



*Research article***Streaming non-submodular maximization subject to a cardinality constraint****Meixia Li, Jiansheng Cai, Jingjing Tan*, Wenchao Wang and Xueling Zhou**

School of Mathematics and Statistics, Weifang University, Weifang 261061, China

* **Correspondence:** Email: tanjingjing1108@163.com; Tel: 15610278308.

Abstract: Many practical applications impose challenges of selecting required elements from a vast amount of data, with the requirement of maximizing non-submodular set functions. In this paper, we design two algorithms for the problem of maximizing a normalized monotone non-submodular set function with a cardinality constraint. By utilizing the concept of a diminishing-return ratio, we first propose a one-pass algorithm called NON-SUBMODULAR-SIEVE-STREAMING⁺⁺, attaining an approximation ratio of $\min\{(1 - \varepsilon)\gamma/2^\gamma, 1 - 1/2^\gamma\}$, where $\varepsilon \in (0, 1)$ is any small real number and $\gamma \in (0, 1]$ is the diminishing return ratio. Then, to reduce the adaptive complexity, we improve the algorithm and derive NON-SUBMODULAR-BATCH-SIEVE-STREAMING⁺⁺ with an approximation ratio $\min\{(1 - 3\varepsilon)\gamma/2^\gamma, 1 - 1/2^\gamma\}$ for any $\varepsilon \in (0, 1/3)$. Finally, numerical experiments are carried out to evaluate the effectiveness and feasibility of the two Streaming algorithms.

Keywords: non-submodular; cardinality constraint; streaming algorithm; diminishing-return ratio**Mathematics Subject Classification:** 90C25

1. Introduction

As society enters the big data era, emerging applications such as social networks are producing massive data in the form of videos, sensor data, etc. These data often exceed the storage capacity of an individual computer, and consequently bring a major problem of how to extract useful information from a large amount of complex data. Considered as a solution to such problems, the streaming model accommodates the data flow in which each element arrives sequentially, while only a bounded number of elements are allowed to stay in the memory during the process. In most scenarios, the objective functions to optimize for the subset from the data flow satisfy the submodularity. However, there are also many applications for which the objective function is non-submodular, such as sparse Gaussian processes [1] and experimental design [2]. This raises the problem of maximizing non-submodular set functions under cardinality constraints. Formally, the problem can be defined as follows:

$$\max_{S \subseteq E} f(S) \tag{1.1}$$

$$\text{s.t. } |S| \leq k,$$

where E is a data stream, $k \in N$, and $f : 2^E \rightarrow R_+$ is a non-negative and normalized monotone non-submodular function with the diminishing-return ratio γ .

1.1. Related work

In recent years, there have been many applications of the streaming model [3–9], as well as maximizing a submodular set function with constraints therein [7, 10–12]. For the non-submodular case in an off-line setting, Das and Kempe [13] introduced the term submodularity ratio to characterize the theoretical performance of their algorithms. Elenberg et al. [14] first designed a streaming algorithm and proved an approximation ratio that depended on the weak submodularity ratio γ_f . Wang et al. [15] used the diminishing-return ratio to design four sieve-streaming algorithms.

For the offline model, rather than the streaming model, there already exist a great deal of elegant results about submodular optimization regarding a cardinality constraint [16–19], a knapsack constraint [20–23], or a matroid constraint [24–26]. For the streaming model, Badanidiyuru et al. [27] first proposed the SIEVE-STREAMING method to maximize a monotone submodular function subject to a cardinality constraint. They proved their algorithm is with an approximation ratio $(\frac{1}{2} - \varepsilon)$, an update time $O(\log k/\varepsilon)$, and a memory cost $O(k \log k/\varepsilon)$ for any $\varepsilon > 0$. Later, Buchbinder et al. [28] proposed an improved approximation algorithm with a constant ratio $\frac{1}{4}$, while the memory complexity was improved to $\Theta(k)$. As a more recent improvement, Kazemi et al. [29] designed SIEVE-STREAMING⁺⁺, which has an identical approximation ratio $(\frac{1}{2} - \varepsilon)$ but a better practical performance, superior to the result of SIEVE-STREAMING in [27]. Unfortunately, their algorithm needs to query the oracle of each incoming item and leads to the dramatical increment of the adaptive complexity. To overcome this obstacle, Kazemi et al. [29] introduced BATCH-SIEVE-STREAMING⁺⁺, a hybrid algorithm that was shown with an approximation ratio $(\frac{1}{2} - \varepsilon)$, and with memory complexity and adaptive complexity, both correlated to the buffer size B . There were also a number of elegant algorithms that are worth mentioning in the literature [30–34].

1.2. Our results and organization

We design two streaming algorithms called NON-SUBMODULAR-SIEVE-STREAMING⁺⁺ and NON-SUBMODULAR-BATCH-SIEVE-STREAMING⁺⁺ for Problem (1.1). By using the diminishing-return ratio, we prove the approximation ratios and analyze the performance of these two algorithms.

The remainder of the paper is organized as follows: in Section 2, we introduce some definitions and results; in Section 3, we design two algorithms and analyze their performance guarantee; and in Section 4, we give numerical experimental results for evaluating the practical performance of the algorithms.

2. Preliminaries

2.1. Definitions and results

In this section, we introduce some definitions and results used throughout the paper. We use E to denote the ground set, and $f : 2^E \rightarrow R^+$ for the objective set function.

For any subsets $S \subseteq T \subseteq E$ and $e \in E \setminus T$, f is called submodular if and only if it satisfies the

diminishing marginal return property, i.e.,

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$$

holds. Equivalently, it can also be defined as

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y),$$

for any subset $X, Y \subseteq E$.

For any given $e \in E$ and $S \subseteq E$, we say the following is the marginal gain of f with respect to e and S :

$$f(\{e|S\}) = f(S \cup \{e\}) - f(S).$$

Furthermore, for any subset $S, T \subseteq E$, the above concept can be extended as follows:

$$f(S|T) = f(S \cup T) - f(T).$$

We say f is monotone if and only if $f(X) \leq f(Y)$ holds for any $X \subseteq Y$, while f is non-negative if and only if $f(S) \geq 0$ holds for any $S \subseteq E$. Besides, f is normalized if and only if $f(\emptyset) = 0$.

With the above definitions, for a normalized nonnegative monotone set function f , the Diminishing-Return Ratio and Weak Submodularity Ratio can be defined as below.

Definition 1. [15] For any $S \subseteq T$ and $e \notin T$, the diminishing-return ratio of f is the largest scalar $\gamma \in (0, 1]$ satisfying

$$\gamma f(e|T) \leq f(e|S).$$

2.2. Practical estimation of diminishing-return ratio

For most streaming optimization problems based on non-submodular objective functions, the diminishing-return ratio serves as a critical quantitative indicator to analyze the algorithmic approximation performance and convergence guarantee. Different from submodular functions with fixed and theoretically bounded diminishing-return characteristics, general non-submodular objective functions possess variable and unknown diminishing-return ratios, which makes it difficult to directly conduct a theoretical algorithm analysis and a practical algorithm deployment. To address this issue, this section presents a universal numerical estimation method for the diminishing-return ratio of general non-submodular functions. The core idea of the estimation method is to quantitatively calculate the marginal gain variation of the objective function under an incremental sample input, so as to fit and obtain the real-time diminishing-return ratio of the target function. Specifically, we continuously calculate the objective function values which correspond to different sample subset scales, record the marginal gain generated by each newly added sample, and further quantify the attenuation degree of the marginal gain to derive the accurate value range of the diminishing-return ratio.

Then, we can express γ in terms of the following:

$$\gamma = \min_{S \subseteq T, e \notin T} \frac{f(e|S)}{f(e|T)}.$$

Definition 2. [14] For an integer k , a monotone non-negative set function $f : 2^E \rightarrow R_+$ is called γ_f -weakly submodular if and only if

$$\gamma_f \leq \gamma_k := \min_{T, S \subseteq E: |T|, |S \setminus T| \leq k} \frac{\sum_{v \in S \setminus T} f(v|T)}{f(S|T)}.$$

When both the numerator and denominator are 0, we define the ratio as 1.

In the algorithm, we would like to omit the details of the computation of the value of f , so we call such a computation of the value an oracle. The time complexity of an algorithm is often measured by the number of oracles.

The diminishing-return ratio is a critical quantitative metric for characterizing the degree of non-submodularity of objective functions. For real-world non-submodular optimization tasks, it is necessary to efficiently estimate and numerically compute this ratio to support the algorithm implementation and theoretical verification. In this subsection, we present a general numerical estimation scheme for the diminishing-return ratio applicable to arbitrary non-submodular objective functions. For a general non-submodular function, we compute the marginal gain difference over randomly sampled subsets and adjacent element additions, and adopt a statistical averaging strategy to obtain a stable numerical estimation of the ratio. Specifically, we repeatedly sample feasible subsets in the discrete domain, calculate the marginal utility of adding individual elements before and after set expansion, and normalize the variation of marginal gains to obtain the numerical value of the diminishing-return ratio. With this practical estimation strategy, the diminishing-return ratio can be effectively quantified for arbitrary real-world non-submodular objectives, which greatly enhances the practicability and generalization of the two proposed streaming optimization algorithms. Importantly, we clarify a critical theoretical distinction for the above estimation scheme. The diminishing-return ratio r is theoretically defined as the minimum value over all admissible sets and feasible inputs, which serves as a rigorous theoretical bound to characterize the objective function's diminishing-return property. All our theoretical guarantees are strictly established under the prerequisite that a valid and rigorous lower bound of γ is obtained. Furthermore, the lower bound of the submodularity ratio cannot be directly transferred or equivalently adopted as the valid lower bound of the diminishing-return ratio r . It is worth noting that the randomly sampled numerical estimates of r used in our experiments are empirical values for practical algorithm execution, which cannot theoretically guarantee to be the certified lower bound of the true γ . Therefore, these estimated values are regarded as algorithmic parameters and sensitivity analysis parameters in all numerical experiments, rather than rigorous theoretical guarantees.

3. Main results

We design two streaming algorithms for Problem (1.1), and generalize the SIEVE-STREAMING⁺⁺ and BATCH-SIEVE-STREAMING⁺⁺ to non-submodular cases. In SIEVE-STREAMING⁺⁺, we reduce the storage space complexity based on two observations. The first is when we guess the value of OPT, where OPT denotes the optimal value of Problem (1.1). The Sieve-Streaming Algorithm uses the maximum value of individual elements as the lower limit of OPT. However, since $OPT \geq LB = \max_{\tau} f(S_{\tau})$, we do not need to study the case when the threshold is less than $LB/2k$. Second, given a threshold τ , the estimated interval of the threshold τ is denoted as T_{ϵ} . We can get a set S_{τ} with the number of elements therein at most LB/τ . Inspired by the idea of SIEVE-STREAMING⁺⁺ for submodular objective functions, we devise Algorithm 1 for non-submodular objective functions.

Algorithm 1**Input:** Non-submodular function f , $0 \leq \gamma \leq 1$, data stream E , k , ε .

```

1:  $\tau_{\min} \leftarrow 0$ ,  $\Delta \leftarrow 0$  and  $LB \leftarrow 0$ 
2: for  $e \in E$  do
3:    $\Delta \leftarrow \max\{\Delta, f(\{e\})\}$ 
4:    $\tau_{\min} = \frac{\max\{LB, \Delta\}}{2^{\gamma \cdot k}}$ 
5:   if  $\tau < \tau_{\min}$  then delete all sets  $S_{\tau}$ 
6:   for  $\tau$  in the set  $T_{\varepsilon} = \{(1 + \varepsilon)^i \mid i \in \mathbb{Z}^+, \tau_{\min}/1 + \varepsilon \leq (1 + \varepsilon)^i \leq \Delta/\gamma\}$ 
7:     if  $\tau$  is a new threshold
8:       then  $S_{\tau} \leftarrow \emptyset$ 
9:     if  $|S_{\tau}|$  is less than  $k$  and  $f(\{e\}|S_{\tau})$  is greater than or equal to  $\gamma \cdot \tau$ 
10:      then update  $S_{\tau}$  to  $S_{\tau} \cup \{e\}$  and  $LB$  to  $\max\{LB, f(S_{\tau})\}$ 
11: return  $\operatorname{argmax}_{S_{\tau}} f(S_{\tau})$ 

```

First, we prove the approximation ratio.

Lemma 1. *For each iteration of Algorithm 1, we can conclude that*

$$f(S_{\tau}) \geq \gamma \cdot \tau \cdot |S_{\tau}|. \quad (3.1)$$

Proof. We prove (3.1) by induction. First, and apparently, Inequality (3.1) is true if $S_{\tau} = \emptyset$ according to the definition of normalized set function. Then, we assume (3.1) holds for the set S_{τ} . Suppose Algorithm 1 adds e into S_{τ} ; then, we have

$$\begin{aligned}
 f(S_{\tau} \cup \{e\}) &\geq \gamma \cdot \tau + f(S_{\tau}) \\
 &\geq \gamma \cdot \tau + \gamma \cdot \tau \cdot |S_{\tau}| \\
 &= \gamma \cdot \tau \cdot (|S_{\tau}| + 1),
 \end{aligned} \quad (3.2)$$

where the first inequality is true according to Line 8 of Algorithm 1, and the second inequality holds by the induction hypothesis. This completes the proof. $\square$

Recall that OPT denotes the optimal value of Problem (1.1). Let $\tau^* = \frac{\text{OPT}}{2^{\gamma \cdot k}}$.

Lemma 2. *Let $\varepsilon > 0$; there exists a value $\tau \in T_{\varepsilon}$ which satisfies the following:*

$$(1 - \varepsilon) \cdot \tau^* \leq \tau < \tau^*. \quad (3.3)$$

Proof. In the following, let $S^* = \arg \max_{S \subseteq E, |S| \leq k} f(S)$, $\text{OPT} = f(S^*)$, $\tau^* = \frac{\text{OPT}}{2^{\gamma \cdot k}}$, and $\Delta = \max_{e \in E} f(\{e\})$. As a result, $\max\{\Delta, LB\} \leq \text{OPT} \leq k \cdot \Delta$. Let $S^* = \{a_1, a_2, \dots, a_{\ell}, a_{\ell+1}, \dots, a_k\}$ be the optimal solution. Without a loss of generality, we assume $S^* \setminus S_{\tau} = \{a_1, a_2, \dots, a_{\ell}\}$ and $S^* \cap S_{\tau} = \{a_{\ell+1}, \dots, a_k\}$. For each $j \in \{1, 2, \dots, \ell\}$, denote $S_j^* = \{a_1, a_2, \dots, a_j\}$ and $S_{\tau, j} \subseteq S_{\tau}$ the encountered subsets when a_j is considered during Algorithm 1. According to the definition of γ and Δ , we have the following:

$$\text{OPT} = f(S^*) = \sum_{j=1}^k f(a_j | S_{j-1}^*)$$

$$\begin{aligned}
&\leq \frac{1}{\gamma} \cdot \sum_{j=1}^k f(a_j | \emptyset) \\
&= \frac{1}{\gamma} \cdot \sum_{j=1}^k f(a_j) \\
&\leq \frac{k\Delta}{\gamma},
\end{aligned} \tag{3.4}$$

where the first inequality can be obtained according to the definition of the Diminishing-Return Ratio.

Hence, we have $\frac{OPT}{2^\gamma \cdot k} \leq \frac{k\Delta}{2^\gamma \cdot k \cdot \gamma} = \frac{\Delta}{2^\gamma \cdot \gamma} \leq \frac{\Delta}{\gamma}$, where the last inequality is true since $\gamma \in [0, 1]$. Then, by the monotonicity of f , we obtain the following:

$$\frac{OPT}{2^\gamma \cdot k} \geq \frac{\max\{LB, \Delta\}}{2^\gamma \cdot k} = \tau_{\min}.$$

As a result, $\tau_{\min} \leq \frac{OPT}{2^\gamma \cdot k} \leq \frac{\Delta}{\gamma}$.

Now, let $\tilde{i} := \lfloor \log_{1+\varepsilon} \frac{OPT}{2^\gamma \cdot k} \rfloor$ and $\tilde{\tau} := (1 + \varepsilon)^{\tilde{i}}$; we have the following:

$$\tilde{\tau} = (1 + \varepsilon)^{\tilde{i}} \leq \frac{OPT}{2^\gamma \cdot k} \leq \frac{\Delta}{\gamma}.$$

Combining the above with $\tilde{\tau} \geq \frac{OPT}{(1+\varepsilon)2^\gamma \cdot k} \geq \max\left\{(1 - \varepsilon)\frac{OPT}{2^\gamma \cdot k}, \frac{\tau_{\min}}{1+\varepsilon}\right\}$, we immediately complete the proof. $\square$

Theorem 1. Let $\varepsilon \in (0, 1)$. Then, the set S produced by Algorithm 1 satisfies $|S| \leq k$ and $f(S) \geq \min\left\{\frac{(1-\varepsilon)\gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma}\right\} OPT$, where $\gamma \in (0, 1]$ is the diminishing-return ratio.

Proof. Assume that the notations S^* and S_j^* ($j = 1, 2, \dots, k$) are the same as that in the proof of Lemma 2. By Lemma 2, there is a threshold τ such that $(1 - \varepsilon) \cdot \tau^* \leq \tau < \tau^*$. Now, consider the set S_τ . By Algorithm 1 Line 9-10, it adds elements with a margin gain of at least $\gamma\tau$ to the set S_τ . Hence, there are two cases.

Case 1 : $|S_\tau| = k$. By Lemma 1, we have the following:

$$f(S_\tau) \geq \gamma \cdot \tau \cdot k \geq \frac{\gamma \cdot (1 - \varepsilon) \cdot OPT \cdot k}{2^\gamma k} = \frac{(1 - \varepsilon)\gamma}{2^\gamma} \cdot OPT. \tag{3.5}$$

Case 2 : $|S_\tau| < k$. We have the following:

$$\begin{aligned}
f(S_\tau \cup S^*) - f(S_\tau) &= f(S_\tau \cup \{a_1, a_2, \dots, a_\ell\}) - f(S_\tau) \\
&= \sum_{j=1}^{\ell} [f(S_\tau \cup S_j^*) - f(S_\tau \cup S_{j-1}^*)] \\
&\leq \frac{1}{\gamma} \cdot \sum_{j=1}^{\ell} [f(S_{\tau,j} \cup \{a_j\}) - f(S_{\tau,j})] \\
&\leq \frac{1}{\gamma} \cdot k \cdot \gamma \cdot \tau
\end{aligned} \tag{3.6}$$

$$\begin{aligned}
&\leq \frac{1}{\gamma} \cdot k \cdot \gamma \cdot \frac{OPT}{2^\gamma k} \\
&= \frac{OPT}{2^\gamma},
\end{aligned}$$

where the first inequality also follows from the definition of the Diminishing-Return Ratio. Furthermore, by the monotonicity of f , we have the following:

$$f(S_\tau \cup S^*) \geq f(S^*) = OPT.$$

Hence,

$$OPT - f(S_\tau) = f(S^*) - f(S_\tau) \leq f(S^* \cup S_\tau) - f(S_\tau) \leq \frac{OPT}{2^\gamma}.$$

As a result, we have the following:

$$f(S_\tau) \geq (1 - \frac{1}{2^\gamma})OPT. \quad (3.7)$$

Combining with Inequality (3.5) and (3.7), we obtain the following:

$$f(S_\tau) \geq \min \left\{ \frac{(1 - \varepsilon) \cdot \gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma} \right\} \cdot OPT, \quad (3.8)$$

which completes the proof. $\square$

In the following, we propose Algorithm 2 by generalizing the BATCH-SIEVE-STREAMING⁺⁺ to the non-submodular case. Similarly, we have two aims against BATCH-SIEVE-STREAMING⁺⁺: first, the number of adaptive rounds should be near-optimal; and second, similar to SIEVE-STREAMING⁺⁺, the algorithm should have an optimal memory complexity. Therefore, when designing Algorithm 2, we continue to retain the two properties above. NSTS is the custom algorithm abbreviation for Non-submodular threshold sampling, which refers to the dedicated sampling subroutine proposed in Algorithm 3 and is invoked by the batch streaming algorithm in Algorithm 2 during the element selection process.

Algorithm 2

Input: Stream of data E , non-submodular set function f , $0 < \gamma \leq 1$, k ,

Threshold, ε , buffer $\mathcal{B}$ with a memory B .

1: $\Delta \leftarrow 0$, $\tau_{\min} \leftarrow 0$, $LB \leftarrow 0$ and $\mathcal{B} \leftarrow \emptyset$

2: **for** $e \in E$ **do**

3: $B = \mathcal{B} \cup \{e\}$

4: **if** the buffer $\mathcal{B}$ is *Threshold* percent full **then**

5: update Δ to $\max\{\Delta, \max_{e \in \mathcal{B}} f(e)\}$, $\tau_{\min}$ to $\frac{\max\{LB, \Delta\}}{2^\gamma k(1+\varepsilon)}$

6: **if** $\tau < \tau_{\min}$ **then** delete all sets S_τ

7: **for** each τ in the set $T_\varepsilon = \{(1 + \varepsilon)^i \mid \tau_{\min} \leq (1 + \varepsilon)^i \leq \Delta/\gamma\}$ **do**

8: **if** τ is a new threshold **then** $S_\tau \leftarrow \emptyset$

9: **if** $|S_\tau|$ is less than k **then** $T \leftarrow \text{NSTS}(f_{S_\tau}, \mathcal{B}, k - |S_\tau|, \tau)$

10: update S_τ to $S_\tau \cup T$

11: $LB = \max_{S_\tau} f(S_\tau)$ and $\mathcal{B} \leftarrow \emptyset$

12: **return** $\text{argmax}_{S_\tau} f(S_\tau)$

In the course of running Algorithm 2, we need to call Algorithm 3. The detail is as follows.

Algorithm 3

Input: Non-submodular set function f , $\gamma \in (0, 1]$, k , threshold τ , ε , set of buffer items $\mathcal{B}$.

```

1:  $S \leftarrow \emptyset$ 
2: while  $|\mathcal{B}|$  is greater than 0 and  $|S|$  is less than  $k$  then
3:    $\mathcal{B}$  is equal to  $\{x \in \mathcal{B} : f(\{x|S\}) \geq \gamma \cdot \tau\}$  and filter out the rest.
4:   for  $i = 1$  to  $\lceil \frac{1}{\varepsilon} \rceil$  do
5:     randomly sample  $x$  uniformly from  $\mathcal{B} \setminus S$ 
6:     if  $f(\{x|S\})$  is less than or equal to  $(1 - \varepsilon) \cdot \gamma \cdot \tau$  then
7:       break and go to Line 2
8:     if  $f(\{x|S\})$  is greater than  $(1 - \varepsilon) \cdot \gamma \cdot \tau$  then
9:       update  $S$  to  $S \cup \{x\}$ 
10:    if  $|S|$  is equal to  $k$  then
11:      return  $S$ 
12:  for  $i \in [\lceil \log_{1+\varepsilon}(1/\varepsilon) \rceil, \lceil \log_{1+\varepsilon} k \rceil - 1]$  do
13:    update  $t$  to  $\min\{\lfloor (1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i \rfloor, |\mathcal{B} \setminus S|, k - |S|\}$ 
14:    random sample set  $T$  of size  $t$  from  $\mathcal{B} \setminus S$ 
15:    if  $|S \cup T|$  is equal to  $k$  then
16:      return  $S \cup T$ 
17:    if  $\frac{f(T|S)}{|T|}$  is less than or equal to  $(1 - \varepsilon) \cdot \gamma \cdot \tau$  then
18:      update  $S$  to  $S \cup T$  and break
19:    else
20:      update  $S$  to  $S \cup T$ 
21: return  $S$ 

```

Now, we shall analyze the approximation ratio of Algorithm 2.

Theorem 2. Let $\varepsilon \in (0, \frac{1}{3})$ and S be the set returned by Algorithm 2. Then, S satisfies $|S| \leq k$ and $f(S) \geq \min\left\{\frac{(1-3\varepsilon)\gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma}\right\} OPT$, where γ is the diminishing-return ratio of f .

Proof. Assume $\mathcal{B}$ is the set of elements buffered from the stream E . Similar to the proof in Lemma 2, we define $S^* = \arg \max_{S \subseteq E, |S| \leq k} f(S)$, $OPT = f(S^*)$, $\tau^* = \frac{OPT}{2^\gamma \cdot k}$. By Lemma 2, there exists a value τ such that $(1 - \varepsilon) \cdot \tau^* \leq \tau < \tau^*$. Let $S^* = \{a_1, a_2, \dots, a_\ell, a_{\ell+1}, \dots, a_k\}$ be the optimal solution. Without a loss of generality, we assume $S^* \setminus S_\tau = \{a_1, a_2, \dots, a_\ell\}$, $S^* \cap S_\tau = \{a_{\ell+1}, \dots, a_k\}$. For each $j \in \{1, 2, \dots, \ell\}$, denote by $S_j^* = \{a_1, a_2, \dots, a_j\}$ and $A_{\tau,j} \subseteq S_\tau$ the encountered subsets when a_j is considered during Algorithm 2. Note that a_j is not added into $A_{\tau,j}$ in every iteration for each $j \in \{1, 2, \dots, \ell\}$. In the following, we focus on τ and its corresponding set of picked items S_τ . There are two cases for the set S_τ .

Case 1. If $|S_\tau| < k$, then similar to the proof of Theorem 1, we have the following:

$$\begin{aligned}
 OPT &\leq f(S^* \cup S_\tau) = f(S_\tau \cup \{a_1, a_2, \dots, a_\ell\}) \\
 &= \sum_{j=1}^{\ell} [f(S_\tau \cup S_j^*) - f(S_\tau \cup S_{j-1}^*)] + f(S_\tau)
 \end{aligned}$$

$$\begin{aligned}
&\leq \frac{1}{\gamma} \sum_{j=1}^{\ell} [f(A_{\tau,j} \cup \{a_j\}) - f(A_{\tau,j})] + f(S_{\tau}) \\
&\leq \frac{1}{\gamma} \cdot k \cdot \gamma \cdot \tau + f(S_{\tau}) \\
&\leq \frac{1}{\gamma} \cdot k \cdot \gamma \cdot \frac{OPT}{2^{\gamma}k} + f(S_{\tau}) \\
&= \frac{OPT}{2^{\gamma}} + f(S_{\tau}).
\end{aligned} \tag{3.9}$$

As a result,

$$f(S_{\tau}) \geq (1 - \frac{1}{2^{\gamma}}) \cdot OPT. \tag{3.10}$$

Case 2. $|S_{\tau}| = k$.

Assume the set S_{τ} of size k is sampled in ℓ iterations of the while-loop, and T_i is the union of sampled batches in the i -th iteration of the while-loop. Furthermore, suppose $T_{i,j}$ is the j -th sampled batch in the i -th iteration of the while loop and $S_{\tau,i,j} = \bigcup_{i,j} T_{i,j}$ is the state of set S_{τ} after the j -th batch insertion in the i -th iteration of the while-loop. Set $S_{\tau,i}$ as the state of set S_{τ} after the i -th iteration of the while-loop. In the following, we prove $f(T_i|S_{\tau,i-1}) \geq (1 - 2\epsilon) \cdot |T_i|$ by three different cases.

(1) The while-loop breaks at Line 7 of Algorithm 3. It is easy to see we have only one $T_{i,j}$. Hence, $f(T_{i,j}|S_{\tau,i,j-1}) \geq \gamma \cdot (1 - \epsilon) \cdot \tau$. Furthermore,

$$\begin{aligned}
f(T_i|S_{\tau,i-1}) &= f(S_{\tau,i-1} \cup T_i) - f(S_{\tau,i-1}) \\
&= \sum_{j=1}^{|T_i|} [f(S_{\tau,i,j}) - f(S_{\tau,i,j-1})] \\
&\geq \gamma \cdot (1 - \epsilon) \cdot \tau \cdot |T_i|.
\end{aligned} \tag{3.11}$$

(2) NON-SUBMODULAR-THRESHOLD-SAMPLING passes the first loop and does not break at Line 18 (i.e., it continues to pick items till $|S_{\tau}| = k$ or the buffer memory is empty). Assume set T_i is sampled in t iterations of the for-loop. Then, it is obvious that $f(T_{i,j}|S_{\tau,i,j-1}) \geq \gamma \cdot (1 - \epsilon) \cdot \tau \cdot |T_{i,j}|$. Hence,

$$\begin{aligned}
f(T_i|S_{\tau,i-1}) &= f((\bigcup_{j=1}^{\lceil \frac{1}{\epsilon} \rceil} T_{i,j}) \cup (\bigcup_{j=\lceil \frac{1}{\epsilon} \rceil+1}^t T_{i,j}) | S_{\tau,i,j-1}) \\
&= \sum_{j=1}^{\lceil \frac{1}{\epsilon} \rceil} f(T_{i,j} | S_{\tau,i,j-1}) + \sum_{j=\lceil \frac{1}{\epsilon} \rceil+1}^t f(T_{i,j} | S_{\tau,i,j-1}) \\
&\geq \sum_{j=1}^t \gamma \cdot (1 - \epsilon) \cdot \tau \cdot |T_{i,j}| \\
&= \gamma \cdot (1 - \epsilon) \cdot \tau \cdot |T_i|.
\end{aligned} \tag{3.12}$$

Note that the algorithm can break at Line 16 without passing the test at Line 17. If $f(T_{i,j}|S_{\tau,i,j-1}) \geq \gamma \cdot (1 - \epsilon) \cdot \tau \cdot |T_{i,j}|$, then the analysis would be exactly the same as above. Otherwise, the remainder is similar to the third case below.

(3) **NON-SUBMODULAR-THRESHOLD-SAMPLING** passes the first loop and breaks at Line 18. We consider it from two observations.

(a) Apparently, for all $1 \leq r < j$, $f(T_{i,r} | S_{\tau,i,r-1}) \geq \gamma \cdot (1 - \varepsilon) \cdot \tau \cdot |T_{i,r}|$.

(b) The number of elements which are picked at the latest iteration of the while-loop is at most ε fraction of all the elements picked, that is,

$$|T_{i,j}| \leq \varepsilon \left| \bigcup_{1 \leq r < j} T_{i,r} \right|.$$

Hence,

$$|T_i| = \left| \bigcup_{1 \leq r \leq j} T_{i,r} \right| = |T_{i,j}| + \left| \bigcup_{1 \leq r < j} T_{i,r} \right| \leq (1 + \varepsilon) \left| \bigcup_{1 \leq r < j} T_{i,r} \right|.$$

Therefore, from the monotonicity of f , we have the following:

$$\begin{aligned} f(T_i | S_{\tau,i-1}) &= f\left(\bigcup_{1 \leq r \leq j} T_{i,r} \mid S_{\tau,i-1}\right) \\ &\geq f\left(\bigcup_{1 \leq r < j} T_{i,r} \mid S_{\tau,i-1}\right) \\ &\geq \gamma \cdot (1 - \varepsilon) \cdot \tau \cdot \left| \bigcup_{1 \leq r < j} T_{i,r} \right| \\ &\geq \frac{\gamma \cdot (1 - \varepsilon) \cdot \tau \cdot |T_i|}{1 + \varepsilon} \\ &\geq \gamma \cdot (1 - 2\varepsilon) \cdot \tau \cdot |T_i|. \end{aligned} \tag{3.13}$$

From the three cases, we have the following:

$$\begin{aligned} f(S_\tau) &= \sum_{i=1}^{\ell} f(T_i \mid S_{\tau,i-1}) \\ &\geq \sum_{i=1}^{\ell} \gamma \cdot (1 - 2\varepsilon) \cdot \tau \cdot |T_i| \\ &= \gamma \cdot (1 - 2\varepsilon) \cdot \tau \cdot k \\ &\geq \gamma \cdot \frac{1 - 3\varepsilon}{2^\gamma} \cdot OPT. \end{aligned} \tag{3.14}$$

Combining (3.10) and (3.14), we eventually complete the proof. $\square$

Based on the geometric batch-size construction rule of Algorithm 3, we supplement a rigorous derivation for the key inequality that the size of the last sampled batch is bounded by an ε fraction of the previously selected elements. Let $B_1, B_a, \dots, B_i$ be the sequential batches sampled via the geometric scaling strategy, where the size of each batch follows a geometric progression governed by a fixed scaling factor. Under the iterative update mechanism of Algorithm 3, the size of each newly generated batch is strictly constrained by the predefined geometric growth rule. After a finite number of iterations, the size of the final sampled batch B_k satisfies $|B_k| \leq \varepsilon \sum_{i=1}^{k-1} |B_i|$, where $\sum_{i=1}^{k-1} |B_i|$ denotes the total cardinality of all previously selected batches. This inequality is a natural consequence of the inherent

geometric batch scaling mechanism, which restricts the scale of the final batch to a small ε -proportion of the cumulative selected element set. Moreover, we explicitly specify the theoretical property of the guarantee presented in Theorem 2. While Algorithm 3 adopts random subset sampling in its practical implementation, both the above-derived inequality and the overall performance guarantee of Theorem 2 are fully deterministic and valid for all algorithm realizations. The randomness of sampling only influences the practical execution process of the algorithm and does not compromise the worst-case theoretical bounds derived from the deterministic geometric batch-size construction. Consequently, all theoretical conclusions in Theorem 2 hold deterministically for every feasible running instance of the proposed algorithm.

For Algorithm 1, we can conclude that the memory complexity is $O(k/\varepsilon)$ and the update time is $O(\log(k/\gamma)/\varepsilon)$, while for Algorithm 2, the memory complexity is $O(B + k/\varepsilon)$ and the adaptive complexity is $O(N \log B \log(k/\gamma)/(B\varepsilon))$. Now, we compare the performance of Algorithms 1 and 2 with the Algorithms in [15], and demonstrate the details in Table 1. Reduced adaptivity is one of the essential motivations behind the development of our batch streaming optimization algorithm (Algorithm 2). To better interpret the derived adaptive complexity bound, we further clarify how three core components of the NSTS mechanism collaboratively constrain the algorithm's adaptive overhead, including buffer invocation times, dynamically maintained thresholds, and adaptive iteration rounds. First, the number of buffer calls characterizes the fundamental computational overhead caused by the element query and state update throughout the streaming process. Second, the dynamically optimized threshold parameters strictly limit the activation frequency of adaptive refinement, thus eliminating unnecessary iterative updates and providing a solid upper bound for adaptive iterations. Third, the inherent batch-processing mechanism of NSTS further restricts the total number of adaptive rounds and avoids the frequent single-element adaptive interactions required by conventional streaming methods. Collectively, these three bounded factors jointly yield the final adaptive complexity bound of Algorithm 2. This explanation explicitly validates the effectiveness of our batch-oriented design in reducing the adaptive computational complexity, enabling the proposed algorithm to achieve better practical efficiency for large-scale streaming optimization tasks.

Table 1. Performance Comparison of Our Algorithms and Algorithms in [15].

Indexes	Algorithm 1	Algorithm 2	Algorithms 3–4
Appro. ratio	$\min \left\{ \frac{(1-\varepsilon)\gamma}{2\gamma}, 1 - \frac{1}{2\gamma} \right\}$	$\min \left\{ \frac{(1-3\varepsilon)\gamma}{2\gamma}, 1 - \frac{1}{2\gamma} \right\}$	$\min \left\{ \frac{\alpha\gamma}{2\gamma}, 1 - \frac{1}{2\gamma} \right\}$
Memory	$O\left(\frac{k}{\varepsilon}\right)$	$O\left(B + \frac{k}{\varepsilon}\right)$	$O\left(\frac{k \log(k/\gamma)}{\varepsilon}\right)$
Update time	$O\left(\frac{\log(k/\gamma)}{\varepsilon}\right)$	—	$O\left(\frac{\log(k/\gamma)}{\varepsilon}\right)$

4. Numerical experiments

In this section, we use Algorithms 1–2 to solve the Bayesian A-optimality with a numerical experimental design. The code was implemented with MATLAB R2018b, and the experiments were carried out in a desktop computer with Intel(R) Core(TM) i7-8700, CPU3.20GHz, and RAM 8.00GB.

It is well known that the objective of the Bayesian experimental design [35] is to choose and perform a set of experiments to optimize a number of statistical criteria. For this purpose, Krause et al. [2] investigated a few criteria, such as the Bayesian A-optimality criterion, aiming to maximally decrease

the variance in the posterior distribution over the parameters. In general, the criterion is not submodular as shown in [2].

For each $x_i \in \mathbb{R}^d$, suppose that there exist n experimental stimuli $\{x_1, x_2, \dots, x_n\}$, which compose the data matrix $X \in \mathbb{R}^{d \times n}$. For a set $S \subseteq \mathcal{E} = \{1, 2, \dots, n\}$, let $X_S := [x_{v_1}, x_{v_2}, \dots, x_{v_s}] \in \mathbb{R}^{d \times |S|}$ be a matrix. In the linear model $y_S = X_S^\top \theta + \omega$, let $\theta \in \mathbb{R}^d$ be the parameter vector and ω be the Gaussian noise whose mean is zero and variance is σ^2 (i.e., $\omega \sim \mathcal{N}(0, \sigma^2 I)$). Moreover, let y_S be the vector that has the dependent variables. Assume the prior has the form of an isotropic Gaussian (i.e., $\theta \sim \mathcal{N}(0, \Lambda^{-1})$, $\Lambda = \beta^2 \mathbf{I}$). Then,

$$\begin{bmatrix} y_S \\ \theta \end{bmatrix} \sim \mathcal{N}(0, \Sigma), \Sigma = \begin{bmatrix} \sigma^2 I + X_S^\top \Lambda^{-1} X_S & X_S^\top \Lambda^{-1} \\ \Lambda^{-1} X_S & \Lambda^{-1} \end{bmatrix}.$$

Hence, $\Sigma_{\theta|y_S} = (\Lambda + \sigma^{-2} X_S X_S^\top)^{-1}$. Define the following:

$$\begin{aligned} F_A(S) : &= \text{tr}(\Sigma_\theta) - \text{tr}(\Sigma_{\theta|y_S}) \\ &= \text{tr}(\Lambda^{-1}) - \text{tr}((\Lambda + \sigma^{-2} X_S X_S^\top)^{-1}). \end{aligned} \quad (4.1)$$

In [36], the authors gave the bound of the submodularity ratio of (4.1), which is as follows.

Proposition 3. [36] Assume normalized stimuli (i.e., $\|x_i\| = 1, \forall i \in \mathcal{E}$). Let the spectral norm of X be $\|X\|$. Then, a) the objective in (4.1) is monotone nondecreasing, and b) its submodularity ratio γ can be lower bounded by $\frac{\beta^2}{\|X\|^2(\beta^2 + \sigma^{-2}\|X\|^2)}$.

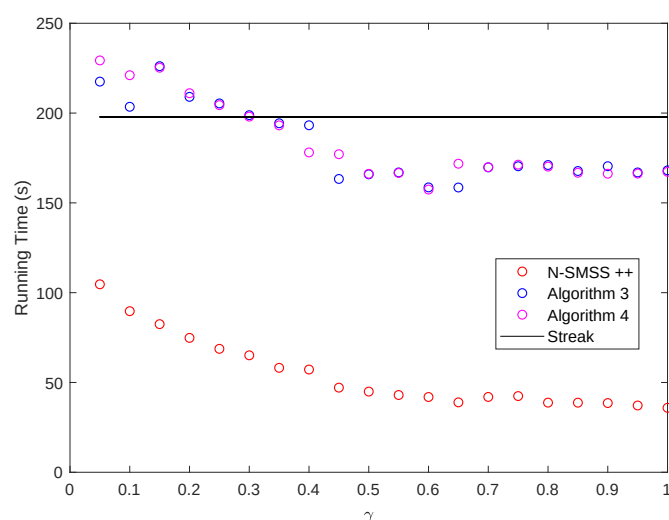
Similar to [15], the diminishing-return ratio γ is enumerated such that $\gamma = i/20$ is used for all $i = 1, \dots, 20$. Let $n = d = 500$. We consider two different values of k that are 50 and 100. We perform 10 instances and obtain the average running times and average total number of oracle queries. Furthermore, we test the relationship between memory complexity and ε or k for Algorithms 1–2. In particular, Algorithm 1 and Algorithm 2 are denoted by N-SMSS++ and N-SMBSS++, respectively. Moreover, we denote the Streak algorithm in [14] by Streak. The numerical results are shown in Figures 1–3. It's easy to see that the Streak is independent of γ . In the individual figures, we use a horizontal line to express the running time as well as the total number of queries.

Figure 1 (a) and (b) show that when the value of γ increases, it takes less time to run our Algorithm 1 as well as Algorithms 3–4 in [15]. For the same value of γ , Algorithm 1 takes even less runtime than Algorithms 3–4, thus confirming the theoretical analysis of the time complexity. Moreover, both Algorithm 1 and Algorithms 3–4 take less time than the Streak algorithm when the value of γ is sufficiently large.

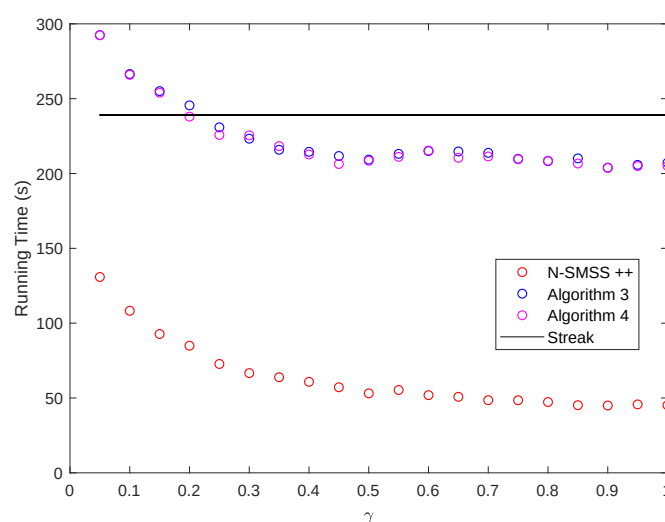
Figure 2 (a) and (b) demonstrate that the total number of querying the oracle of our Algorithm 1 and Algorithms 3–4 in [15] decreases as the value of γ increases. For the same value of γ , Algorithm 1 consumes fewer oracles than Algorithms 3–4. This result is mainly because the choice of $\tau_{\min}$ is related to the value of $f(S_\tau)$. Furthermore, the curves of Algorithm 1 and Algorithms 3–4 almost overlap, as the gap between them is small. Besides, both Algorithms 1 and Algorithms 3–4 need fewer querying oracles than the Streak algorithm for a sufficiently large value of γ .

Figure 3 (a) illustrates that the amount of consumed memory decreases with the increment of the value of ε . Moreover, the amount of memory in Algorithm 1 is less than that in Algorithm 2 for the same ε . Figure 3 (b) shows that with the increasing value of k , the amount of consumed memory also grows.

Furthermore, the amount of memory in Algorithm 1 is less than that consumed by Algorithm 2 for the same k . The comparison results demonstrate that our batch streaming algorithm achieves competitive and stable solution quality compared with the state-of-the-art offline greedy method, fully verifying the optimization effectiveness and rationality of Algorithm 2 in handling streaming non-submodular optimization tasks. Different from the memory efficiency evaluation, the above analysis further supplements the core performance evaluation targeted at Algorithm 2, which effectively demonstrates the inherent advantages of our designed batch adaptive strategy. Through the refined and comprehensive numerical discussions, the experimental validation of the proposed algorithm is significantly enhanced in completeness and persuasiveness.

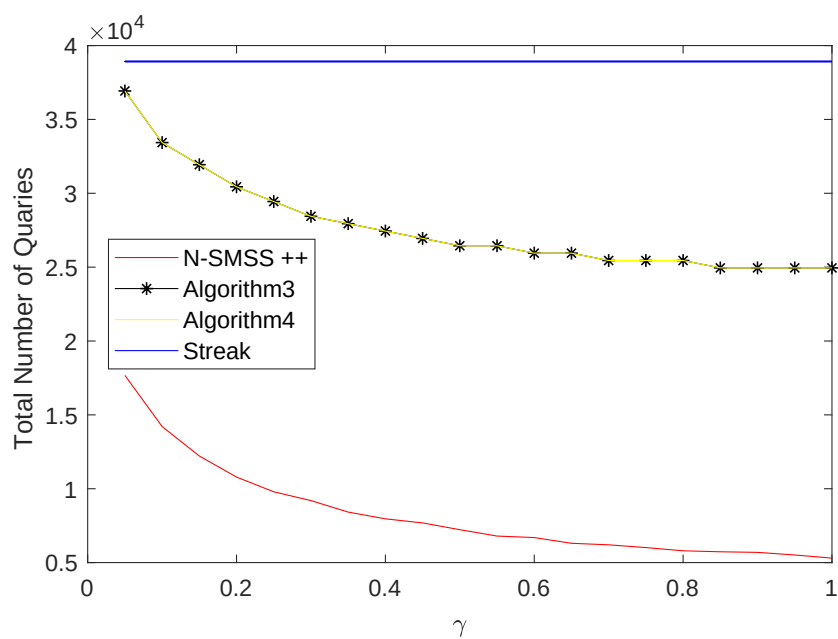
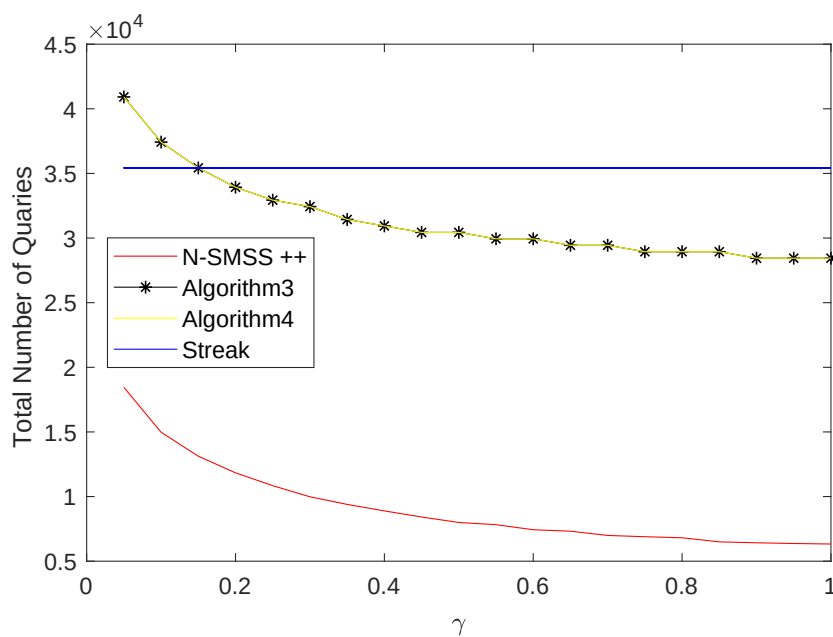


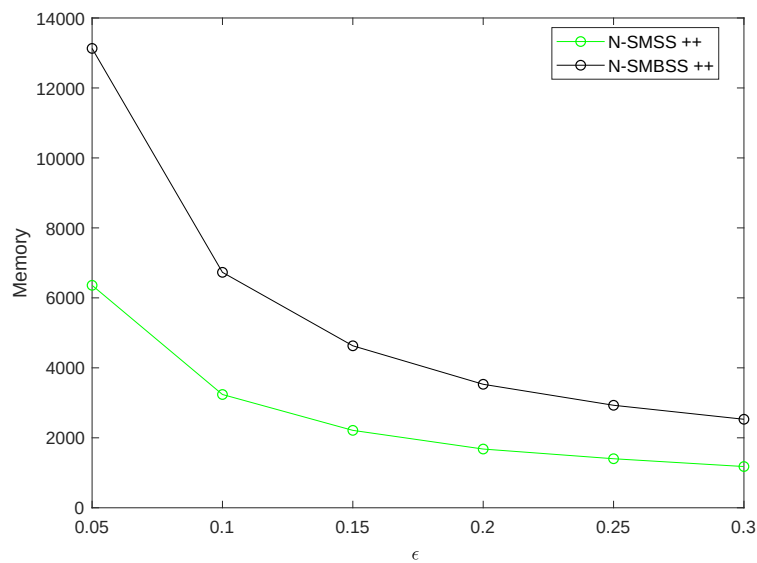
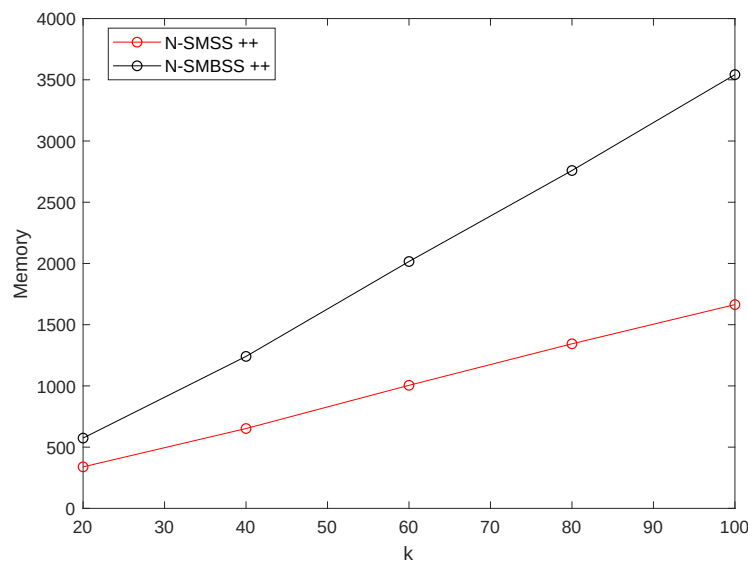
(a) Running time for $k = 50$



(b) Running time for $k = 100$

Figure 1. Performance comparison of running time over all values of γ .

(a) Total number of oracle queries for $k = 50$ (b) Total number of oracle queries for $k = 100$ **Figure 2.** Performance comparison of the total number of oracle queries over all values of γ .

(a) Relationship between memory and ϵ for $k = 100$ (b) Relationship between memory and k for $\epsilon = 0.2$ **Figure 3.** The relationship between memory and either ϵ or k .

5. Conclusions

In this paper, we designed two streaming algorithms for the problem of maximizing a normalized monotone non-submodular set function with a cardinality constraint regarding diminishing-return ratio. The first algorithm, called $\text{NON-SUBMODULAR-SIEVE-STREAMING}^{++}$, is a one-pass algorithm that generalizes $\text{SIEVE-STREAMING}^{++}$ to a non-submodular set function. We achieved an approximation ratio of $\min\left\{\frac{(1-\epsilon)\gamma}{2^\gamma}, 1 - \frac{1}{2^\gamma}\right\}$, with a memory complexity $O\left(\frac{k}{\epsilon}\right)$ and an update time $O\left(\frac{\log(k/\gamma)}{\epsilon}\right)$ per element. Through the analysis of the performance of the algorithm, we obtained the same approx-

imate ratio and update time per element but also obtained an improved memory complexity, compared with Algorithms 3–4 in [15]. In the second algorithm, called NON-SUBMODULAR-BATCH-SIEVE-STREAMING⁺⁺, we first buffered a portion of the data stream and then applied a parallel threshold for the filtering. Essentially, Algorithm 2 extends BATCH-SIEVE-STREAMING⁺⁺ but with more complicated details such that it can handle the non-submodular case. Then, the algorithm was shown with an approximation ratio $\min\left\{\frac{(1-3\varepsilon)\gamma}{2\gamma}, 1 - \frac{1}{2\gamma}\right\}$, a memory complexity $O(B + \frac{k}{\varepsilon})$, and an adaptive complexity $O(N \log B \log(k/\gamma)/(B\varepsilon))$. Lastly, we carried out numerical experiments to demonstrate the performance gain of our algorithm against Algorithms 3–4 in [15] as well as the Streak algorithm in [14]. In the future, we shall further investigate streaming non-submodular maximization with a sliding window, and non-submodular maximization with a submodularity ratio and curvature.

Acknowledgments

This work was supported by Natural Science Foundation of China (Nos. 12301417, 12071351) and Shandong Province Natural Science Foundation (No. ZR2020MA021).

Conflict of interest

The authors have no relevant financial or non-financial interests to disclose.

Author contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Meixia Li, Jiansheng Cai, Jingjing Tan, Wenchao Wang and Xueling Zhou. The first draft of the manuscript was written by Meixia Li and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Use of Generative-AI tools declaration

The author declares that they did not use any artificial intelligence (AI) tools in the creation of this article.

References

1. N. Lawrence, M. Seeger, R. Herbrich, Fast sparse Gaussian process methods: The informative vector machine, in *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2003, 625–632.
2. A. Krause, A. Singh, C. Guestrin, Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies, *J. Mach. Learn. Res.*, **9** (2008), 235–284.
3. M. Ajtai, T. S. Jayram, R. Kumar, D. Sivakumar, Approximate counting of inversions in a data stream, in *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, 2002, 370–379.
4. D. Dueck, B. J. Frey, Non-metric affinity propagation for unsupervised image categorization, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2007, 1–8.

5. K. El-Arini, C. Guestrin, Beyond keyword search: Discovering relevant scientific literature, in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011, 439–447.
6. K. El-Arini, G. Veda, D. Shahaf, C. Guestrin, Turning down the noise in the blogosphere, in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2009, 289–298.
7. R. Gomes, A. Krause, Budgeted nonparametric learning from data streams, in Proceedings of the International Conference on Machine Learning (ICML), 2010, 391–398.
8. S. Guha, N. Mishra, R. Motwani, L. O’Callaghan, Clustering data streams, in Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS), 2000, 359–366.
9. C. Reed, Z. Ghahramani, Scaling the Indian buffet process via submodular maximization, in Proceedings of the 30th International Conference on Machine Learning (ICML), PMLR 28, 2013, 1013–1021. Available from: <https://proceedings.mlr.press/v28/reed13.html>
10. D. Du, Y. Li, N. Xiu, D. Xu, Simultaneous approximation of multi-criteria submodular function maximization, *J. Oper. Res. Soc. China*, **2** (2014), 271–290. <https://doi.org/10.1007/s40305-014-0053-z>
11. B. Goldengorin, D. Ghosh, A multilevel search algorithm for the maximization of submodular functions applied to the quadratic cost partition problem, *J. Global Optim.*, **32** (2005), 65–82. <https://doi.org/10.1007/s10898-004-5909-z>
12. A. Krause, D. Golovin, Submodular function maximization, in Tractability: Practical Approaches to Hard Problems, Cambridge University Press, Cambridge, 2014, 71–104. <https://doi.org/10.1017/CBO9781139177801.004>
13. A. Das, D. Kempe, Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection, in Proceedings of the International Conference on Machine Learning (ICML), 2011, 1057–1064.
14. E. Elenberg, A. G. Dimakis, M. Feldman, A. Karbasi, Streaming weak submodularity: Interpreting neural networks on the fly, in Advances in Neural Information Processing Systems 30 (NeurIPS 2017), 2017, 4044–4054.
15. Y. Wang, D. Xu, Y. Wang, D. Zhang, Non-submodular maximization on massive data streams, *J. Global Optim.*, **76** (2020), 729–743. <https://doi.org/10.1007/s10898-019-00840-8>
16. A. Breuer, E. Balkanski, Y. Singer, The FAST algorithm for submodular maximization, arXiv:1907.06173 (2019).
17. E. Balkanski, A. Rubinstein, Y. Singer, An exponential speedup in parallel running time for submodular maximization without loss in approximation, in Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2019, 283–302.
18. U. Feige, A threshold of $\ln n$ for approximating set cover, *J. ACM*, **45** (1998), 634–652. <https://doi.org/10.1145/285055.285059>
19. G. L. Nemhauser, L. A. Wolsey, M. L. Fisher, An analysis of approximations for maximizing submodular set functions—I, *Math. Program.*, **14** (1978), 265–294. <https://doi.org/10.1007/BF01588971>

20. A. Badanidiyuru, J. Vondrák, Fast algorithms for maximizing submodular functions, in Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2014, 1497–1514.
21. A. Ene, H. Nguyen, A nearly-linear time algorithm for submodular maximization with a knapsack constraint, in Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP), 2019, 53:1–53:12.
22. A. Kulik, H. Shachnai, T. Tamir, Approximations for monotone and nonmonotone submodular maximization with knapsack constraints, *Math. Oper. Res.*, **38** (2013), 729–739. <https://doi.org/10.1287/moor.2013.0592>
23. M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, *Oper. Res. Lett.*, **32** (2004), 41–43. [https://doi.org/10.1016/S0167-6377\(03\)00062-2](https://doi.org/10.1016/S0167-6377(03)00062-2)
24. N. Buchbinder, M. Feldman, M. Garg, Deterministic $(\frac{1}{2}) + \varepsilon$ -approximation for submodular maximization over a matroid, in Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2019, 241–254.
25. Y. Filmus, J. Ward, Monotone submodular maximization over a matroid via non-oblivious local search, *SIAM J. Comput.*, **43** (2014), 514–542. <https://doi.org/10.1137/130920277>
26. J. Vondrák, Optimal approximation for the submodular welfare problem in the value oracle model, in Proceedings of the 40th ACM Symposium on Theory of Computing (STOC), 2008, 67–74.
27. A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, A. Krause, Streaming submodular maximization: Massive data summarization on the fly, in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2014, 671–680.
28. N. Buchbinder, M. Feldman, R. Schwartz, Online submodular maximization with preemption, in Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2015, 1202–1216.
29. E. Kazemi, M. Mitrović, M. Zadimoghaddam, S. Lattanzi, A. Karbasi, Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity, in Proceedings of the 36th International Conference on Machine Learning (ICML), PMLR 97, 2019, 3311–3320.
30. J. Tan, R. Yang, Y. Zhang, M. Zhu, Streaming algorithms for non-submodular functions maximization with d -knapsack constraint on the integer lattice, *Asia-Pac. J. Oper. Res.*, **40** (2023), Article ID 107766 (16 pages). <https://doi.org/10.1142/S0217595923400183>
31. X. Zhao, M. Li, B. Qu, CQ-like algorithm for nonconvex split feasibility problem with multiple output sets, *Asia-Pac. J. Oper. Res.*, **43** (2026), 1–28. <https://doi.org/10.1142/S0217595925500289>
32. X. Zhao, M. Li, B. Qu, Inertial alternating projection gradient algorithm for nonconvex split feasibility problem with multiple output sets, *J. Ind. Manag. Optim.*, **21** (2025), 6190–6218. <https://doi.org/10.3934/jimo.2025128>
33. M. Feldman, A. Karbasi, E. Kazemi, Do less, get more: Streaming submodular maximization with subsampling, in Advances in Neural Information Processing Systems 31 (NeurIPS 2018), 2018, 730–740.
34. B. Mirzasoleiman, S. Jegelka, A. Krause, Streaming non-monotone submodular maximization: Personalized video summarization on the fly, arXiv:1706.03583 (2018).

-
35. K. Chaloner, I. Verdinelli, Bayesian experimental design: A review, *Stat. Sci.*, **10** (1995), 273–304. <https://doi.org/10.1214/ss/1177009939>
 36. A. A. Bian, J. M. Buhmann, A. Krause, S. Tschitschek, Guarantees for greedy maximization of non-submodular functions with applications, in Proceedings of the 34th International Conference on Machine Learning (ICML), PMLR 70, 2017, 498–507. Available from: <https://proceedings.mlr.press/v70/bian17a.html>



AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)