
Streaming Stochastic Submodular Maximization with On-Demand User Requests

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 We explore a novel problem in streaming submodular maximization, in-
2 spired by the dynamics of news-recommendation platforms. We consider a
3 setting where users can visit a news website at any time, and upon each
4 visit, the website must display up to k news items. User interactions are in-
5 herently stochastic: each news item presented to the user is consumed with
6 a certain acceptance probability by the user, and each news item covers
7 certain topics. Our goal is to design a streaming algorithm that maximizes
8 the expected total topic coverage. To address this problem, we establish
9 a connection to submodular maximization subject to a matroid constraint.
10 We show that we can effectively adapt previous methods to address our
11 problem when the number of user visits is known in advance or linear-size
12 memory in the stream length is available. However, in more realistic sce-
13 narios where only an upper bound on the visits and sublinear memory is
14 available, the algorithms fail to guarantee any bounded performance. To
15 overcome these limitations, we introduce a new online streaming algorithm
16 that achieves a competitive ratio of $1/8\delta$, where δ controls the approxima-
17 tion quality. Moreover, it requires only a single pass over the stream, and
18 uses memory independent of the stream length. Empirically, our algorithms
19 consistently outperform the baselines.

20 1 Introduction

21 Streaming submodular maximization has been extensively studied under various feasibility
22 constraints, including *matchoid* [7, 8, 10], *matroid* [9], *knapsack* [21, 22] and *k-set* [20]
23 constraints. In the classical setting, the goal is to select a single subset of items from a data
24 stream and output it when the stream ended. However, this classical framework falls short in
25 many real-world scenarios, for example, when seeking to maximize *content coverage* in online
26 news recommendation. In such a setting, users visit a news website multiple times and expect
27 to receive a relevant set of articles on each visit. Meanwhile, new articles continuously arrive
28 at the content-provider’s server in a stream fashion. The long-term objective is to maximize
29 the total content coverage across user visits. Existing streaming submodular maximization
30 algorithms are ill-suited for this use case, as they are designed to produce a single solution
31 and do not support multiple or on-demand user interactions during the stream.

32 Motivated by this limitation, we introduce a novel problem called *Streaming Stochastic Sub-*
33 *modular Maximization with On-demand Requests (S3MOR)*. While the setting is broadly
34 applicable to streaming submodular maximization tasks with multiple output requirements,
35 we focus on the news recommendation task for illustration. Therefore, in *S3MOR*, the data
36 stream consists of incoming news items that the system aggregates into a personalized news

Table 1: Comparison of our algorithms. $N \in \mathbb{N}$ denotes the stream length, $k \in \mathbb{N}$ number of news items to present per user access, $T \in \mathbb{N}$ and $T' \in \mathbb{N}$ are the exact number and upper bound of user accesses, and $\delta \in \mathbb{N}$ an approximation parameter.

Algorithm	Competitive ratio	Time	Space
LMGREEDY	1/2	$\mathcal{O}(NT'k)$	$\mathcal{O}(N + kT)$
STORM	$1/4(T' - T + 1)$	$\mathcal{O}(NT'k)$	$\mathcal{O}(T'k)$
STORM++	$1/8\delta$	$\mathcal{O}(NT'^2k/\delta)$	$\mathcal{O}(T'^2k/\delta)$

website. Each item V_i covers a set of topics and is associated with a probability p_i , representing the probability that a user will click on the item when it is presented. A user may visit the system arbitrarily often and at any time during the stream. At each visit, the system must select and present a subset of at most k items from those that have arrived in the stream up to that point. If a user clicks on item V_i , they are exposed to all topics associated with that item. The objective is to maximize the expected total number of unique topics covered by all user clicks across all visits. This setting is challenging due to its online character:

- **On-demand:** At any point, the system must be ready to output a subset of news items.
- **Irrevocability:** Once a subset is presented to the user, it cannot be modified.

To the best of our knowledge, no existing algorithm directly addresses the *S3MOR* problem.

Our contributions. We develop approximation algorithms for the novel problem of streaming stochastic submodular maximization with on-demand requests (*S3MOR*). Our algorithms are based on reducing *S3MOR* to *streaming submodular maximization under a partition matroid constraint* such that our framework can use existing online streaming algorithms to solve *S3MOR*. We analyze the competitive ratios of the obtained algorithms:

- We prove that provided sufficient memory to store the whole stream, a greedy algorithm can achieve the best possible 1/2 competitive ratio for the *S3MOR* problem.
- If the exact number T of user visits is known, the reduction preserves the competitive ratio of the underlying online streaming submodular algorithm.
- However, in realistic scenarios, the number of user visits T is unknown and we need to guess the number of visits as $T' > T$. For this case, we introduce an algorithm STORM, which has a competitive ratio of $\frac{1}{4(T'-T+1)}$.
- To achieve a better competitive ratio, we propose a second algorithm STORM++ that makes multiple concurrent guesses of T , maintaining one solution per guess. It then greedily aggregates outcomes across guesses. This algorithm achieves a competitive ratio of $1/8\delta$, where δ is a tunable parameter that balances efficiency and solution quality. The space and time complexity increase by a factor of $\delta T'$ relative to the underlying streaming algorithm.

We summarize the competitive ratio, space and time complexity for all algorithms mentioned above in Table 1. We validate our approach through empirical experiments on both large-scale and small-scale real-world datasets. Our results show the effectiveness of our algorithms in terms of coverage quality and memory usage, in comparison to relevant baselines.

Notation and background. We assume familiarity with constrained submodular maximization and competitive analysis. We provide formal definitions in Appendix A in the supplementary materials.

Related work

Submodular optimization plays a fundamental role in machine learning, combinatorial optimization, and data analysis [13, 26, 29], capturing problems with diminishing returns, and finding applications in data summarization [34, 42], non-parametric learning [17, 45], recommendation systems [1, 33, 38, 44], influence maximization [24], and network monitoring [13]. The classical greedy algorithm achieves a $(1 - 1/e)$ -approximation for monotone submodular maximization under cardinality constraints [39], and extends to matroid constraints [6, 12].

We study stochastic submodular coverage functions, where the objective depends on random realizations but selections must be made non-adaptively. This contrasts with adaptive submodularity [16, 18, 37, 41], which models sequential decision-making with feedback and supports strong guarantees for greedy strategies. Related work includes stochastic variants of set and submodular cover, where the objective itself is random; notable examples analyze adaptivity gaps and approximation bounds under oracle access [15, 19].

Classical greedy algorithms for (constrained) submodular optimization [27, 35, 39] assume random access to the full dataset, which is infeasible in large-scale or streaming settings. This challenge has motivated streaming algorithms that make irrevocable decisions under memory constraints [4, 5, 9, 22, 23, 36]. Buchbinder et al. [5] apply the preemption paradigm, i.e., replacing elements with better ones, to achieve a tight $(1/2 - \varepsilon)$ -approximation for monotone functions under cardinality constraints. Sieve++ [23] uses thresholding to achieve a $(1/2 - \varepsilon)$ -approximation under cardinality constraints using only $\mathcal{O}(k)$ memory. Extensions to matroid and partition constraints include [7, 8, 10]; for instance, Chekuri et al. [8] gives a $1/4p$ -competitive online streaming algorithm for monotone p -matchoids. These approaches, however, do not handle stochastic item utilities or repeated, unpredictable user access. We address this gap by studying stochastic submodular coverage under partition constraints in a streaming model with limited memory and unknown user access requests.

To extend submodularity to sequences, Alaei et al. [1] introduce sequence-submodularity and sequence-monotonicity, showing that a greedy algorithm achieves a $(1 - 1/e)$ -approximation, with applications in online ad allocation. Similarly, Ohsaka and Yoshida [40] propose k -submodular functions to model optimization over k disjoint subsets, providing constant-factor approximations for monotone cases. In contrast to their settings, our problem involves making irrevocable decisions as an incoming stream of items progresses, requiring each of the k subsets to be constructed incrementally rather than all at once after seeing the entire input.

Finally, diversity-aware recommendation and coverage problems are closely related, as many systems must recommend item sets rather than single items, motivating submodular approaches to maximize coverage and user satisfaction [2, 43]. Ashkan et al. [3] propose a greedy offline method for maximizing utility under a submodular diversity constraint. Yue and Guestrin [44] use submodular bandits for diversified retrieval, learning user interaction probabilities adaptively rather than relying on a known oracle, as in our setting.

2 Streaming submodular maximization with on-demand requests

Problem setting. Consider $\mathcal{V} = \{V_1, V_2, \dots\}$ denoting the set of all items appearing in the news item stream. Each news item $V_i \in \mathcal{V}$ covers a subset of topics from a predefined set of topics $C = \{c_1, c_2, \dots, c_d\}$, where d denotes the total number of topics. Formally, $V_i \subseteq C$ for each $i \in [n]$. Additionally, each news item V_i is associated with a probability $p_i \in [0, 1]$, modeling the probability that a user will click the item when presented to them. When a user clicks a news item, we say that the user is exposed to all topics in the news item.

We study a streaming setting. At each time step i , a news item $V_i \in \mathcal{V}$ arrives in the system. Moreover, we assume that we observe a binary variable $\tau_i \in \{0, 1\}$ indicating whether the user accesses the system at time step i . Specifically, $\tau_i = 1$ denotes an access at time step i , and $\tau_i = 0$ otherwise. Whenever $\tau_i = 1$, the system should present at most k news items from the available news items $\{V_1, \dots, V_i\}$. Now, suppose a user visits the system T times. The system outputs T sets $\mathcal{S}^1, \dots, \mathcal{S}^T$, where \mathcal{S}^t represents the items presented to the user at their t -th visit, for $t \in [T]$. We allow the same item to be presented at multiple visits; however, we treat each appearance as a distinct copy. We make this choice as the user has multiple chances to click on an item if the item is presented multiple times. Therefore, for any $t \neq s$, we have $\mathcal{S}^t \cap \mathcal{S}^s = \emptyset$. We define the set of all items presented to the user over the entire sequence of visits as $\mathcal{S} = \bigcup_{t=1}^T \mathcal{S}^t$.

Objective function. We define $f(\mathcal{S})$ as the expected number of topics covered by the user after all T visits, where the expectation is taken over the probabilities of the user clicking at the news items. To formally define $f(\mathcal{S})$, we introduce the following notation. Let $\sigma_j \in \{0, 1\}$ be an indicator whether topic c_j is covered after the user has been shown the

news sets $\mathcal{S}^1, \dots, \mathcal{S}^T$; that is, whether the user clicks on at least one news item that covers topic c_j . The probability that topic c_j is not covered, denoted by $\Pr(\sigma_j = 0)$, is equal to the probability that the user does not click on any of the news items that cover topic c_j .

Since the user clicks on item V_i with probability p_i , the probability that they do not click on it is $1 - p_i$ and, thus, $\Pr(\sigma_j = 0) = \prod_{t \in [T]} \prod_{V_i \in \mathcal{S}^t, V_i \ni c_j} (1 - p_i)$, leading to the expectation $\mathbb{E}[\sigma_j] = 1 - \prod_{t \in [T]} \prod_{V_i \in \mathcal{S}^t, V_i \ni c_j} (1 - p_i)$. By linearity of expectation, the expected number of topics covered by a user, which we denote by $f(\mathcal{S})$, is given by

$$f(\mathcal{S}) = \mathbb{E} \left[\sum_{j=1}^d \sigma_j \right] = \sum_{j=1}^d \mathbb{E}[\sigma_j] = \sum_{j=1}^d \left(1 - \prod_{t \in [T]} \prod_{V_i \in \mathcal{S}^t, V_i \ni c_j} (1 - p_i) \right). \quad (1)$$

Lemma 2.1. $f(\mathcal{S})$ is a non-decreasing submodular set function.

Problem definition. We formally define our new problem *streaming stochastic submodular maximization with on-demand requests (S3MOR)* below.

Problem 2.2 (S3MOR). We define a news item stream \mathbf{S} as a sequence of triples: $\mathbf{S} = ((V_1, p_1, \tau_1), \dots, (V_N, p_N, \tau_N))$ where N is the (unknown) length of the stream. At each time step i , the system receives a triple (V_i, p_i, τ_i) , where $V_i \in \mathcal{V}$, $p_i \in [0, 1]$, and $\tau_i \in \{0, 1\}$. Whenever $\tau_i = 1$, the system selects up to k news items from the set of items $\{V_1, \dots, V_i\}$ received so far to present to the user. Let $\mathcal{S} = \{\mathcal{S}^1, \dots, \mathcal{S}^T\}$ denote the sets of news items presented to the user over T accesses. The objective is to maximize the expected number of distinct topics the user is exposed to by the end of the stream, measured by the function $f(\mathcal{S})$.

There are two key challenges in our streaming setting: (i) the system does not know in advance when or how many times a user will access it; and (ii) the system has to make decisions based on the information available so far, and these decisions are irrevocable. In the following, we show how the problem can be reduced to a submodular maximization problem subject to a partition matroid constraint. We then leverage this connection to develop solutions under various memory constraints.

Reduction to submodular maximization under a partition matroid constraint.

We first reduce the S3MOR problem (Problem 2.2) to submodular maximization under a partition matroid constraint, which enables us to apply efficient streaming algorithms with provable guarantees. Let $r: t \mapsto r_t$ map a user's t -th access to its access time, and $(r_t)_{t=1}^T$ be the sequence of times at which the user accesses the system over an unknown time period. Whenever the user accesses the system, it presents up to k news items selected from all the news items available since the beginning of the stream, denoted by $\{V_i\}_{i=1}^{r_t}$.

Let V_i^t be a copy of the item V_i associated with the t -th visit. Let \mathcal{V}^t be a collection of these copies until the t -th visit, namely, $\mathcal{V}^t = \{V_i^t\}_{i=1}^{r_t} = \{V_1^t, \dots, V_{r_t}^t\}$. Without loss of generality, we treat each copy as a distinct item, i.e., $V_i^t \neq V_i^s$ for all $t \neq s$. Clearly, $\mathcal{V}^t \cap \mathcal{V}^s = \emptyset$ for $t \neq s$.

Moreover, let $\mathcal{S}^t \subseteq \mathcal{V}^t$ be the set of news items that the system presents to the user at their t -th visit. Let $\mathcal{S} = \bigcup_{q=1}^T \mathcal{S}^q$ be the union of all sets of presented items. Analogously to

Equation (1), we adapt the function $f: 2^{\bigcup_{t=1}^T \mathcal{V}^t} \rightarrow \mathbb{R}$ such that $f(\mathcal{S})$ is the expected number of topics covered by the (copies) of news items that the user clicked on after $\mathcal{S}^1, \dots, \mathcal{S}^T$ have been presented to the user.

Observation 2.3. Let $(\bigcup_{t=1}^T \mathcal{V}^t, \mathcal{F})$ be a set system, where \mathcal{F} is a collection of subsets of $\bigcup_{t=1}^T \mathcal{V}^t$, and \mathcal{F} is constructed in the following way: for any set $X \subseteq \bigcup_{t=1}^T \mathcal{V}^t$, if $|X \cap \mathcal{V}^t| \leq k$ for any $1 \leq t \leq T$, then $X \in \mathcal{F}$. Observe that $(\bigcup_{t=1}^T \mathcal{V}^t, \mathcal{F})$ is a partition matroid. Problem 2.2 can be equivalently formulated as $\max_{X \in \mathcal{F}} f(X)$.

Based on Observation 2.3, we develop a baseline algorithm for the Problem 2.2. Suppose that the system stores all the news items up to when the user accesses the system at the t -th time, and also has stored all the previous news item sets presented to the user, i.e., $\bigcup_{q=1}^{t-1} \mathcal{S}^q$. In this case, the stored news items up to time r_t constitute one partition of the matroid, i.e., $\mathcal{V}^t = \{V_i^t\}_{i=1}^{r_t}$. We then apply the local greedy algorithm by Fisher et al. [12],

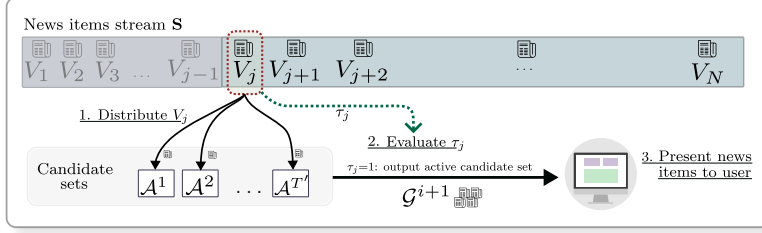


Figure 1: Schematic view of our algorithm STORM. We first initiate T' empty active candidate sets. For each incoming news item in the stream \mathbf{S} , we decide whether it can be added to/swapped into each of the active candidate set. When a user submits a request, i.e., $\tau_j = 1$, we select the best active candidate set, present it to the user, and deactivate it.

179 i.e., the system greedily selects k news items from \mathcal{V}^t that maximize the marginal gain
 180 of the objective function with respect to $f(\cup_{q=1}^{t-1} \mathcal{S}^q)$. This algorithm does not require any
 181 knowledge of T , and has a competitive ratio of $1/2$. Since all incoming news items need to
 182 be stored, the memory usage is $\Omega(N + kT)$, and the running time is $\sum_{q=1}^T \mathcal{O}(|\mathcal{V}^q|k)$. We
 183 present the algorithm in Algorithm 3 (LMGREEDY) in Appendix B.

184 **Theorem 2.4.** *Algorithm 3 for Problem 2.2 is $1/2$ -competitive. Moreover, the competitive*
 185 *ratio of $1/2$ is tight, i.e., for Problem 2.2, no streaming algorithm can achieve a competitive*
 186 *ratio better than $1/2$ without violating the irrevocability constraint.*

187 The drawback of LMGREEDY is that its memory usage depends linearly on the stream
 188 length, making it infeasible for large streams. Thus, we consider the setting where the
 189 system has limited memory and can store only a user-specific subset of news items.

190 3 Algorithms with limited memory

191 We now consider the more realistic case where the system does not have unlimited memory.
 192 First, we note that there is a lower bound of $\Theta(kT)$ that is necessary for achieving a bounded
 193 competitive ratio. Consider an adversarial example where the system has only $\Theta(kT')$
 194 memory, with $T' < T$. The stream contains more than $\Theta(kT)$ items, each covering a
 195 distinct topic, with the users click probability equal to 1. After the final item arrives, the
 196 user submits $\Theta(T)$ requests. Since the system can store at most $\Theta(kT')$ items, no algorithm
 197 can achieve a competitive ratio better than $\Theta(T'/T)$, which becomes arbitrarily poor as
 198 $T' \ll T$. Therefore, in the following, we assume an upper bound T' on the number of user
 199 visits. Furthermore, the system has available memory $\Omega(kT')$, and hence, it is possible to
 200 store the news items that may be presented across these visits.

201 3.1 A first memory-efficient streaming algorithm

202 Our first memory-efficient algorithm processes the incoming stream \mathbf{S} as follows. First, it
 203 initializes T' empty candidate sets \mathcal{A}^i for $i \in [T']$, all of which are initially *active*. Upon
 204 the arrival of a news item from the stream \mathbf{S} , for each active candidate set, the algorithm
 205 either adds the item to the set or replaces an existing item if the set is full. When a user
 206 accesses the system, the algorithm selects one of the active candidate sets to present to the
 207 user. The chosen candidate set is then marked as *inactivate* and will not be updated further.
 208 Figure 1 shows an overview.

209 Based on our reduction to submodular maximization under a partition matroid constraint,
 210 we can adapt any streaming algorithm designed for matroid constraints, provided it produces
 211 irrevocable outputs. Notable examples include the algorithms proposed by Chakrabarti and
 212 Kale [7], Chekuri et al. [8], and Feldman et al. [10], all of which achieve a $1/4$ approximation
 213 ratio. In our work, we specifically adapt the algorithm introduced by Chekuri et al. [8].

214 Our algorithm, named STORM, is shown in Algorithm 1. It keeps track of the number of
 215 visits using a counter i . Assume the user has visited i times when news item V_j arrives.

Algorithm 1: STORM

Input: Estimated visits $T' \in \mathbb{N}$, budget $k \in \mathbb{N}$.

```

1  $\mathcal{A}^1, \dots, \mathcal{A}^{T'} \leftarrow \emptyset, \mathcal{T} = \{1, \dots, T'\}, \mathcal{G}^0 \leftarrow \emptyset,$ 
    $i \leftarrow 0$ 
2 for  $(V, \tau, p)$  in the stream do
3   for  $j \in \mathcal{T}$  do
4     if  $|\mathcal{A}^j| < k$  then  $\mathcal{A}^j \leftarrow \mathcal{A}^j \cup \{V\}$ 
5     else
6        $V' \leftarrow \arg \min_{V' \in \mathcal{A}^j} \nu(f, \cup_{q \in [T']} \mathcal{A}^q, V)$ 
7       if  $f(V | \cup_{q \in [T']} \mathcal{A}^q) \geq$ 
          $2\nu(f, \cup_{q \in [T']} \mathcal{A}^q, V')$  then
          $\mathcal{A}^j \leftarrow \mathcal{A}^j \setminus \{V'\} \cup \{V\}$ 
8   if  $\tau = 1$  then
9      $j^* \leftarrow \arg \max_{j \in \mathcal{T}} f(\mathcal{A}^j | \bigcup_{t=0}^i \mathcal{G}^t)$ 
10     $\mathcal{T} \leftarrow \mathcal{T} \setminus \{j^*\}$   $\triangleright$  Deactivate candidate set
        $\mathcal{A}^{j^*}$ 
11     $\mathcal{G}^{i+1} = \mathcal{A}^{j^*}$ 
12    output  $\mathcal{G}^{i+1}$ 
13     $i \leftarrow i + 1$ 
```

Algorithm 2: STORM++

Input: Estimated visits $T' \in \mathbb{N}$, budget $k \in \mathbb{N}$, parameter $\delta \in \mathbb{N}$.

```

1  $\mathcal{G}^0 \leftarrow \emptyset, \mathcal{B}^0 \leftarrow \emptyset, i \leftarrow 0,$ 
    $\mathcal{P} \leftarrow \left\{ \delta, 2\delta, \dots, \lceil \frac{T'}{\delta} \rceil \delta \right\}$ 
2 for  $\tilde{T} \in \mathcal{P}$  do Initialize STORM  $(\tilde{T}, k) \triangleright$ 
   Maintain STORM for different  $\tilde{T}$ 
3 for  $(V, \tau, p)$  in the stream do
4   for  $\tilde{T} \in \mathcal{P}$  do
5      $\mathcal{G}^{i+1, (\tilde{T})} \leftarrow \text{STORM}(\tilde{T},$ 
        $k).step(V, \tau, p) \triangleright$  Process
        $(V, \tau, p)$  by executing lines 3-13
       of Algorithm 1
6   if  $\tau = 1$  then
7      $\tilde{T}^* \leftarrow \arg \max_{\tilde{T} \in \mathcal{P}} f(\mathcal{G}^{i+1, (\tilde{T})} |$ 
        $\bigcup_{j=0}^i \mathcal{B}^j)$ 
8      $\mathcal{B}^{i+1} \leftarrow \mathcal{G}^{i+1, (\tilde{T}^*)}$ 
9     output  $\mathcal{B}^{i+1}$ 
10     $i \leftarrow i + 1$ 
```

216 This implies that, prior to the arrival of V_j , i active sets from i partitions have already
 217 been presented to the user, and thus, neither V_j nor any subsequent item will be added to
 218 these partitions. The algorithm iterates over the remaining $T' - i$ active candidate sets, and
 219 determines whether V_j should be added to any of them, with replacement if necessary.

220 The decision to replace an existing item V with the current item V_j is based on the com-
 221 parison of their incremental value with respect to set $\cup_{t \in [T']} \mathcal{A}^t$. Specifically, we adopt the
 222 notation $\nu(f, \cup_{t \in [T']} \mathcal{A}^t, V)$ from Chekuri et al. [8] and define

$$\nu(f, \cup_{t \in [T']} \mathcal{A}^t, V) = f_{\widehat{\cup_{t \in [T']} \mathcal{A}^t}}(V), \text{ with } \widehat{\cup_{t \in [T']} \mathcal{A}^t} = \{V' \in \cup_{t \in [T']} \mathcal{A}^t : V' < V\}.$$

223 Here, $V' < V$ indicates item V' is added to the set $\cup_{t \in [T']} \mathcal{A}^t$ before item V . For a specific
 224 set $\mathcal{A}^{t'}$, V_j replaces an item in $\mathcal{A}^{t'}$ if $f_{\cup_{t \in [T']} \mathcal{A}^t}(V_j) \geq 2 \cdot \min_{V' \in \mathcal{A}^{t'}} \nu(f, \cup_{t \in [T']} \mathcal{A}^t, V)$.

225 Upon the t -th user visit, the system selects which active candidate set to output in a greedy
 226 manner. To be specific, let $\mathcal{G}^1, \dots, \mathcal{G}^{t-1}$ denote the candidate sets that have already been
 227 presented to the user, and \mathcal{T} represent the indices of the remaining active sets. The system
 228 selects the active set that offers the highest incremental gain with respect to the union of
 229 previously presented sets. Formally, let $j^* = \arg \max_{j \in \mathcal{T}} f(\mathcal{A}^j | \bigcup_{j=1}^{t-1} \mathcal{G}^j)$, and the system
 230 outputs $\mathcal{G}^t = \mathcal{A}^{j^*}$.

231 **Theorem 3.1.** *Let T be the number of user accesses and T' a given upper bound. Alg. 1*
 232 *has competitive ratio $\frac{1}{4(T'-T+1)}$, space complexity $\mathcal{O}(T'k)$, and time complexity $\mathcal{O}(NkT')$.*

233 If the exact number of visits T is known, we can set the upper bound $T' = T$ and obtain a
 234 $1/4$ -competitive ratio using Algorithm 1. However, the competitive ratio of Algorithm 1 can
 235 be arbitrarily bad if T' is significantly larger than T . Note that our analysis is tight. We
 236 can demonstrate that for any fixed T , the competitive ratio of Algorithm 1 is at most $\frac{1}{T'+1}$,
 237 which is only a constant factor away from our analysis. We provide the proof in Appendix C
 238 in the supplementary.

239 3.2 A memory-efficient streaming algorithm with bounded competitive ratio

240 As shown in the previous section, to achieve a large enough competitive ratio with STORM,
 241 the system must have a good enough estimate of the number of user visits T . However,
 242 in practice, this information is typically unavailable in advance. To address this issue,

we discretize the range $[T']$ to generate multiple guesses for the value of T , and run a separate instance of Algorithm 1 for each guess. When a user visits the system, each copy of Algorithm 1 outputs a solution, and we select the best solution over all the outputs.

Specifically, Algorithm 2 (named STORM++) first initializes the set of guesses of T as $\mathcal{P} = \{\delta i \mid i \in \lceil [T'/\delta] \rceil\}$, where $\delta \in \mathbb{N}$ is a parameter controlling the trade-off between competitive ratio as well as the space and time complexity. In this construction, there always exists a guess $\tilde{T} \in \mathcal{P}$ that is between T and $T + \delta$. Therefore, if we would run Algorithm 1 with \tilde{T} as the input parameter, and present the results to the user, we are guaranteed to have a competitive ratio of at least $1/4\delta$ as shown in Theorem 3.1; however, we do not know the value of \tilde{T} in advance. Thus, we adopt a greedy strategy to aggregate the results from all the copies of Algorithm 1 to determine an output for each user visit. At the i -th user visit, we collect the i -th output from each running copy of STORM with a different guess $\tilde{T} \in \mathcal{P}$ and select the one that maximizes the incremental gain. This selection rule leads to the following result.

Theorem 3.2. *Algorithm 2 has a competitive ratio of $1/8\delta$, space complexity $\mathcal{O}(T'^2 k/\delta)$, and time complexity $\mathcal{O}(NkT'^2/\delta)$.*

4 Empirical evaluation

We empirically evaluate the performance of our proposed algorithms STORM and STORM++. We denote STORM(T) and STORM(T') as the STORM algorithm with the input parameter set to T (i.e., exact number of user accesses) and T' (i.e., upper bound of user accesses), resp.

We design our experiments to answer the following research questions: **(Q1)** How effective are the solutions provided by STORM++ and STORM(T') compared to other baselines and to each other? **(Q2)** What is the impact of parameters δ and k on the performance of STORM++ and STORM(T')? **(Q3)** How sensitive are STORM++ and STORM(T') to the upper bound T' and do our algorithms require highly-accurate estimation of T' to perform well?

We use the following **baselines**:

- **LMGREEDY**: Our upper-bound baseline, which assumes linear memory size in the stream length, i.e., potentially unbounded memory (Algorithm 3).
- **SIEVE++**: Based on Kazemi et al. [23], we implement a heuristic version of the SIEVE++ algorithm to solve the *S3MOR* problem. The adaption proceeds as follows: (1) Before the first user visit, we run the exact SIEVE++ algorithm and obtain output set \mathcal{S}^1 . (2) For each subsequent i -th output, we run SIEVE++ on the data stream between the $(i-1)$ -th and i -th user visits, using the marginal gain function $f(\cdot \mid \bigcup_{j=1}^{i-1} \mathcal{S}^j)$ as the objective.
- **PREEMPTION**: We adapt the algorithm proposed by Buchbinder et al. [5] for solving the *S3MOR* problem in the same way as described above for SIEVE++.

Datasets: We use six real-world datasets, including four rating datasets KuaiRec, Anime, Beer and Yahoo, and two multi-label dataset RCV1 and Amazon. The number of items these dataset contains vary from thousands to one million, and the topic sets size vary from around 40 to around 4000. We provide the details in Appendix D.1 in the supplementary.

Experimental setting: To simulate the news item stream, we randomly shuffle the items to form a stream of length N . To simulate the user visit sequence, we start with an all-zeros sequence and then randomly select T indices, and set those positions to one. We establish an upper bound on the number of visits by setting $T' = T + \Delta T$ for some positive ΔT . We generate synthetic click probabilities, and provide the details in Appendix D.1.

In all experiments, we randomly selected 50 users and report the average expected coverage for the Yahoo, RCV1, and Amazon datasets. Standard deviations and expected coverage results for the other three datasets are provided in Appendix D in the supplementary. We also report runtime and memory usage on the largest dataset, Amazon.

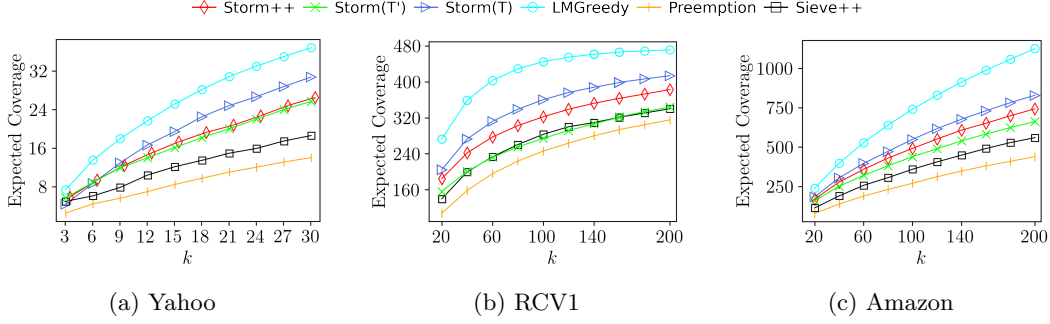


Figure 2: Empirical variation in expected coverage as a function of budget k on all datasets. Parameter $T = 5$, $\delta = 25$ and $\Delta T = 45$ are fixed

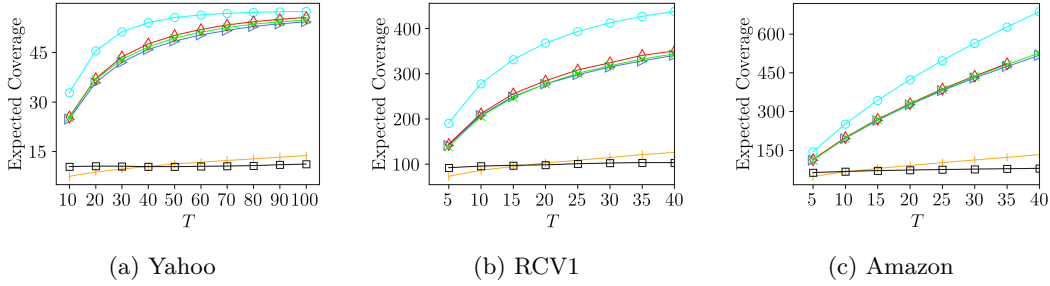


Figure 3: Empirical variation in expected coverage as a function of number of visit T on all datasets. Parameter $k = 10$, $\delta = 10$ and $\Delta T = 10$ are fixed.

291 All experiments ran on a MacBook Air (Model Mac15,12) equipped with an Apple M3 chip
 292 (8-core: 4 performance and 4 efficiency cores), 16 GB of memory, and a solid-state drive
 293 (SSD). The source code and datasets are available.¹

294 Results and discussion

295 **Impact of k and T on the expected coverage.** As shown in Figure 2, LMGREEDY
 296 achieves the best performance across all datasets and values of k . Among our proposed
 297 methods, STORM(T) consistently performs the best, followed by STORM++, then STORM(T').
 298 Compared to baselines SIEVE++ and PREEMPTION, our algorithms generally perform better
 299 across datasets and values of k , with exceptions on RCV1, where PREEMPTION performs
 300 comparably to STORM(T'). In Figure 3, we observe a trend shift: STORM++ slightly outper-
 301 forms STORM(T') and STORM(T) across datasets and T values. STORM(T') and STORM(T)
 302 perform comparably. This performance shift is due to smaller δ and ΔT in this experi-
 303 ment, which lead to improved competitive ratios for STORM(T') and STORM++. Although
 304 STORM(T) offers the strongest theoretical guarantee of the three algorithms, STORM(T') and
 305 STORM++ can empirically outperform it (we give an example of this effect in Appendix D.2).

306 **Performance regarding parameters δ and ΔT .** Figure 4a shows that STORM++'s
 307 expected coverage decreases as δ increases, consistent with theory: as δ grows from 3 to
 308 30, the performance guarantee drops by a factor of 10. This is reflected empirically with
 309 coverage declines across datasets, for example, from 130 to 120 on Amazon.

310 Figure 4b shows that STORM(T')'s performance declines with increasing ΔT , matching the
 311 theoretical drop in competitive ratio (e.g., 10-fold decrease from $\Delta T = 10$ to 100). In
 312 contrast, STORM++ is theoretically unaffected.

313 **Number of oracle calls, runtime and memory.** Figure 4c shows that the number of
 314 oracle calls increases rapidly and linearly with budget k for both STORM(T') and STORM++,

¹<https://anonymous.4open.science/r/Streaming-Submodular-maximization-with-on-demand-user-request-0095>

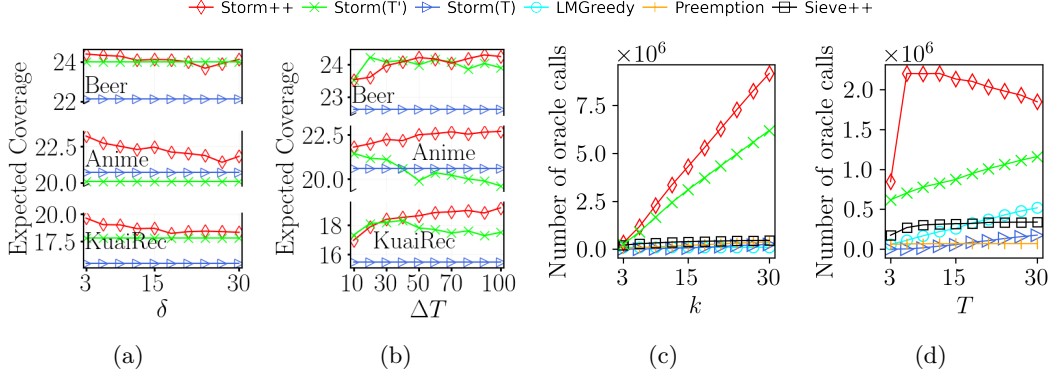


Figure 4: Left two: Impact of ΔT and δ on expected coverage. In (a) and (b), we fix $k = 5$ and $T = 10$. When varying δ , we fix $\Delta T = 60$; when varying ΔT , we fix $\delta = 10$. Right two: Oracle call counts on the Beer dataset. We fix $\delta = 25$ and $\Delta T = 45$. When varying k , we fix $T = 5$; when varying T , we fix $k = 5$.

Table 2: Runtime and mem. usage for the Amazon dataset. We set k, T , and δ to 10, $T' = 20$.

	STORM++	STORM(T')	STORM(T)	LMGREEDY	SIEVE++	PREEMPTION
Runtime (s)	35.13	7.94	4.38	88.13	307.43	136.04
Memory (MB)	2.00	1.28	0.72	28.99	0.91	0.03

while remaining relatively stable for other algorithms. Similarly, Figure 4d indicates a linear increase in oracle calls with user visits T for STORM(T'), STORM(T), and LMGREEDY, but a much slower growth for SIEVE++ and PREEMPTION. Notably, Figure 4d also reveals a non-monotonic trend: oracle calls first rise, then decline with increasing T . This increase is due to the guessing strategy for T under $\delta = 25$ and $\Delta T' = 45$, where the number of guesses \mathcal{T} increases ($\mathcal{T} = \{25, 50\}$ for $T = 3$ and $\mathcal{T} = \{25, 50, 75\}$ for $T \in [6, 30]$). The decrease is due to STORM++’s irrevocability—as more subsets are presented to the user, fewer items remain in \mathcal{S} for potential swap, thereby reducing the number of subsequent oracle calls.

Table 2 shows that our proposed algorithms require much less memory than LMGREEDY, and use comparable memory to the other two baselines. STORM(T') and STORM(T) are also significantly faster than the remaining algorithms.

5 Conclusion

We investigate a novel online setting for streaming submodular maximization where users submit on-demand requests throughout the stream. In this context, we introduce the *Streaming Stochastic Submodular Maximization with On-demand Requests (S3MOR)* problem with the goal of maximizing the expected coverage of the selected items. We first propose a memory-efficient approximation algorithm that highlights the trade-off between memory usage and solution quality, but its competitive ratio can be poor. To solve this, we propose a parameter-dependent memory-efficient approximation algorithm to trade off between the competitive ratio and the memory and time usage. Our empirical evaluations show that our proposed algorithms consistently outperform the baselines. We believe our approach will have a positive social impact. For instance, it can be applied to enhance the diversity of news content recommended to the users, thereby helping to expose them to multiple viewpoints and broaden their knowledge. While our approach optimizes topic coverage and can be used to promote content diversity, its misuse, such as deliberately optimizing a submodular utility function that favors polarized or biased content, could risk amplifying existing biases. However, such outcomes would require intentional manipulation of the objective function. When applied responsibly, our approach does not pose such risks.

References

- [1] Saeed Alaei, Ali Makhdoumi, and Azarakhsh Malekian. 2021. Maximizing sequence-submodular functions and its application to online advertising. *Management Science* 67, 10 (2021), 6030–6054.
- [2] Bushra Alhijawi, Arafat Awajan, and Salam Fraihat. 2022. Survey on the objectives of recommender systems: Measures, solutions, evaluation methodology, and new perspectives. *Comput. Surveys* 55, 5 (2022), 1–38.
- [3] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal Greedy Diversity for Recommendation.. In *IJCAI*, Vol. 15. 1742–1748.
- [4] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 671–680.
- [5] Niv Buchbinder, Moran Feldman, and Roy Schwartz. 2019. Online submodular maximization with preemption. *ACM Transactions on Algorithms (TALG)* 15, 3 (2019), 1–31.
- [6] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.
- [7] Amit Chakrabarti and Sagar Kale. 2015. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming* 154 (2015), 225–247.
- [8] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. 2015. Streaming algorithms for submodular function maximization. In *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I* 42. Springer, 318–330.
- [9] Marwa El Halabi, Federico Fusco, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. 2023. Fairness in streaming submodular maximization over a matroid constraint. In *International Conference on Machine Learning*. PMLR, 9150–9171.
- [10] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. 2018. Do less, get more: Streaming submodular maximization with subsampling. *Advances in Neural Information Processing Systems* 31 (2018).
- [11] Moran Feldman, Paul Liu, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. 2021. Streaming submodular maximization under matroid constraints. *arXiv preprint arXiv:2107.07183* (2021).
- [12] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. 1978. *An analysis of approximations for maximizing submodular set functionsII*. Springer.
- [13] Satoru Fujishige. 2005. *Submodular functions and optimization*. Vol. 58. Elsevier.
- [14] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiabin Mao, and Tat-Seng Chua. 2022. KuaiRec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 540–550.
- [15] Michel Goemans and Jan Vondrák. 2006. Stochastic covering and adaptivity. In *Latin American symposium on theoretical informatics*. Springer, 532–543.
- [16] Daniel Golovin and Andreas Krause. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research* 42 (2011), 427–486.
- [17] Ryan Gomes and Andreas Krause. 2010. Budgeted Nonparametric Learning from Data Streams.. In *ICML*, Vol. 1. Citeseer, 3.

- [18] Alkis Gotovos, Amin Karbasi, and Andreas Krause. 2015. Non-Monotone Adaptive Submodular Maximization. In *IJCAI*. 1996–2003.
- [19] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. 2008. Set covering with our eyes closed. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 347–356.
- [20] Ran Haba, Ehsan Kazemi, Moran Feldman, and Amin Karbasi. 2020. Streaming submodular maximization under a k-set system constraint. In *International Conference on Machine Learning*. PMLR, 3939–3949.
- [21] Chien-Chung Huang, Naonori Kakimura, and Yuichi Yoshida. 2020. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica* 82, 4 (2020), 1006–1032.
- [22] Chien-Chung Huang, Theophile Thiery, and Justin Ward. 2021. Improved multi-pass streaming algorithms for submodular maximization with matroid constraints. *arXiv preprint arXiv:2102.09679* (2021).
- [23] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. 2019. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning*. PMLR, 3311–3320.
- [24] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 137–146.
- [25] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
- [26] Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability* 3, 71-104 (2014), 3.
- [27] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 420–429.
- [28] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5, Apr (2004), 361–397.
- [29] Yajing Liu, Edwin KP Chong, Ali Pezeshki, and Zhenliang Zhang. 2020. Submodular optimization problems and greedy strategies: A survey. *Discrete Event Dynamic Systems* 30, 3 (2020), 381–412.
- [30] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. 165–172.
- [31] Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 1020–1025.
- [32] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*. 897–908.
- [33] Anay Mehrotra and Nisheeth K Vishnoi. 2023. Maximizing submodular functions for recommendation in the presence of biases. In *Proceedings of the ACM Web Conference 2023*. 3625–3636.

- 438 [34] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. 2016. Fast
439 constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning*. PMLR, 1358–1367.
- 441 [35] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and
442 Andreas Krause. 2015. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.
- 444 [36] Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. 2018. Streaming non-
445 monotone submodular maximization: Personalized video summarization on the fly. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- 447 [37] Marko Mitrovic, Ehsan Kazemi, Moran Feldman, Andreas Krause, and Amin Karbasi.
448 2019. Adaptive sequence submodularity. *Advances in Neural Information Processing Systems* 32 (2019).
- 450 [38] Houssam Nassif, Kemal Oral Cansizlar, Mitchell Goodman, and SVN Vishwanathan.
451 2018. Diversifying music recommendations. *arXiv preprint arXiv:1810.01482* (2018).
- 452 [39] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of
453 approximations for maximizing submodular set functionsI. *Mathematical programming* 14 (1978), 265–294.
- 455 [40] Naoto Ohsaka and Yuichi Yoshida. 2015. Monotone k-submodular function maximiza-
456 tion with size constraints. *Advances in Neural Information Processing Systems* 28
457 (2015).
- 458 [41] Shaojie Tang and Jing Yuan. 2020. Adaptive cascade submodular maximization. *arXiv preprint arXiv:2007.03592* (2020).
- 460 [42] Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection
461 and active learning. In *International conference on machine learning*. PMLR, 1954–
462 1963.
- 463 [43] Qiong Wu, Yong Liu, Chunyan Miao, Yin Zhao, Lu Guan, and Haihong Tang. 2019. Re-
464 cent advances in diversified recommendation. *arXiv preprint arXiv:1905.06589* (2019).
- 465 [44] Yisong Yue and Carlos Guestrin. 2011. Linear submodular bandits and their application
466 to diversified retrieval. *Advances in Neural Information Processing Systems* 24 (2011).
- 467 [45] Ghahramani Zoubin. 2013. Scaling the indian buffet process via submodular maximiza-
468 tion. In *International Conference on Machine Learning*. PMLR, 1013–1021.

469 A Preliminaries

470 We use $[j]$, with $j \in \mathbb{N}$, to denote the set $\{1, 2, \dots, j\}$. A *set system* is a pair (U, \mathcal{I}) , where U
 471 is a ground set and \mathcal{I} is a family of subsets of U .

472 **Definition A.1** (Matroid). *A set system (U, \mathcal{I}) is a matroid if*

- 473 (i) $\emptyset \in \mathcal{I}$,
- 474 (ii) for $B \in \mathcal{I}$ and any $A \subseteq B$ we have $A \in \mathcal{I}$, and
- 475 (iii) if $A \in \mathcal{I}$, $B \in \mathcal{I}$ and $|A| < |B|$, then there exists $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

476 **Definition A.2** (Partition matroid). *A matroid $\mathcal{M} = (U, \mathcal{I})$ is a partition matroid if there*
 477 *exists a partition $\{E_i\}_{i=1}^r$ of U , i.e., $U = \cup_{i=1}^r E_i$ and $E_i \cap E_j = \emptyset$ for all $i \neq j$, and*
 478 *non-negative integers k_1, \dots, k_r , such that $\mathcal{I} = \{I \subseteq U : |I \cap E_i| \leq k_i \text{ for all } i \in [r]\}$.*

479 A non-negative set function $f: 2^U \rightarrow \mathbb{R}_{\geq 0}$ is called *submodular* if for all sets $A \subseteq B \subseteq U$ and
 480 all elements $e \in U \setminus B$, it is $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. The function f is *monotone*
 481 if for all $A \subseteq B$ it is $f(A) \leq f(B)$. Let U be a finite ground set, and let $f: 2^U \rightarrow \mathbb{R}_{\geq 0}$ be
 482 a *monotone submodular* set function. The *constrained submodular maximization problem* is
 483 to find $\max_{S \in \mathcal{I}} f(S)$, where $\mathcal{I} \subseteq 2^U$ represents the feasibility constraints.

484 For ease of notation, for any two sets $A \subseteq U$ and $B \subseteq U$, and any item $e \in U$, we write
 485 $A + B$ for $A \cup B$, $A + e$ for $A \cup \{e\}$, $A - B$ for $A \setminus B$, and $A - e$ for $A \setminus \{e\}$. We write
 486 $f(A \mid B)$ and $f(e \mid B)$ for the incremental gain respectively defined as $f(A \cup B) - f(B)$ and
 487 $f(A \cup \{e\}) - f(A)$.

488 We evaluate the performance of online streaming algorithms that make irrevocable decisions
 489 using the notion of *competitive ratio*. An algorithm is called *c-competitive* if, for every input
 490 stream σ , the value $\mathcal{A}(\sigma)$ achieved by algorithm \mathcal{A} satisfies $\frac{\mathcal{A}(\sigma)}{\mathbf{OPT}(\sigma)} \geq c$, where $\mathbf{OPT}(\sigma)$
 491 denotes the value of an optimal offline solution with full access to the input stream.

492 B Omitted proofs of Section 2

493 **Lemma 2.1.** *$f(\mathcal{S})$ is a non-decreasing submodular set function.*

494 *Proof.* For any set \mathcal{S} , we can rewrite Equation (1). Denote $\mathcal{S} = \cup_{t=1}^T \mathcal{S}^t$ and we allow
 495 multiple copies of V_i to appear in \mathcal{S} (the copies come from each \mathcal{S}^t). Note that \mathcal{S} is a
 496 multi-set, and

$$f(\mathcal{S}) = \sum_{j=1}^d \left(1 - \prod_{t \in [T]} \prod_{V_i \in \mathcal{S}^t, V_i \ni c_j} (1 - p_i) \right) = \sum_{j=1}^d \left(1 - \prod_{V_i \in \mathcal{S}, V_i \ni c_j} (1 - p_i) \right).$$

497 Let $g_j(\mathcal{S}) = 1 - \prod_{V_i \in \mathcal{S}, V_i \ni c_j} (1 - p_i)$. Then $f(\mathcal{S}) = \sum_{j=1}^d g_j(\mathcal{S})$. To prove the lemma, it
 498 suffices to show that g_j is non-decreasing and submodular for all $j \in [d]$.

499 **Non-decreasing property:** For any two sets $\mathcal{X} \subseteq \mathcal{Y}$ and for all $j \in [d]$, it is

$$g_j(\mathcal{Y}) - g_j(\mathcal{X}) = \left(1 - \prod_{V_i \in \mathcal{Y} \setminus \mathcal{X}, V_i \ni c_j} (1 - p_i) \right) \prod_{V_i \in \mathcal{X}, V_i \ni c_j} (1 - p_i) \geq 0.$$

Algorithm 3: LMGREEDY

Input: Budget $k \in \mathbb{N}$.

```

1  $\mathcal{G}^0, \dots, \mathcal{G}^T \leftarrow \emptyset, \mathcal{G} \leftarrow \emptyset, \mathcal{V}^1 \leftarrow \emptyset, t = 1$ 
2 for  $(V, \tau, p)$  in the stream do
3    $\mathcal{V}^t \leftarrow \mathcal{V}^t \cup \{V\}$ 
4   if  $\tau = 1$  then
5      $\mathcal{N}^t \leftarrow \mathcal{V}^t$ 
6     for  $j = 1$  to  $k$  do
7        $V^* \leftarrow \arg \max_{V \in \mathcal{N}^t} f(\mathcal{G} \cup \mathcal{G}^t \cup \{V\})$ 
8        $\mathcal{G}^t \leftarrow \mathcal{G}^t \cup \{V^*\}$ 
9        $\mathcal{N}^t \leftarrow \mathcal{N}^t \setminus V^*$ 
10    output  $\mathcal{G}^t$ 
11     $\mathcal{G} = \mathcal{G} \cup \mathcal{G}^t$ 
12     $\mathcal{V}^{t+1} \leftarrow \mathcal{V}^t$ 
13     $t \leftarrow t + 1$ 

```

Submodularity: Let $\mathcal{X} \subseteq \mathcal{Y}$, and let $Z \in \overline{\mathcal{Y}}$ be an element not in \mathcal{Y} . Then, for all $j \in [d]$, the following holds:

$$\begin{aligned}
& g_j(\mathcal{Y} \cup \{Z\}) - g_j(\mathcal{Y}) - [g_j(\mathcal{X} \cup \{Z\}) - g_j(\mathcal{X})] \\
&= \left(1 - \prod_{V_i \in \{Z\}, V_i \ni c_j} (1 - p_i)\right) \prod_{V_i \in \mathcal{Y}, V_i \ni c_j} (1 - p_i) - \left(1 - \prod_{V_i \in \{Z\}, V_i \ni c_j} (1 - p_i)\right) \prod_{V_i \in \mathcal{X}, V_i \ni c_j} (1 - p_i) \\
&= \left(1 - \prod_{V_i \in \{Z\}, V_i \ni c_j} (1 - p_i)\right) \left(\prod_{V_i \in \mathcal{Y} \setminus \mathcal{X}, V_i \ni c_j} (1 - p_i) - 1\right) \prod_{V_i \in \mathcal{X}, V_i \ni c_j} (1 - p_i) \leq 0.
\end{aligned}$$

502

□

Theorem 2.4. *Algorithm 3 for Problem 2.2 is $1/2$ -competitive. Moreover, the competitive ratio of $1/2$ is tight, i.e., for Problem 2.2, no streaming algorithm can achieve a competitive ratio better than $1/2$ without violating the irrevocability constraint.*

Proof. We first prove that Algorithm 3 is $1/2$ competitive for Problem 2.2.

Let $\mathcal{O}^1, \dots, \mathcal{O}^T$ be the optimal output sets, and let $\mathcal{G}^1, \dots, \mathcal{G}^T$ be the output sets obtained by Algorithm 3. Since we treat each copy of the same item as a distinct item, it follows that for any $i \neq j$, $\mathcal{O}^i \cap \mathcal{O}^j = \emptyset$ and $\mathcal{G}^i \cap \mathcal{G}^j = \emptyset$. We let $\mathcal{G} = \bigcup_{t=1}^T \mathcal{G}^t$ and $\mathcal{O} = \bigcup_{t=1}^T \mathcal{O}^t$.

We prove that the following holds:

$$\begin{aligned}
f(\mathcal{O}) &\stackrel{(a)}{\leq} f(\mathcal{G}) + \sum_{V \in \mathcal{O} \setminus \mathcal{G}} f(V \mid \mathcal{G}) \stackrel{(b)}{=} f(\mathcal{G}) + \sum_{i=1}^T \sum_{V \in \mathcal{O}^i \setminus \mathcal{G}} f(V \mid \mathcal{G}) \\
&\stackrel{(c)}{\leq} f(\mathcal{G}) + \sum_{t=1}^T \sum_{V \in \mathcal{O}^i} f(V \mid \mathcal{G}) \stackrel{(d)}{\leq} f(\mathcal{G}) + \sum_{t=1}^T \sum_{V \in \mathcal{O}^i} f(V \mid \bigcup_{j=1}^i \mathcal{G}^j).
\end{aligned} \tag{2}$$

Inequalities (a) and (d) holds by submodularity, (c) holds by monotonicity, and equality (b) holds because $\{\mathcal{O}^1, \dots, \mathcal{O}^T\}$ form a partition of \mathcal{O} .

Next, we bound the final term of Equation (2). We denote $\mathcal{O}^i = \{O_1^i, \dots, O_k^i\}$, and $\mathcal{G}^i = \{G_1^i, \dots, G_k^i\}$, with G_j^i representing the j -th item that is added to \mathcal{G}^i during the greedy selection steps. Note that, we can assume $|\mathcal{G}| = |\mathcal{O}| = k$, because f is monotone. The

516 following holds

$$\begin{aligned}
\sum_{V \in \mathcal{O}^i} f(V \mid \bigcup_{j=1}^i \mathcal{G}^j) &= \sum_{l=1}^k f(\mathcal{O}_l^i \mid \bigcup_{j=1}^i \mathcal{G}^j) \stackrel{(a)}{\leq} \sum_{l=1}^k f(\mathcal{O}_l^i \mid \bigcup_{j=1}^{i-1} \mathcal{G}^j + \bigcup_{s=1}^{l-1} \mathcal{G}_s^i) \\
&\stackrel{(b)}{\leq} \sum_{l=1}^k f(\mathcal{G}_l^i \mid \bigcup_{j=1}^{i-1} \mathcal{G}^j + \bigcup_{s=1}^{l-1} \mathcal{G}_s^i) = f(\bigcup_{j=1}^i \mathcal{G}^j - \bigcup_{j=1}^{i-1} \mathcal{G}^j),
\end{aligned} \tag{3}$$

517 where inequality (a) holds by submodularity, and inequality (b) holds by design of the greedy
518 algorithm (Line 7 of Algorithm 3).

519 Combining Equation (2) and Equation (3) finishes the proof:

$$f(\mathcal{O}) \leq f(\mathcal{G}) + \sum_{t=1}^T f(\bigcup_{j=1}^i \mathcal{G}^j - \bigcup_{j=1}^{i-1} \mathcal{G}^j) = 2f(\mathcal{G}) - f(\mathcal{G}^0) \leq 2f(\mathcal{G}). \tag{4}$$

520 Furthermore, we prove that the competitive ratio of $1/2$ is tight for Problem 2.2.

521 Consider a simple adversarial example with budget $k = 1$ and a stream given by

$$\mathbf{S} = ((V_1, 0, 1), (V_2, 1, 1), (V_3, 1, 1)),$$

522 where $V_1 = \{c_1, c_2\}$ and $V_2 = \{c_3, c_4\}$. For any algorithm, if it selects $\mathcal{S}^1 = \{V_1\}$ as the
523 first set to present, the adversary sets $V_3 = \{c_1, c_2\}$. Conversely, if the algorithm selects
524 $\mathcal{S}^1 = \{V_2\}$, then the adversary sets $V_3 = \{c_3, c_4\}$.

525 In either case, the algorithm can only cover two topics, while the optimal solution covers all
526 four. Thus, the competitive ratio is at most $\frac{2}{4} = \frac{1}{2}$, which completes the proof.

527 □

528 C Omitted proofs of Section 3

529 Before presenting the omitted proofs in Section 3, we provide a detailed description of the
530 reductions used in our analysis.

531 We begin by formally defining the streaming submodular maximization problem subject to
532 a partition matroid constraint (*S2MM*). Our definition adapts the streaming submodular
533 maximization framework under a p -matchoid constraint, as introduced by Chekuri et al. [8],
534 and is stated as follows:

535 **Problem C.1 (*S2MM*).** *Let $\mathcal{M} = (\bar{\mathbf{S}}, \mathcal{I})$ be a partition matroid and \hat{f} a submodular func-*
536 *tion. The elements of $\bar{\mathbf{S}}$ are presented in a stream, and we order $\bar{\mathbf{S}}$ by order of appearance.*
537 *The goal of the S2MM problem is to select a subset of items $\mathcal{A} \in \mathcal{I}$ that maximizes $\hat{f}(\mathcal{A})$.*

538 We demonstrate two types of reductions from an instance of *S3MOR* to an instance of
539 *S2MM*, depending on whether we know the exact value of T . Algorithm 1, as presented in
540 the main content, is based on the second reduction, where we only know an upper bound
541 T' of T .

542 C.1 The case when the exact number of visits T is known

543 First, we show that if the exact number of user visits T is known, the *S3MOR* problem
544 reduces to the *S2MM* problem. Specifically, any streaming algorithm for *S2MM* that satisfies
545 the *irrevocable output requirement* can solve *S3MOR* while maintaining the same competitive
546 ratio. The *irrevocable output requirement* requires that the output subset for any partition
547 becomes fixed irrevocably once all its items have appeared in the stream.

548 **The reduction.** Given an instance of the *S3MOR* problem, where the stream is given by

$$\mathbf{S} = ((V_1, p_1, \tau_1), \dots, (V_N, p_N, \tau_N)),$$

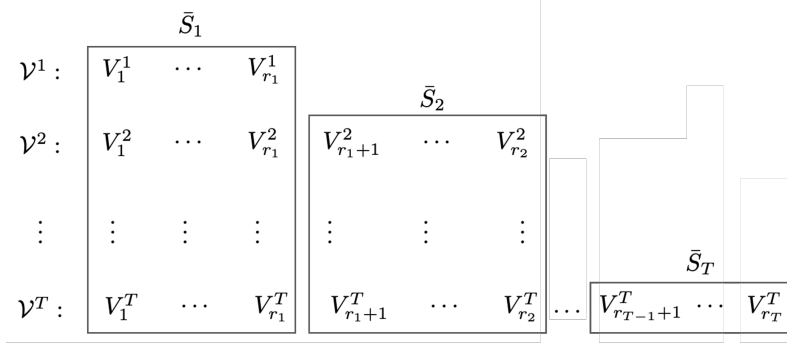


Figure 5: Illustration of the reduction when the exact number of visits T is known. The original item stream for the $S3MOR$ problem is (V_1, \dots, V_{r_T}) , with user accesses occurring at time steps r_1, \dots, r_T . The constructed stream for the $S2MM$ problem is $\bar{\mathbf{S}} = \bigsqcup_{t=1}^T \bar{\mathbf{S}}_t$.

549 with a budget k , and the function f as defined in Equation (1), we can construct a corre-
550 sponding instance of the $S2MM$ problem.

551 Recall that r_t denotes the time step at which the user submits the t -th visit (with $r_0 = 0$), and
552 V_i^t denotes a copy of item V_i with copy identifier t . We use \bigsqcup to denote the concatenation
553 of streams. An instance of the $S2MM$ problem with input stream $\bar{\mathbf{S}}$ can be constructed
554 as follows:

- 555 1. Let $\bar{\mathbf{S}}_t = \bigsqcup_{i=r_{t-1}+1}^{r_t} (V_i^t, \dots, V_i^T)$ for all $t \in [T]$, to be specific,

$$\bar{\mathbf{S}}_t = \left(\underbrace{V_{r_{t-1}+1}^t, \dots, V_{r_{t-1}+1}^T}_{i=r_{t-1}+1}, \underbrace{V_{r_{t-1}+2}^t, \dots, V_{r_{t-1}+2}^T}_{i=r_{t-1}+2}, \dots, \underbrace{V_{r_t}^t, \dots, V_{r_t}^T}_{i=r_t} \right).$$

- 556 2. Let $\bar{\mathbf{S}} = \bigsqcup_{t=1}^T \bar{\mathbf{S}}_t$ be the concatenation of sub-streams $\bar{\mathbf{S}}_1, \dots, \bar{\mathbf{S}}_T$, i.e., $\bar{\mathbf{S}} =$
557 $(\bar{\mathbf{S}}_1, \dots, \bar{\mathbf{S}}_T)$.

- 558 3. Let $\mathcal{V}^t = \{V_i^t\}_{i=1}^{r_t} = \{V_1^t, \dots, V_{r_t}^t\}$.

- 559 4. Construct \mathcal{I} in the following way: for any set $X \subseteq \bigcup_{t=1}^T \mathcal{V}^t$, if $|X \cap \mathcal{V}^t| \leq k$ for all
560 $1 \leq t \leq T$, then $X \in \mathcal{I}$.

- 561 5. Let $p_i^t = p_i$ denote a copy of p_i with copy identifier $t \in [T]$. Define $\hat{f} : 2^{\bigcup_{t=1}^T \mathcal{V}^t} \rightarrow$
562 $\mathbb{R}_{\geq 0}$. Specifically, for any $\mathcal{A} \subseteq \bigcup_{t=1}^T \mathcal{V}^t$, let $\mathcal{A}^t = \mathcal{A} \cap \mathcal{V}^t$. Then, \hat{f} is defined as
563 follows:

$$\hat{f}(\mathcal{A}) = \sum_{j=1}^d \left(1 - \prod_{t \in [T]} \prod_{V_i^t \in \mathcal{A}^t, V_i^t \ni c_j} (1 - p_i^t) \right), \quad (5)$$

564 and the goal of $S2MM$ is to find $\mathcal{A} \in \mathcal{I}$ that maximizes $\hat{f}(\mathcal{A})$.

565 We observe that $\mathcal{M} = (\bar{\mathbf{S}}, \mathcal{I})$, as defined above, forms a partition matroid on the constructed
566 stream $\bar{\mathbf{S}}$, with $\bar{\mathbf{S}} = \bigcup_{t=1}^T \mathcal{V}^t$ and $\mathcal{V}^i \cap_{i \neq j} \mathcal{V}^j = \emptyset$. By construction, for each visit $t \in [T]$, the
567 partition \mathcal{V}^t contains exactly the set of items that arrived in the system before the t -th user
568 visit in the $S3MOR$ problem. Since the $S2MM$ problem and the $S3MOR$ problem share
569 equivalent objective functions and equivalent constraints, we conclude the following: for any
570 optimal solution $\bar{\mathcal{O}} \in \mathcal{I}$ of the $S2MM$ problem, the sequence of subsets

$$\bar{\mathcal{O}} \cap \mathcal{V}^1, \dots, \bar{\mathcal{O}} \cap \mathcal{V}^T$$

571 yields optimal output for the $S3MOR$ problem at each corresponding t -th user visit.

We can further conclude that any streaming algorithm achieving an α -competitive solution for the $S2MM$ problem also yields an α -competitive solution for the $S3MOR$ problem. For this equivalence to hold, the streaming algorithm for $S2MM$ must satisfy the *irrevocable output requirement*: for each partition \mathcal{V}^t of the stream, once the final item from this partition has arrived, the solution that belongs to this partition, $\mathcal{A} \cap \mathcal{V}^t$, can no longer be changed. This requirement excludes algorithms like that of Feldman et al. [11], whose solution set only becomes available after the stream terminates. In contrast, the algorithms proposed by Chakrabarti and Kale [7], Chekuri et al. [8], and Feldman et al. [10] are suitable. These three algorithms all meet the requirement while providing $1/4$ -competitive solutions for the $S2MM$ problem.

C.2 The case when only an upper bound T' of T is available

Consider the case where only an upper bound T' of the true number of user visits T is known. We prove the following: (1) An instance of the $S2MM$ problem can be constructed such that any feasible solution to the $S2MM$ problem contains a valid solution for the $S3MOR$ problem. (2) Any streaming algorithm that solves the $S2MM$ problem and satisfies the *irrevocable output requirement* can be adapted to solve the $S3MOR$ problem, albeit with a reduced competitive ratio. Specifically, the competitive ratio decays by a factor of $1/(T' - T + 1)$.

In contrast to the reduction we present in Appendix C.1, the construction of the stream for the $S2MM$ problem depends on both the original stream \mathbf{S} , and the algorithmic procedure described in Algorithm 1. We will prove that:

1. Algorithm 1 generates a corresponding stream $\hat{\mathbf{S}}$ for the $S2MM$ problem
2. The candidate sets $\{\mathcal{A}^1, \dots, \mathcal{A}^{T'}\}$ maintained by Algorithm 1 forms a solution to the $S2MM$ problem; the output $\{\mathcal{G}^1, \dots, \mathcal{G}^T\}$ of Algorithm 1 forms a solution to the $S3MOR$ problem.

By Algorithm 1 line 11, we can construct a bijective mapping π between the index sets of $\{\mathcal{G}^1, \dots, \mathcal{G}^T\}$ and $\{\mathcal{A}^1, \dots, \mathcal{A}^{T'}\}$. This mapping satisfies the equality:

$$\mathcal{G}^t = \mathcal{A}^{\pi(t)}, \quad \text{for all } t \in [T].$$

Let $\mathcal{T}^0 = \{1, 2, \dots, T'\}$ denote the initial index set of all active candidate sets. For each $t \geq 1$, let \mathcal{T}^t represent the index set of active candidate sets before the t -th visit and after the $(t-1)$ -th visit. The active index sets evolve as follows:

- (1) For all $t \in [T]$: $\mathcal{T}^t = \mathcal{T}^{t-1} \setminus \{\pi(t)\}$, and
- (2) for all $t \in \{T+1, \dots, T'\}$: $\mathcal{T}^t = \mathcal{T}^T$.

The reduction.

1. We construct the stream $\hat{\mathbf{S}}$ for $S2MM$ as follows

$$\hat{\mathbf{S}}_t = \begin{cases} \bigsqcup_{i=r_{t-1}+1}^{r_t} \bigsqcup_{a \in \mathcal{T}^t} (V_i^a), & \text{for } t \in [T], \\ \bigsqcup_{i=r_{T-1}+1}^{r_{T'}} \bigsqcup_{a \in \mathcal{T}^T} (V_i^a), & \text{for } t \in \{T+1, \dots, T'\}. \end{cases}$$

2. Let $\hat{\mathbf{S}} = \bigsqcup_{t=1}^T \hat{\mathbf{S}}_t$ be the concatenation of $\hat{\mathbf{S}}_1, \dots, \hat{\mathbf{S}}_T$.

3. Let $\rho(t) = \pi^{-1}(t)$, and

$$\hat{\mathcal{V}}^t = \begin{cases} \{V_i^t\}_{i=1}^{r_{\rho(t)}} = \{V_1^t, \dots, V_{r_{\rho(t)}}^t\}, & \text{for } t \in \{\pi(1), \dots, \pi(T)\}, \\ \{V_1^t, \dots, V_{r_T}^t\}, & \text{for } t \in \mathcal{T}^0 \setminus \{\pi(1), \dots, \pi(T)\}. \end{cases}$$

4. Construct $\hat{\mathcal{I}}$ in the following way: for any set $X \subseteq \bigcup_{t=1}^{T'} \hat{\mathcal{V}}^t$, if $|X \cap \hat{\mathcal{V}}^t| \leq k$ for all $1 \leq t \leq T'$, then $X \in \hat{\mathcal{I}}$.

	$\hat{\mathbf{S}}_1$	$\hat{\mathbf{S}}_2$	$\hat{\mathbf{S}}_3$
$\mathcal{G}^2 \subseteq \hat{\mathcal{V}}^1 :$	$V_1^1 \quad \cdots \quad V_{r_1}^1$	$V_{r_1+1}^1 \quad \cdots \quad V_{r_2}^1$	
$\mathcal{G}^3 \subseteq \hat{\mathcal{V}}^2 :$	$V_1^2 \quad \cdots \quad V_{r_1}^2$	$V_{r_1+1}^2 \quad \cdots \quad V_{r_2}^2$	$V_{r_2+1}^2 \quad \cdots \quad V_{r_3}^2$
$\hat{\mathcal{V}}^3 :$	$V_1^3 \quad \cdots \quad V_{r_1}^3$	$V_{r_1+1}^3 \quad \cdots \quad V_{r_2}^3$	$V_{r_2+1}^3 \quad \cdots \quad V_{r_3}^3$
$\mathcal{G}^1 \subseteq \hat{\mathcal{V}}^4 :$	$V_1^4 \quad \cdots \quad V_{r_1}^4$		
$\hat{\mathcal{V}}^5 :$	$V_1^5 \quad \cdots \quad V_{r_1}^5$	$V_{r_1+1}^5 \quad \cdots \quad V_{r_2}^5$	$V_{r_2+1}^5 \quad \cdots \quad V_{r_3}^5$

Figure 6: An example illustrating the reduction when only an upper bound T' of the true number of visits T is available. In this example, $T = 3$ and $T' = 5$. The original item stream for the *S3MOR* problem is given by $(V_1, V_2, \dots, V_{r_3})$, with user visits occurring at time steps r_1, r_2 , and r_3 . The constructed stream for the *S2MM* problem is $\hat{\mathbf{S}} = \bigsqcup_{t=1}^3 \hat{\mathbf{S}}_t$, where $\hat{\mathbf{S}}_1 = \bigsqcup_{i=1}^{r_1} \bigsqcup_{a \in [5]} (V_i^a)$, $\hat{\mathbf{S}}_2 = \bigsqcup_{i=r_1+1}^{r_2} \bigsqcup_{a \in \{1,2,3,5\}} (V_i^a)$, and $\hat{\mathbf{S}}_3 = \bigsqcup_{i=r_2+1}^{r_3} \bigsqcup_{a \in \{2,3,5\}} (V_i^a)$. Algorithm 1 maintains and updates $\mathcal{A}^t \in \hat{\mathcal{V}}^t$ for all $t \in [5]$ upon the arrival of each item in $\hat{\mathbf{S}}$. At the visit time steps r_1, r_2 , and r_3 , it outputs $\mathcal{G}^1 = \mathcal{A}^4$, $\mathcal{G}^2 = \mathcal{A}^1$, and $\mathcal{G}^3 = \mathcal{A}^2$, respectively. By the end of the stream, the union $\bigcup_{t=1}^5 \mathcal{A}^t$ constitutes a $1/4$ -competitive solution for the *S2MM* problem on input $\hat{\mathbf{S}}$. The combined solution $\bigcup_{t=1}^3 \mathcal{G}^t$ is a $\frac{1}{4 \times (5-3+1)}$ -competitive solution for the original *S3MOR* problem.

610 5. Let $p_i^t = p_i$ denote a copy of p_i with copy identifier $t \in [T']$. The goal is to select
611 $\mathcal{A} = \bigcup_{t=1}^{T'} \mathcal{A}^t \in \hat{\mathcal{I}}$ that maximizes $f(\mathcal{A})$. The submodular function \hat{f} is given as
612 follows:

$$\hat{f}(\mathcal{A}) = \sum_{j=1}^d \left(1 - \prod_{t \in [T']} \prod_{V_i^t \in \mathcal{A}^t, V_i^t \ni c_j} (1 - p_i^t) \right). \quad (6)$$

613 Given the above construction, we can verify that $\mathcal{M} = (\hat{\mathbf{S}}, \hat{\mathcal{I}})$ is an instance of a partition
614 matroid defined on the constructed stream $\hat{\mathbf{S}}$, with $\hat{\mathbf{S}} = \bigcup_{t=1}^{T'} \hat{\mathcal{V}}^t$ and $\hat{\mathcal{V}}^i \cap_{i \neq j} \hat{\mathcal{V}}^j = \emptyset$. By
615 construction, the partitions $\hat{\mathcal{V}}^{\pi(t)}$ of $\hat{\mathbf{S}}$ satisfy: (1) for each $t \in [T]$, $\hat{\mathcal{V}}^{\pi(t)}$ contains exactly the
616 set of items that arrived before the t -th user visit in the *S3MOR* problem (2) The remaining
617 $T' - T$ partitions each contain the same set of items that arrived before the final user visit.
618 The design ensures that each output \mathcal{G}^t satisfies $\mathcal{G}^t \subseteq \hat{\mathcal{V}}^{\pi(t)}$ and $|\mathcal{G}^t| \leq k$, for all $t \in [T]$.
619 Consequently, the sequence $\mathcal{G}^1, \dots, \mathcal{G}^T$ constitutes feasible output for the *S3MOR* problem.

620 Following Algorithm 1, we conclude that for all $t \in [T']$, we have $\mathcal{A}^t \subseteq \hat{\mathcal{V}}^t$ with $|\mathcal{A}^t| \leq k$.
621 Consequently, the union $\bigcup_{t=1}^{T'} \mathcal{A}^t$ forms a feasible solution to the *S2MM* problem.

622 Since Algorithm 1 essentially applies the streaming algorithm of Chekuri et al. [8] to the
623 constructed stream $\hat{\mathbf{S}}$, the solution $\bigcup_{t=1}^{T'} \mathcal{A}^t$ achieves a $1/4$ -approximation guarantee for the
624 *S2MM* problem.

625 C.3 Omitted proofs

626 **Theorem 3.1.** Let T be the number of user accesses and T' a given upper bound. Alg. 1
627 has competitive ratio $\frac{1}{4(T'-T+1)}$, space complexity $\mathcal{O}(T'k)$, and time complexity $\mathcal{O}(NkT')$.

628 *Proof.* Following the notations and conclusions we have in Appendix C.2, we are ready to
 629 prove Theorem 3.1. Let $\mathcal{O}^1, \dots, \mathcal{O}^T$ be the optimal solution for the *S3MOR* problem, and
 630 let $\hat{\mathcal{O}} = \bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t$, where $\hat{\mathcal{O}}^t = \hat{\mathcal{O}} \cap \hat{\mathcal{V}}^t$, denotes the optimal solution for the *S2MM*. We have

$$\begin{aligned}
 f\left(\bigcup_{t=1}^T \mathcal{G}^t\right) &= f\left(\bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)} + \mathcal{A}^{\pi(T)}\right) = f\left(\mathcal{A}^{\pi(T)} \mid \bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right) + f\left(\bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right) \\
 &\stackrel{(a)}{\geq} \frac{1}{T' - T + 1} \sum_{s \in \mathcal{T} \setminus \bigcup_{t=1}^{T-1} \{\pi(t)\}} f\left(\mathcal{A}^s \mid \bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right) + f\left(\bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right) \\
 &\geq \frac{1}{T' - T + 1} \sum_{s \in \mathcal{T} \setminus \bigcup_{t=1}^{T-1} \{\pi(t)\}} f\left(\mathcal{A}^s \mid \bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right) + \frac{1}{T' - T + 1} f\left(\bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right) \\
 &\stackrel{(b)}{\geq} \frac{1}{T' - T + 1} f\left(\bigcup_{t=1}^{T'} \mathcal{A}^t\right) \stackrel{(c)}{\geq} \frac{1}{4(T' - T + 1)} f\left(\bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t\right) \stackrel{(d)}{\geq} \frac{1}{4(T' - T + 1)} f\left(\bigcup_{t=1}^T \mathcal{O}^t\right),
 \end{aligned} \tag{7}$$

631 where inequality (a) holds because by line 9 of Algorithm 1, that $\mathcal{A}^{\pi(T)}$ is selected with
 632 the largest marginal gain in terms of $f\left(\bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)}\right)$. Inequality (b) holds by submodu-
 633 larity. Inequality (c) holds because, as concluded in Appendix C.2, $\bigcup_{t=1}^{T'} \mathcal{A}^t$ achieves a
 634 $1/4$ -approximation guarantee for the *S2MM* problem. Finally, inequality (d) holds by con-
 635 tradiction: if $f\left(\bigcup_{t=1}^T \mathcal{O}^t\right) > f\left(\bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t\right)$, then there exists a feasible solution for the *S2MM*
 636 problem where for all $t \in [T]$, $\mathcal{A}^t = \mathcal{O}^t$, and regardless of the choice of $\mathcal{A}^{T+1}, \dots, \mathcal{A}^T$, it
 637 holds that $f\left(\bigcup_{t=1}^{T'} \mathcal{A}^t\right) \geq f\left(\bigcup_{t=1}^T \mathcal{O}^t\right) > f\left(\bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t\right)$, which contradicts that $\hat{\mathcal{O}}$ is an opti-
 638 mal solution.

639 **Complexity analysis.** The space complexity is in $\mathcal{O}(T'k)$ because STORM only needs to
 640 maintain T' candidate sets of size k . The time complexity is measured by the number of
 641 oracle calls to measure the f value for any selected subset. Upon the arrival of each item,
 642 STORM makes $T'k$ oracle calls, thus the time complexity is in $\mathcal{O}(NT'k)$.

643 □

644 **Tightness of our results.** Note that the analysis in Equation (7) is tight. In the following,
 645 we show that for any value of T , we can construct a stream such that STORM cannot do
 646 better than $\frac{1}{T' - T + 1}$, which is only a constant factor away from our analysis in Theorem 3.1.

647 To see this, consider an instance of the *S3MOR* problem with budget $k = 1$, the user visits
 648 one time at the end of the stream, and we have an estimated number of user visits $T' > 1$,
 649 the stream is constructed as below:

$$\mathbf{S}_1 = ((\{1\}, 1, 0), (\{2\}, 1, 0), \dots, (\{T'\}, 1, 0), (\{1, \dots, T'\}, 1, 1))$$

650 The $\text{STORM}(T')$ algorithm outputs one of the first T' items and achieves an expected topic
 651 coverage of value 1, while the optimal solution is to select the last item and obtain an
 652 expected coverage of value T' . The competitive ratio in this instance is $\frac{1}{T'}$, while Alg. 1 has
 653 competitive ratio $\frac{1}{4(T' - T + 1)} = \frac{1}{4T'}$.

654 Note that for all possible numbers of user visits T and $T' > T$, we can construct a stream
 655 such that STORM cannot do better than $\frac{1}{T' - T + 1}$. For example, when $T = 2$, we can construct
 656 a sub-stream \mathbf{S}_2 as follows

$$\mathbf{S}_2 = ((\{T' + 1, T' + 2\}, 1, 0), \dots, (\{2T' - 1, 2T'\}, 1, 0), (\{T' + 1, \dots, 3T' - 3\}, 1, 1))$$

657 By concatenating \mathbf{S}_1 and \mathbf{S}_2 , we obtain a new stream where the user visits for 2 times. With
 658 budget $k = 1$, $\text{STORM}(T')$ achieves an expected coverage 3, while the optimal expected
 659 coverage is $3(T' - 1)$, leading to a competitive ratio of $\frac{1}{T' - 1}$, with is again a $1/4$ factor away
 660 from $\frac{1}{4(T' - T + 1)} = \frac{1}{T' - 1}$.

661 **Lemma C.2.** Let T be the number of user accesses and T' be a given upper bound. Algo-
 662 rithm 2 generates a set of guesses of T as $\mathcal{P} = \{\delta i \mid i \in \lceil [T'/\delta] \rceil\}$. Let T^* be the smallest
 663 integer in \mathcal{P} that is larger than or equal to T . Let $\mathcal{G}^1, \dots, \mathcal{G}^T$ be the output of STORM(T^*),
 664 and let $\mathcal{B}^1, \dots, \mathcal{B}^T$ be the output of Algorithm 2, then it holds that

$$f(\bigcup_{t=1}^T \mathcal{B}^t) \geq f(\bigcup_{t=1}^T \mathcal{G}^t) + \sum_{t=1}^{T-1} \left[f(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s) - f(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s) \right]. \quad (8)$$

665 *Proof.* We prove Equation (8) via induction.

666 When $T = 1$, according to line 7 of Algorithm 2 that \mathcal{B}^1 is chosen with the largest marginal
 667 gain, it holds that $f(\mathcal{B}^1) \geq f(\mathcal{G}^1)$.

668 The base case for the induction starts from $T = 2$. It is

$$\begin{aligned} f(\mathcal{B}^1, \mathcal{B}^2) &= f(\mathcal{B}^1) + f(\mathcal{B}^2 \mid \mathcal{B}^1) \stackrel{(a)}{\geq} f(\mathcal{G}^1) + f(\mathcal{G}^2 \mid \mathcal{B}^1) \\ &= f(\mathcal{G}^1 \cup \mathcal{B}^1) - f(\mathcal{B}^1 \mid \mathcal{G}^1) + f(\mathcal{G}^2 \mid \mathcal{B}^1) \\ &= f(\mathcal{B}^1) + f(\mathcal{G}^1 \mid \mathcal{B}^1) - f(\mathcal{B}^1 \mid \mathcal{G}^1) + f(\mathcal{G}^2 \mid \mathcal{B}^1) \\ &\stackrel{(b)}{\geq} f(\mathcal{G}^1 \cup \mathcal{G}^2 \mid \mathcal{B}^1) + f(\mathcal{B}^1) - f(\mathcal{B}^1 \mid \mathcal{G}^1) = f(\mathcal{B}^1, \mathcal{G}^1, \mathcal{G}^2) - f(\mathcal{B}^1 \mid \mathcal{G}^1) \\ &= f(\mathcal{G}^1, \mathcal{G}^2) + f(\mathcal{B}^1 \mid \mathcal{G}^1 \cup \mathcal{G}^2) - f(\mathcal{B}^1 \mid \mathcal{G}^1), \end{aligned}$$

669 where inequality (a) holds because $f(\mathcal{B}^1) \geq f(\mathcal{G}^1)$ and $f(\mathcal{B}^2 \mid \mathcal{B}^1) > f(\mathcal{G}^2 \mid \mathcal{B}^1)$. Inequal-
 670 ity (b) holds by submodularity.

671 We then prove the induction step. Assume for $T = j$, it holds

$$f(\bigcup_{t=1}^j \mathcal{B}^t) \geq f(\bigcup_{t=1}^j \mathcal{G}^t) + \sum_{t=1}^{j-1} \left[f(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s) - f(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s) \right]. \quad (9)$$

672 We can use ϕ_j to represent the second term in Equation (9), i.e., $f(\bigcup_{t=1}^j \mathcal{B}^t) \geq f(\bigcup_{t=1}^j \mathcal{G}^t) +$
 673 ϕ_j , we can then show that for $T = j + 1$ it holds

$$\begin{aligned} f(\bigcup_{t=1}^{j+1} \mathcal{B}^t) &= f(\mathcal{B}^{j+1} \mid \bigcup_{t=1}^j \mathcal{B}^t) + f(\bigcup_{t=1}^j \mathcal{B}^t) \stackrel{(a)}{\geq} f(\mathcal{G}^{j+1} \mid \bigcup_{t=1}^j \mathcal{B}^t) + f(\bigcup_{t=1}^j \mathcal{G}^t) + \phi_j \\ &= f(\bigcup_{t=1}^j \mathcal{G}^j \mid \bigcup_{t=1}^j \mathcal{B}^t) + f(\bigcup_{t=1}^j \mathcal{B}^t) - f(\bigcup_{t=1}^j \mathcal{B}^t \mid \bigcup_{t=1}^j \mathcal{G}^t) + f(\mathcal{G}^{j+1} \mid \bigcup_{t=1}^j \mathcal{B}^t) + \phi_j \\ &\stackrel{(b)}{\geq} f(\bigcup_{t=1}^{j+1} \mathcal{G}^t \mid \bigcup_{t=1}^j \mathcal{B}^t) + f(\bigcup_{t=1}^j \mathcal{B}^t) - f(\bigcup_{t=1}^j \mathcal{B}^t \mid \bigcup_{t=1}^j \mathcal{G}^t) + \phi_j \\ &= f(\bigcup_{t=1}^{j+1} \mathcal{G}^t) + f(\bigcup_{t=1}^j \mathcal{B}^t \mid \bigcup_{t=1}^{j+1} \mathcal{G}^t) - f(\bigcup_{t=1}^j \mathcal{B}^t \mid \bigcup_{t=1}^j \mathcal{G}^t) + \phi_j \\ &= f(\bigcup_{t=1}^{j+1} \mathcal{G}^t) + \sum_{t=1}^j \left[f(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s) - f(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s) \right]. \end{aligned}$$

674 Here, inequality (a) holds by the induction assumption, and (b) holds by submodularity.
 675 This completes the induction. \square

676 **Lemma C.3.** Let T be the number of user accesses and T' be a given upper bound. Algo-
 677 rithm 2 generates a set of guesses of T as $\mathcal{P} = \{\delta i \mid i \in \lceil [T'/\delta] \rceil\}$. Let T^* be the smallest
 678 integer in \mathcal{P} that is larger than or equal to T . Let $\mathcal{G}^1, \dots, \mathcal{G}^T$ be the output of STORM(T^*),
 679 and let $\mathcal{B}^1, \dots, \mathcal{B}^T$ be the output of Algorithm 2. It holds that

$$f(\bigcup_{t=1}^T \mathcal{B}^t) \geq \frac{1}{2} f(\bigcup_{t=1}^T \mathcal{G}^t). \quad (10)$$

680 *Proof.* Lemma C.3 is a direct result of Lemma C.2, as we can show that

$$\sum_{t=1}^{T-1} \left[f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) - f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s\right) \right] \geq -f\left(\bigcup_{t=1}^T \mathcal{B}^t\right).$$

681 To prove that the above inequality holds, we proceed to show that

$$\sum_{t=1}^{T-1} \left[-f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) + f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s\right) \right] \leq f\left(\bigcup_{t=1}^T \mathcal{B}^t\right).$$

682

683 Indeed, it is

$$\begin{aligned} & \sum_{t=1}^{T-1} \left[-f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) + f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s\right) \right] \\ &= \sum_{t=1}^{T-1} \left[f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^t \mathcal{G}^s\right) - f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) \right] \\ &= f(\mathcal{B}^1 \mid \mathcal{G}^1) + \sum_{t=1}^{T-2} \left[f\left(\bigcup_{s=1}^{t+1} \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) - f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) \right] - f\left(\bigcup_{s=1}^{T-1} \mathcal{B}^s \mid \bigcup_{s=1}^T \mathcal{G}^s\right) \\ &\leq f(\mathcal{B}^1) + \sum_{t=1}^{T-2} \left[f\left(\bigcup_{s=1}^{t+1} \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) - f\left(\bigcup_{s=1}^t \mathcal{B}^s \mid \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) \right] \\ &= f(\mathcal{B}^1) + \sum_{t=1}^{T-2} \left[f\left(\bigcup_{s=1}^{t+1} \mathcal{B}^s + \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) - f\left(\bigcup_{s=1}^t \mathcal{G}^s\right) - f\left(\bigcup_{s=1}^t \mathcal{B}^s + \bigcup_{s=1}^{t+1} \mathcal{G}^s\right) + f\left(\bigcup_{s=1}^{t+1} \mathcal{G}^s\right) \right] \\ &= f(\mathcal{B}^1) + \sum_{t=1}^{T-2} f(\mathcal{B}^{t+1} \mid \bigcup_{s=1}^t \mathcal{B}^s + \bigcup_{s=1}^{t+1} \mathcal{G}^s) \leq f(\mathcal{B}^1) + \sum_{t=1}^{T-2} f(\mathcal{B}^{t+1} \mid \bigcup_{s=1}^t \mathcal{B}^s) \\ &= f\left(\bigcup_{t=1}^{T-1} \mathcal{B}^t\right) \leq f\left(\bigcup_{t=1}^T \mathcal{B}^t\right) \end{aligned} \tag{11}$$

684 Combining Equation (12) with Equation (8), we can obtain Equation (10).

685

□

686 **Theorem 3.2.** *Algorithm 2 has a competitive ratio of $1/8\delta$, space complexity $\mathcal{O}(T'^2 k/\delta)$,
687 and time complexity $\mathcal{O}(NkT'^2/\delta)$.*

688 *Proof.* Let T^* be the smallest integer in $\mathcal{P} = \{\delta i \mid i \in \lceil T'/\delta \rceil\}$ that is larger than or equal
689 to T . Let $\mathcal{G}^1, \dots, \mathcal{G}^{T^*}$ be the output of STORM(T^*)

690 Observe that $T^* \leq T + \delta - 1$, otherwise we can set T^* as $T^* - \delta$ and it is still an upper
691 bound of T . By Theorem 3.1, it holds that

$$f\left(\bigcup_{t=1}^T \mathcal{G}^t\right) \geq \frac{1}{4(T^* - T + 1)} f\left(\bigcup_{t=1}^T \mathcal{O}^t\right) \geq \frac{1}{4\delta} f\left(\bigcup_{t=1}^T \mathcal{O}^t\right). \tag{13}$$

692 Combining Equation (13) with Lemma C.3, we conclude that

$$f\left(\bigcup_{t=1}^T \mathcal{B}^t\right) \geq \frac{1}{2} f\left(\bigcup_{t=1}^T \mathcal{G}^t\right) \geq \frac{1}{8\delta} f\left(\bigcup_{t=1}^T \mathcal{O}^t\right). \tag{14}$$

693 Since STORM++ maintains $\lceil T'/\delta \rceil$ copies of STORM, each copy of STORM has time complexity
694 in $\mathcal{O}(NkT')$ and space complexity in $\mathcal{O}(T'k)$, the space complexity for the STORM++ is
695 $\mathcal{O}(T'^2 k/\delta)$, and the time complexity is $\mathcal{O}(NkT'^2/\delta)$. □

D Additional content for Section 4

D.1 Dataset and experimental setting

We use the following six datasets:

- **KuaiRec** [14]: Recommendation logs from a video-sharing mobile app, containing metadata such as each video’s categories and watch ratio. We use the *watch ratio*, computed as the user’s viewing time divided by the video’s duration, as the user rating score. The dataset includes 7 043 videos across 100 categories.
- **Anime**:² A dataset of animation ratings in the range $[1, 10]$, consisting of 7745 animation items across 43 genres.
- **Beer**:³ This dataset contains BeerAdvocate reviews [32, 31], along with categorical attributes for each beer. After filtering out beers and reviewers with fewer than 10 reviews, the final dataset includes 9 000 beers across 70 categories.
- **Yahoo**:⁴ A music rating dataset with ratings in the range $[1, 5]$. After filtering out users with fewer than 20 ratings, the dataset includes 136 736 songs across 58 genres.
- **RCV1** [28]: A multi-label dataset, each item is a news story from Reuters, Ltd. for research use. We sample 462 225 stories across 476 unique labels.
- **Amazon** [30]: A multi-label dataset consisting of 1 117 006 Amazon products over 3 750 unique labels. This dataset is a subset of the publicly available dataset.

Experimental setting To obtain the click probabilities for each item, we generate them uniformly at random in the range $[0, 0.2]$ for the RCV1 and Amazon datasets. For the four item-rating datasets, we estimate click probabilities by computing a low-rank completion of the user-item rating matrix using matrix factorization [25]. This yields latent feature vectors w_u for each user u and v_m for each item m , where the inner product $w_u^\top v_m$ approximates user u ’s rating for item m . We then apply standard min-max normalization to the predicted ratings and linearly scale the results to the range $[0, 0.5]$ to obtain click probabilities. We set the latent feature dimension to 15 for the KuaiRec dataset, and to 20 for the remaining three rating datasets. While click probability ranges can vary in real-world scenarios, where larger ranges can lead to faster convergence toward the maximum expected coverage value, we intentionally set the probabilities to be small in our experiments. This allows us to observe performance changes over a wider range of parameter settings.

Implementation To enhance the computational efficiency of the LMGREEDY algorithm (described in Algorithm 3), we implement lines 6–9 using the STOCHASTIC-GREEDY approach introduced by Mirzasoleiman et al. [35]. For the implementations of STORM and STORM++, we incorporate the sampling technique from Feldman et al. [10]. Specifically, when adding each item (with potential replacement) to each candidate set, we ignore it with probability $2/3$.

D.2 Omitted results

In this section, we present additional experimental results and analyses. Specifically, we show how the expected coverage changes as the budget k varies in Figure 7, and as the number of user visits T varies in Figure 8. The standard deviation of the expected coverage, as k and T are varied, is reported in Figure 10 and Figure 11, respectively. Finally, we present the effect of varying δ and ΔT on the expected coverage in Figure 9.

Why Storm++ and Storm(T') outperform Storm(T)? We present a simple example where STORM(T') outperforms STORM(T). Consider the input stream $\mathbf{S} = ((\{1\}, 1, 0), (\{2\}, 1, 0), (\{3, 4\}, 0.9, 1), (\{5, 6\}, 1, 1))$ with budget $k = 1$ and actual number of

²<https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database>

³<https://cseweb.ucsd.edu/~jmcauley/datasets.html>

⁴<https://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67>

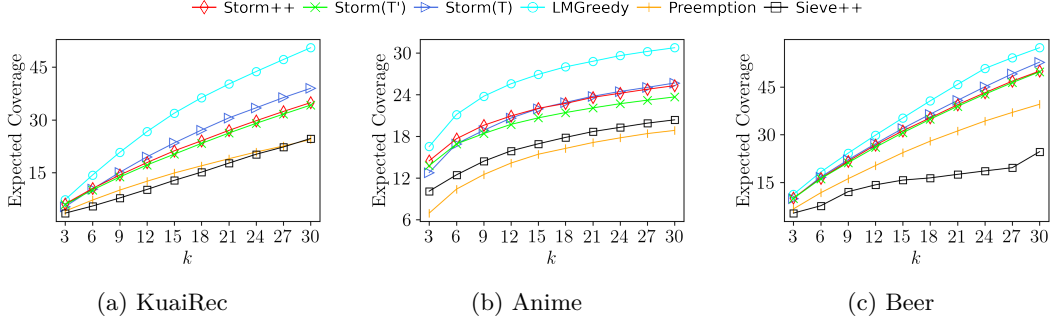


Figure 7: Empirical variation in expected coverage as a function of budget k on three smaller datasets. Parameter $T = 5$, $\delta = 25$ and $\Delta T = 45$ are fixed

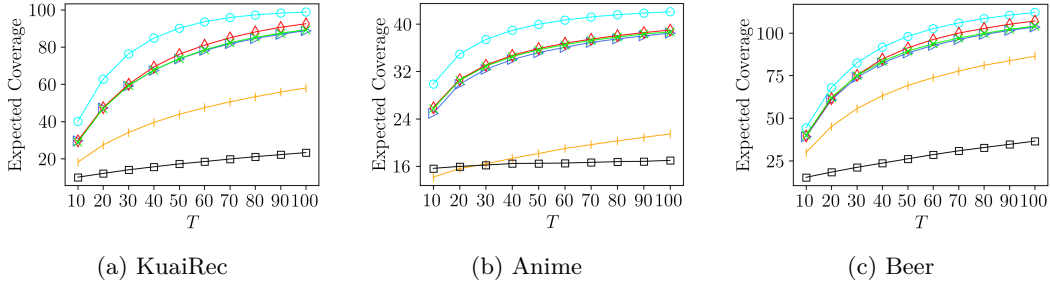


Figure 8: Empirical variation in expected coverage as a function of number of visits T on three smaller datasets. Parameter $k = 10$, $\delta = 10$ and $\Delta T = 10$ are fixed.

visits $T = 2$. The algorithm STORM(2) selects either $\{1\}$ or $\{2\}$ for the first visit, and $\{5, 6\}$ for the second, achieving expected coverage of 3.

If we overestimate the visits and set $T' = 3$, STORM(3) instead selects $\{3, 4\}$ first and $\{5, 6\}$ second, yielding an expected coverage of 3.8, which is higher than when the number of visits is known exactly. Likewise, STORM++ with $T' = 3$ and $\delta = 3$ matches STORM(3), and achieves the same coverage of 3.8.

This example shows that while STORM(T) has the strongest theoretical guarantee, STORM(T') and STORM++ can outperform it empirically, which explains the trends observed in our experiments.

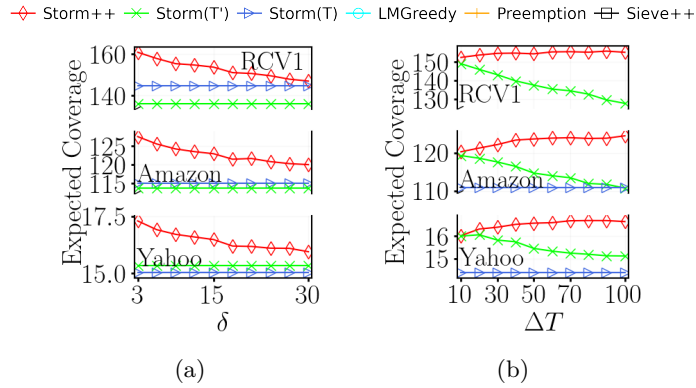


Figure 9: Impact of ΔT and δ on expected coverage on three larger datasets. In (a) and (b), we fix $k = 5$ and $T = 10$. When varying δ , we fix $\Delta T = 60$; when varying ΔT , we fix $\delta = 10$.

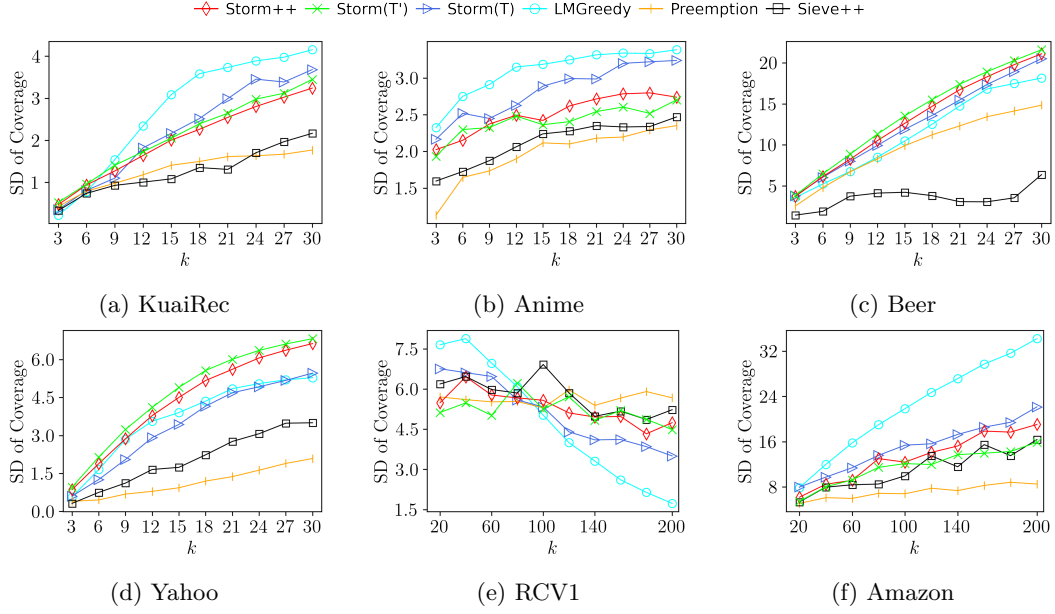


Figure 10: Empirical variation in standard deviation for the expected coverage as a function of budget k on all datasets. Parameter $T = 5$, $\delta = 25$ and $\Delta T = 45$ are fixed.

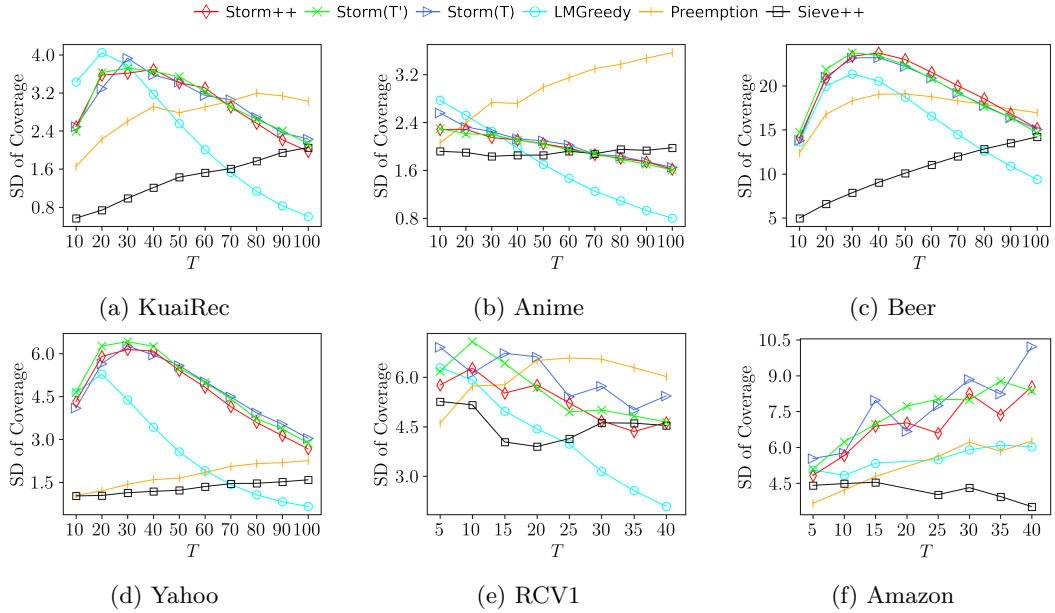


Figure 11: Empirical variation in standard deviation for the expected coverage as a function of the number of user visits T on all datasets. Parameter $k = 10$, $\delta = 10$ and $\Delta T = 10$ are fixed.