varepsilon)^q|q\in N,\frac{m}{1+\varepsilon}\leqslant(1+\varepsilon)^q\leqslant\frac{km}{\gamma_f}\}$;
for $v\in V_\varepsilon$, **do**
  set $\boldsymbol{s}^v\leftarrow\boldsymbol{0}$;
  for $i=1,2,\dots,n$ **do**
    if $\boldsymbol{s}^v(G)<k$, **then**
      $\alpha\leftarrow\text{BinarySearch}\left(f,\boldsymbol{s}^v,\boldsymbol{c},e_i,k,\frac{\gamma_f v/2^{\gamma_f}-f(\boldsymbol{s}^v)}{k-\boldsymbol{s}^v(G)}\right)$;
      $\boldsymbol{s}^v\leftarrow\boldsymbol{s}^v+\alpha\chi_e$;
      return $\boldsymbol{s}^v$
    end if
  end for
end for
return $\boldsymbol{s}=\arg\max_{v\in V_\varepsilon^n}f(\boldsymbol{s}^v)$.

Lemma 1 Suppose $\boldsymbol{s}_i$ is the output of the i-th iteration in Algorithm 2, then we have

$$f(\boldsymbol{s}_i)\geqslant\frac{\gamma_f v\boldsymbol{s}_i(G)}{2^{\gamma_f}k}\tag{2}$$

Proof From Algorithm 2, we know the initial vector $\boldsymbol{s}_0=\boldsymbol{0}$, so Formula (2) holds naturally. Then, we assume that Formula (2) holds for the i-th iteration; we only need to prove that it holds for the $(i+1)$-th iteration. If we let α_{i+1} be the output returned from Algorithm 1, then we have

$$f(\boldsymbol{s}_i+\alpha_{i+1}\chi_{e_{i+1}})=f(\alpha_{i+1}\chi_{e_{i+1}}|\boldsymbol{s}_i)+f(\boldsymbol{s}_i)\geqslant\frac{\alpha_{i+1}(\frac{\gamma_f v}{2^{\gamma_f}}-f(\boldsymbol{s}_i))}{k-\boldsymbol{s}_i(G)}+f(\boldsymbol{s}_i)\geqslant\frac{\alpha_{i+1}\times\gamma_f v}{2^{\gamma_f}(k-\boldsymbol{s}_i(G))}+\frac{k-\boldsymbol{s}_i(G)-\alpha_{i+1}}{k-\boldsymbol{s}_i(G)}f(\boldsymbol{s}_i).$$

Together with the introduced hypothesis, we conclude that

$$f(\boldsymbol{s}_i+\alpha_{i+1}\chi_{e_{i+1}}) \geqslant \frac{\alpha_{i+1}\gamma_f v}{2^{\gamma_f}(k-\boldsymbol{s}_i(G))}+\frac{k-\boldsymbol{s}_i(G)-\alpha_{i+1}}{k-\boldsymbol{s}_i(G)}\times\frac{\gamma_f v \boldsymbol{s}_i(G)}{2^{\gamma_f}k} = \frac{\gamma_f v(k-\boldsymbol{s}_i(G))(\boldsymbol{s}_i(G)+\alpha_{i+1})}{2^{\gamma_f}k(k-\boldsymbol{s}_i(G))} = \frac{\gamma_f v}{2^{\gamma_f}k}(\boldsymbol{s}_i(G)+\alpha_{i+1}) = \frac{\gamma_f v}{2^{\gamma_f}k}\boldsymbol{s}_{i+1}(G),$$

which completes the proof. ■

In the following Lemma 2, we consider the marginal increment of χ_e with $e \in \{\boldsymbol{s}^*\} \setminus \{\tilde{\boldsymbol{s}}\}$.

Lemma 2 Suppose $\tilde{\boldsymbol{s}}$ is the final output of Algorithm 2 and $\tilde{\boldsymbol{s}}(G) < k$, then we have

$$f(\chi_e|\tilde{\boldsymbol{s}}) < \frac{v}{2^{\gamma_f}k},$$

where $e \in \{\boldsymbol{s}^*\} \setminus \{\tilde{\boldsymbol{s}}\}$.

Proof Assume that Algorithm 2 produces a vector $\boldsymbol{s}'_e$ right before the element e's arrival. α_e is the output of the BinarySearch subroutine satisfying the following two inequations:

$$\frac{f(\alpha_e\chi_e|\boldsymbol{s}'_e)}{\alpha_e} \geqslant \frac{\gamma_f v/2^{\gamma_f}-f(\boldsymbol{s}'_e)}{k-\boldsymbol{s}_e(G)} \tag{3}$$

and

$$\frac{f((\alpha_e+1)\chi_e|\boldsymbol{s}'_e)}{\alpha_e+1} < \frac{\gamma_f v/2^{\gamma_f}-f(\boldsymbol{s}'_e)}{k-\boldsymbol{s}_e(G)} \tag{4}$$

Let $\boldsymbol{s}_e=\boldsymbol{s}'_e+\alpha_e\chi_e \leqslant \tilde{\boldsymbol{s}}$. Therefore,

$$\begin{aligned} f(\chi_e|\tilde{\boldsymbol{s}}) \leqslant{} & \frac{1}{\gamma_f}f(\chi_e|\boldsymbol{s}_e) = \\ & \frac{1}{\gamma_f}f(\chi_e|\boldsymbol{s}'_e+\alpha_e\chi_e) = \\ & \frac{1}{\gamma_f}[f(\boldsymbol{s}'_e+(\alpha_e+1)\chi_e)-f(\boldsymbol{s}'_e+\alpha_e\chi_e)] = \\ & \frac{1}{\gamma_f}[f(\boldsymbol{s}'_e+(\alpha_e+1)\chi_e)-f(\boldsymbol{s}'_e)+f(\boldsymbol{s}'_e)- \\ & f(\boldsymbol{s}'_e+\alpha_e\chi_e)] = \\ & \frac{1}{\gamma_f}[f((\alpha_e+1)\chi_e|\boldsymbol{s}'_e)-f(\alpha_e\chi_e|\boldsymbol{s}'_e)] < \\ & \frac{1}{\gamma_f}\left[(\alpha_e+1)\frac{\gamma_f v/2^{\gamma_f}-f(\boldsymbol{s}'_e)}{k-\boldsymbol{s}_e(G)} - \alpha_e\frac{\gamma_f v-2^{\gamma_f}f(\boldsymbol{s}'_e)}{2^{\gamma_f}(k-\boldsymbol{s}_e(G))}\right] < \\ & \frac{1}{\gamma_f}\frac{\gamma_f v-2^{\gamma_f}f(\boldsymbol{s}'_e)}{2^{\gamma_f}(k-\boldsymbol{s}_e(G))} \leqslant \\ & \frac{1}{\gamma_f}\frac{\gamma_f v-\gamma_f v\cdot\boldsymbol{s}_e(G)}{2^{\gamma_f}(k-\boldsymbol{s}_e(G))} = \frac{v}{2^{\gamma_f}k} \end{aligned} \tag{5}$$

This completes the proof. ■

Based on the Lemmas 1 and 2, we can analyze the performance of Algorithm 2. In the analysis, Lemma 1 is used for the case $\boldsymbol{s}(G)=k$, and Lemma 2 is used for the case $\boldsymbol{s}(G)<k$.

Theorem 1 For any $\varepsilon \in (0,1)$ and given function $f \in F_c$, Algorithm 2 is a two-pass algorithm, the approximation is $\min\{(1-\varepsilon)\gamma_f/2^{\gamma_f},(1-1/\gamma_f^w 2^{\gamma_f})\}$, the memory complexity is $O(\frac{k}{\varepsilon}\log\frac{k}{\varepsilon})$, and the query times per element is $O(\frac{\log k}{\varepsilon}\log\frac{\log k}{\varepsilon})$.

Proof Suppose $\boldsymbol{s}^*=\sum_{e\in\{\boldsymbol{s}^*\}}\chi_e$. From the definition of the DR ratio, we obtain

$$OPT=f(\boldsymbol{s}^*)=f\left(\sum_{e\in\{\boldsymbol{s}^*\}}\chi_e\right) \leqslant \frac{1}{\gamma_f}\sum_{e\in\{\boldsymbol{s}^*\}}f(\chi_e) \leqslant \frac{k\cdot m}{\gamma_f}.$$

Moreover,

$$m=\max_{e\in G}f(\chi_e) \leqslant f(\boldsymbol{s}^*) \tag{6}$$

Combined Formulas (5) and (6), we have

$$m \leqslant OPT \leqslant \frac{k\cdot m}{\gamma_f}.$$

Let $\bar{i}=\lfloor\log_{1+\varepsilon}OPT\rfloor$, thus there exists $v=(1+\varepsilon)^{\bar{i}}$, such that $v \leqslant OPT$ and

$$v \geqslant \frac{OPT}{1+\varepsilon} \geqslant \max\left\{(1-\varepsilon)OPT, \frac{m}{1+\varepsilon}\right\} \tag{7}$$

From Formulas (6) and (7), we obtain

$$v \in [(1-\varepsilon)OPT, OPT].$$

Next, for $v \in V_\varepsilon$, we denote $\boldsymbol{s}^v$ as the output of Algorithm 2, and consider the following two cases:

(1) $\boldsymbol{s}^v(G)=k$

According to Lemma 1, we have

$$f(\boldsymbol{s}^v) \geqslant \frac{\gamma_f v \boldsymbol{s}^v(G)}{2^{\gamma_f}k} = \frac{(1-\varepsilon)\gamma_f}{2^{\gamma_f}}OPT \tag{8}$$

(2) $\boldsymbol{s}^v(G)<k$

From Lemma 2 and the weak DR ratio, we have

$$\begin{aligned} f(\boldsymbol{s}^*\cup\boldsymbol{s}^v)-f(\boldsymbol{s}^v) \leqslant{} & \frac{1}{\gamma_f^w}\sum_{e\in suup^+\{\boldsymbol{s}^*-\boldsymbol{s}^v\}} f((\boldsymbol{s}^*-\boldsymbol{s}^v)(e)\chi_e|\boldsymbol{s}^v) \leqslant \\ & \frac{1}{\gamma_f^w}\sum_{e\in\{\boldsymbol{s}^*\}\setminus\{\boldsymbol{s}^v\}} f(\chi_e|\boldsymbol{s}^v) < \\ & \frac{1}{\gamma_f^w}\sum_{e\in\{\boldsymbol{s}^*\}\setminus\{\boldsymbol{s}^v\}}\frac{v}{2^{\gamma_f}k} \leqslant \\ & \frac{v}{\gamma_f^w 2^{\gamma_f}} \leqslant \frac{OPT}{\gamma_f^w 2^{\gamma_f}}. \end{aligned}$$

As $f(\boldsymbol{s}^* \cup \boldsymbol{s}^v) - f(\boldsymbol{s}^v) \geqslant OPT - f(\boldsymbol{s}^v)$, we obtain

$$f(\boldsymbol{s}^v) \geqslant (1 - \frac{1}{\gamma_f^w 2^{\gamma_f}})OPT \tag{9}$$

Combining Formulas (8) and (9), we obtain

$$f(\boldsymbol{s}^v) \geqslant \min\left\{\frac{(1-\varepsilon)\gamma_f}{2^{\gamma_f}}, 1 - \frac{1}{\gamma_f^w 2^{\gamma_f}}\right\} OPT.$$

From Algorithm 2, we know that $\tilde{s} = \arg\max_{v \in V_\varepsilon} f(\boldsymbol{s}^v)$. Thus, we can conclude that

$$f(\tilde{s}) \geqslant f(\boldsymbol{s}^v) \geqslant \min\left\{\frac{(1-\varepsilon)\gamma_f}{2^{\gamma_f}}, 1 - \frac{1}{\gamma_f^w 2^{\gamma_f}}\right\} OPT$$

The proof is completed. ■

2.2 One-pass streaming algorithm

For the streaming algorithm, the number of rounds to read data is an important index to measure the performance of such an algorithm. Naturally, we want to obtain a one-pass streaming algorithm by dynamically updating the maximum value of the standard unit vector according to the arriving elements and replacing the estimation value range of the OPT at the same time. We design the one-pass streaming algorithm for the NMCC, which is stated in Algorithm 3. Here, γ_f is unknown at the beginning. However, as $f(\chi_e) > 0$, we also know $\gamma_f > 0$. Thus, there is an expression $\varepsilon \in (0,1)$ that satisfies $\gamma_f > \varepsilon$.

Theorem 2 For any given $\varepsilon \in (0,1)$ and function $f \in F_{\boldsymbol{c}}$, Algorithm 3 is a one-pass algorithm, the approximation is $\min\{(1-\varepsilon)\gamma_f/2^{\gamma_f}, (1 - 1/\gamma_f^w 2^{\gamma_f})\}$, the memory complexity is $O(\frac{k}{\varepsilon}\log\frac{k}{\varepsilon^2})$, and the query times per element is $O(\frac{\log k}{\varepsilon}\log\frac{k}{\varepsilon^2})$.

Algorithm 3 Streaming algorithm with one pass

Input: $f \in F_c$, stream of data G, cardinality constraints k, $\varepsilon \in (0,1)$.

Output: Vector $\boldsymbol{s} \in \mathbf{N}^G$.

```
V_ε = {(1+ε)^q | q ∈ N};
for v ∈ V_ε, do
    Set s^v ← 0;
    m ← 0;
    for i = 1, 2, ..., n do
        m ← max_{e∈G} {m, f(χ_e)};
        V_ε^i = {(1+ε)^q | q ∈ N, m/(1+ε) ⩽ (1+ε)^q ⩽ 2km/(γ_f ε)};
        Delete all s^v, where v ∉ V_ε^i;
        for v ∈ V_ε^i do
            if s^v(G) < k, then
                α ← BinarySearch(f, s^v, c, e_i, k, (γ_f v/2^{γ_f} − f(s^v)) / (k − s^v(G)));
                s^v ← s^v + αχ_e;
            end if
        end for
    end for
end for
return s = arg max_{v∈V_ε^n} f(s^v).
```

Proof Similar to the proof of Theorem 1, we can obtain a $v \in V_\varepsilon^i$, such that $(1-\varepsilon)OPT \leqslant v \leqslant OPT$. Let $\boldsymbol{s}^v$ be the output solution with respect to v. Therefore, $f(\boldsymbol{s}^v) \geqslant \min\{(1-\varepsilon)\gamma_f/2^{\gamma_f}, (1 - 1/\gamma_f^w 2^{\gamma_f})\} \cdot OPT$.

As the return vector of Algorithm 3 satisfies

$$\boldsymbol{s}^v = \arg\max_{v \in V_\varepsilon^n} f(\boldsymbol{s}^v),$$

we can conclude that

$$f(\boldsymbol{s}) \geqslant f(\boldsymbol{s}^v) \geqslant \min\{(1-\varepsilon)\gamma_f/2^{\gamma_f}, (1 - 1/\gamma_f^w 2^{\gamma_f})\} \cdot OPT.$$

■

2.3 Improved one-pass streaming algorithm

Asides from the number of rounds to reading the data, the memory complexity is also an important index to measure the performance of a streaming algorithm. In Algorithm 3, the memory complexity relies on the amount of $v \in V_\varepsilon^i$ and the cardinality constraint k. Then, we take measures to reduce the lower amount of v in order to reduce memory complexity. The improved one-pass streaming algorithm is presented as following.

Theorem 3 For any given $\varepsilon \in (0,1)$ and function $f \in F_c$, Algorithm 4 is a one-pass $\min\{(1 -$

Algorithm 4 Streaming algorithm with one-pass

Input: $f \in F_c$, stream of data G, cardinality constraints k, $\varepsilon \in (0,1)$.

Output: Vector $\boldsymbol{s} \in \mathbf{N}^G$.

```
V_ε = {(1+ε)^q | q ∈ N};
for v ∈ V_ε, do
    Set s^v ← 0;
    ζ ← 0, m ← 0, β ← 0;
    for i = 1, 2, ..., n do
        m ← max{m, f(χ_{e_i})};
        ζ ← max{m, β};
        V_ε^i = {(1+ε)^q | q ∈ N, ζ/(1+ε) ⩽ (1+ε)^q ⩽ 2km/(γ_f ε)};
        Delete all s^v, where v ∉ V_ε^i;
        for v ∈ V_ε^i do
            if s^v(G) < k, then
                α ← BinarySearch(f, s^v, c, e_i, k, (γ_f v/2^{γ_f} − f(s^v)) / (k − s^v(G)));
                s^v ← s^v + αχ_e;
            end if
        end for
        β ← max_{v∈V_ε^i} {β, f(s^v)};
    end for
end for
return s = arg max_{v∈V_ε^n} f(s^v).
```

$\varepsilon)\gamma_f/2^{\gamma_f}, (1-1/\gamma_f^w 2^{\gamma_f})\}$-approximation algorithm for NMCC, with $O(\frac{k}{\varepsilon^2})$ memory complexity and $O(\frac{\log k}{\varepsilon}\log\frac{k}{\varepsilon})$ query times per item.

Proof We can obtain the approximation by using the same method as Theorem 2. Thus, we omit the explanation here, and skip to the analysis of the memory complexity of Algorithm 4. On the one hand, after the $(i-1)$-th iteration we obtain

$$m_i = \max_{l=1,2,\cdots i} f(\chi_{e_i}),$$

and

$$\beta_i = \max_{l=1,2,\ldots i}\max_{v^l\in V_\varepsilon^l} f({\boldsymbol s_l}^{v^l}).$$

Therefore, we have

$$\zeta_i = \max\{m_i, \beta_{i-1}\},$$

and

$$V_\varepsilon^i = \left\{(1+\varepsilon)^q \mid \frac{\zeta_i}{1+\varepsilon} \leqslant (1+\varepsilon)^q \leqslant \frac{2km_i}{\gamma_f\varepsilon}\right\}.$$

On the other hand, after the i-th iteration we have

$$\beta_i = \max\{\beta_{i-1}, \max_{v^i\in V_\varepsilon^i} f({\boldsymbol s_i}^{v^i})\}.$$

It is easy to check that $\beta_{i-1}\leqslant\beta_i$. Thus, there is no need to save the vector ${\boldsymbol s_i}^{v^i}$ with $v^i\leqslant\beta_i$. We need only to consider the vector ${\boldsymbol s_i}^{v^i}$ corresponding to these $v_p^i \in \overline{V}_\varepsilon^i = \left\{(1+\varepsilon)^t \mid \frac{\beta_i}{1+\varepsilon} \leqslant (1+\varepsilon)^t \leqslant \frac{2km_i}{\gamma_f\varepsilon}\right\}$. The number of v_p^i in $\overline{V}_\varepsilon^i$ is $\lceil\frac{1}{\varepsilon}\log\frac{2k}{\varepsilon^2}\rceil$.

From Lemma 1, we obtain

$$\boldsymbol{s_i}^{v_p^i}(G) \leqslant \frac{2^{\gamma_f}k}{\gamma_f v_p^i} f(\boldsymbol{s_i}^{v_p^i}) \leqslant \frac{2^{\gamma_f}k\beta_i}{\gamma_f(1+\varepsilon)^p\beta_i} \leqslant \frac{2^{\gamma_f}k}{\gamma_f(1+\varepsilon)^p}.$$

Then, for each iteration, the memory complexity is expressed as follows:

$$\sum_{p=0}^{\lceil\frac{1}{\varepsilon}\log\frac{2k}{\varepsilon^2}\rceil} \frac{2^{\gamma_f}k}{\gamma_f(1+\varepsilon)^p} \leqslant \frac{2k}{\varepsilon}\sum_{p=0}^{\lceil\frac{1}{\varepsilon}\log\frac{2k}{\varepsilon^2}\rceil}\frac{1}{(1+\varepsilon)^p} = O\left(\frac{k}{\varepsilon^2}\right).$$

For any $e\in G$, the number of v is at most $O(\frac{1}{\varepsilon}\log\frac{k}{\varepsilon^2})$. Meanwhile, for each $e\in G$ and per v, the query time is $O(\log k)$. Thus, for each item, the update time is $O(\frac{\log k}{\varepsilon}\log\frac{k}{\varepsilon^2})$. ■

3 Conclusion

In this paper, we first design a two-pass streaming algorithm for NMCC with $f\in F_c^{\gamma_f,\gamma_f^w}$. We combine the exponential growth method for estimating the OPT with binary search to bring down the pass of Algorithm 1. Moreover, we improve the memory complexity. Meanwhile, we also analyze the performance of the algorithm by introducing the DR ratio and weak DR ratio, including the approximation ratio, memory complexity, and the query times of each item.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 11871081), the Natural Science Foundation of Shandong Province (No. ZR2022MA034), the Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS202116), the Fundamental Research Project of Shenzhen City (No. JCYJ20210324102012033), the National Natural Science Foundation of China (No. 11901558), and the National Natural Science Foundation of China (No. 11801310).

References

[1] H. Lin and J. Bilmes, A class of submodular functions for document summarization, in *Proc. 49th Annu. Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, OR, USA, 2011, pp. 510–520.

[2] D. Kempe, J. Kleinberg, and É. Tardos, Maximizing the spread of influence through a social network, in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003, pp. 137–146.

[3] J. Lee, *Encyclopedia of Environmetrics*. New York, NY, USA: John Wiley & Sons, Ltd., 2006, pp. 1229–1234.

[4] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, Streaming submodular maximization: Massive data summarization on the fly, in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, pp. 671–680.

[5] T. Soma, N. Kakimura, K. Inaba, and K. Kawarabayashi, Optimal budget allocation: Theoretical guarantee and efficient algorithm, in *Proc. 31st Int. Conf. Machine Learning*, Beijing, China, 2014, pp. I-351–I-359.

[6] E. Balkanski, A. Rubinstein, and Y. Singer, An exponential speedup in parallel running time for submodular maximization without loss in approximation, in *Proc. 30th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Diego, CA, USA, 2019, pp. 283–302.

[7] C. Chekuri and K. Quanrud, Submodular function maximization in parallel via the multilinear relaxation, in *Proc. 30th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Diego, CA, USA, 2019, pp. 303–322.

[8] A. Das, and D. Kempe, Algorithms for subset selection in linear regression, in *Proc. 40th Annu. ACM Symp. Theory of Computing*, Victoria, Canada, 2008, pp. 45–54.

[9] A. Ene and H. L. Nguyen, Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time, in *Proc. 30th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Diego, CA, USA, 2019, pp. 274–282.

[10] A. Das and D. Kempe, Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and

dictionary selection, in *Proc. 28th Int. Conf. Machine Learning*, Washington, DC, USA, 2011, pp. 1057–1064.

[11] D. Golovin and A. Krause, Adaptive submodularity: Theory and applications in active learning and stochastic optimization, *J. Artif. Intell. Res.*, vol. 42, no. 1, pp. 427–486, 2011.

[12] S. Gong, Q. Nong, W. Liu, and Q. Fang, Parametric monotone function maximization with matroid constraints, *J. Glob. Optim.*, vol. 75, no. 3, pp. 833–849, 2019.

[13] C. Huang and N. Kakimura, Improved streaming algorithms for maximizing monotone submodular functions under a knapsack constraint, in *Proc. 16th Int. Symp. Algorithms and Data Structures*, Edmonton, Canada, 2019, pp. 438–451.

[14] A. Norouzi-Fard, J. Tarnawski, S. Mitrović, A. Zandieh, A. Mousavifar, and O. Svensson, Beyond 1/2-approximation for submodular maximization on massive data streams, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 3829–3838.

[15] A. Shioura, On the pipage rounding algorithm for submodular function maximization—a view from discrete convex analysis, *Discrete Math. Algorithms Appl.*, vol. 1, no. 1, pp. 1–23, 2009.

[16] R. Q. Yang, D. C. Xu, Y. J. Jiang, Y. S. Wang, and D. M. Zhang, Approximating robust parameterized submodular function maximization in large-scales, *Asia Pac. J. Oper. Res.*, vol. 36, no. 4, p. 1950022, 2019.

[17] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, An analysis of approximations for maximizing submodular set functions-I, *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[18] U. Feige, A threshold of ln n for approximating set cover, *J. ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[19] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, *Oper. Res. Lett.*, vol. 32, no. 1, pp. 41–43, 2004.

[20] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, Maximizing a monotone submodular function subject to a matroid constraint, *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, 2011.

[21] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, Efficient sensor placement optimization for securing large water distribution networks, *J. Water Resour. Plann. Manage.*, vol. 134, no. 6, pp. 516–526, 2008.

[22] M. Kapralov, I. Post, and J. Vondrák, Online submodular welfare maximization: Greedy is optimal, in *Proc. 24th Annu. ACM-SIAM Symp. Discrete Algorithms*, New Orleans, LA, US, 2013, pp. 1216–1225.

[23] R. Khanna, E. R. Elenberg, A. G. Dimakis, S. Negahban, and J. Ghosh, Scalable greedy feature selection via weak submodularity, in *Proc. 20th Int. Conf. Artificial Intelligence and Security*, Fort Lauderdale, FL, USA, 2017, pp. 1560–1568.

[24] N. Lawrence, M. Seeger, and R. Herbrich, Fast sparse Gaussian process methods: The informative vector machine, in *Proc. 15th Int. Conf. Neural Information Processing Systems*, Cambridge, MA, USA, 2002, pp. 625–632.

[25] Y. Lin, W. Chen, and J. C. S. Lui, Boosting information spread: An algorithmic approach, in *Proc. of the 2017 IEEE 33rd Int. Conf. Data Engineering*, San Diego, CA, USA, 2017, pp. 883–894.

[26] T. Soma and Y. Yoshida, A generalization of submodular cover via the diminishing return property on the integer lattice, in *Proc. 28th Int. Conf. Neural Information Processing Systems*, Montreal, Canada, 2015, pp. 847–855.

[27] L. A. Wolsey, Maximising real-valued submodular functions: Primal and dual heuristics for location problems, *Math. Oper. Res.*, vol. 7, no. 3, pp. 410–425, 1982.

[28] X. Zhu, J. Yu, W. Lee, D. Kim, S. Shan, and D. Z. Du, New dominating sets in social networks, *J. Glob. Optim.*, vol. 48, no. 4, pp. 633–642, 2010.

[29] A. Krause, A. Singh, and C. Guestrin, Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies, *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, 2008.

[30] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, Guarantees for Greedy maximization of non-submodular functions with applications, in *Proc. 34th Int. Conf. Machine Learning*, Sydney, Australia, 2017, pp. 498–507.

[31] I. Bogunovic, J. Zhao, and V. Cevher, Robust maximization of non-submodular objectives, in *Proc. 21st Int. Conf. Artificial Intelligence and Statistics*, Playa Blanca, Spain, 2018, pp. 890–899.

[32] A. Kuhnle, J. D. Smith, V. G. Crawford, and M. T. Thai, Fast maximization of non-submodular, monotonic functions on the integer lattice, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 2786–2795.

[33] U. Feige and R. Izsak, Welfare maximization and the supermodular degree, in *Proc. 4th Conf. Innovations in Theoretical Computer Science*, Berkeley, CA, USA, 2013, pp. 247–256.

[34] T. Horel and Y. Singer, Maximization of approximately submodular functions, in *Proc. 30th Int. Conf. Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 3053–3061.

[35] Y. J. Jiang, Y. S. Wang, D. C. Xu, R. Q. Yang, and Y. Zhang, Streaming algorithm for maximizing a monotone non-submodular function under d-knapsack constraint, *Optim. Lett.*, vol. 14, no. 5, pp. 1235–1248, 2020.

[36] T. Soma and Y. Yoshida, Maximizing monotone submodular functions over the integer lattice, *Math. Program.*, vol. 172, no. 1, pp. 539–563, 2018.

[37] Y. J. Wang, D. C. Xu, Y. S. Wang, and D. M. Zhang, Non-submodular maximization on massive data streams, *J. Glob. Optim.*, vol. 76, no. 4, pp. 729–743, 2020.

[38] Z. Zhang, D. Du, Y. Jiang, and C. Wu, Maximizing DR-submodular + supermodular functions on the integer lattice subject to a cardinality constraint, *J. Glob. Optim.*, vol. 80, no. 3, pp. 595–616, 2021.

Jingjing Tan received the PhD degree from Beijing University of Technology, China in 2016. Currently, she is an associate professor at Weifang University. Her research interests include combinatorial optimization, submodular maximization, machine learning, and approximation algorithm.

Yicheng Xu received the PhD degree in mathematics from Beijing University of Technology, China in 2018. Currently, he is an associate professor at Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences. Prior to that, he worked as a postdoctoral researcher and assistant professor at SIAT. His research interests include algorithm design and analysis among others.

Yue Sun received the MEng degree from Beijing University of Technology, China in 2022. She is a PhD candidate at Beijing University of Technology. Her research interests include approximation algorithms, submodular optimization scheduling theory, and submodular optimization and clustering.

Juan Zou received the PhD degree from Qufu Normal University, China in 2014. She is an associate professor at Qufu Normal University in China. Her research interests are in approximation algorithms, submodular optimization, and scheduling theory.