arXiv:2601.10901v1 [cs.DS] 15 Jan 2026

具有按需用户请求的流式随机子模最大化

Honglian WangKTH皇家理工学院斯德哥尔摩，瑞典

Sijing TuKTH皇家理工学院斯德哥尔摩，瑞典

Lutz Oettershagen利物浦大学利物浦，英国

Aristides GionisKTH皇家理工学院斯德哥尔摩，瑞典

摘要

我们研究了一个受新闻推荐平台动态启发的流式子模最大化新问题。我们考虑这样一种场景：用户可以随时访问新闻网站，而网站每次必须展示至多 k 条新闻。用户交互本质上具有随机性：向用户展示的每条新闻都会以一定的接受概率被用户阅读，且每条新闻都涵盖若干主题。我们的目标是设计一种流式算法，使预期的主题总覆盖最大化。为解决这一问题，我们建立了该问题与受拟阵约束的子模最大化之间的联系。我们证明，当用户访问次数预先已知，或可使用与流长度呈线性关系的内存时，可以有效地调整以往方法来解决这一问题。然而，在更贴近现实的场景中，如果只能获得访问次数的上界且内存为次线性，算法便无法保证任何有界的性能。为克服这些局限，我们提出了一种新的在线流式算法，其竞争比为$1/(8\delta)$，其中 δ 控制近似质量。此外，该算法只需对数据流进行单次遍历，且所用内存与流长度无关。实验结果表明，我们的算法始终优于基线方法。

1 引言

在各种可行性约束下，流式子模最大化已得到广泛研究，包括匹配拟阵 [7, 8, 10], 拟阵 [9], 背包 [21, 22] 和 k-集合 [20] 约束。在经典设定中，目标是从数据流中选取一个项目子集，并在数据流结束时输出该子集。然而，在许多现实场景中，这种经典框架并不适用，例如在线新闻推荐中的内容覆盖最大化。在这种设定下，用户会多次访问新闻网站，并期望每次访问时都能收到一组相关的文章。同时，新文章以数据流的形式持续到达内容提供商服务器。长期目标是最大化用户访问期间的总内容覆盖。现有的流式子模最大化算法并不适用于这一用例，因为它们旨在生成单个解，且不支持数据流期间的多次或按需用户交互。

受这一局限的启发，我们提出了一个称为具有按需请求的流式随机子模最大化 （*S3MOR*）的新问题。虽然该设定广泛适用于具有多种输出要求的流式子模最大化任务，但我们以新闻推荐任务为例进行说明。因此，在*S3MOR*中，数据流由不断到来的新闻条目组成，系统将这些新闻条目汇总到一个个性化新闻网站中。每个项目 V_i 涵盖一组

第39届神经信息处理系统大会（NeurIPS 2025）。

表1：我们算法的比较。 $N \in \mathbb{N}$ 表示数据流长度， $k \in \mathbb{N}$ 表示每次用户访问要展示的新闻条目数， $T \in \mathbb{N}$ 和 $T' \in \mathbb{N}$ 分别表示用户访问的确切次数和上界， $\delta \in \mathbb{N}$表示近似参数。

算法	竞争比	Time	空间	响应时间
LMGREEDY	$1/2$	$\mathcal{O}(NT'k)$	$\Theta(N+kT)$	$\mathcal{O}(Nk)$
STORM	$1/4(T'-T+1)$	$\mathcal{O}(NT'k)$	$\mathcal{O}(T'k)$	$\mathcal{O}(T')$
STORM++	$1/(8\delta)$	$\mathcal{O}(NT'^2k/\delta)$	$\mathcal{O}(T'^2k/\delta)$	$\mathcal{O}(T')$

主题，并与概率 p_i相关联，该概率表示用户看到该项目时点击它的概率。用户可以在数据流期间任意频繁地访问系统，并且访问时间不受限制。每次访问时，系统必须从截至该时刻已经到达数据流中的项目中选择并呈现至多 k 个物品。如果用户点击项目 V_i，他们就会接触到与该项目相关联的所有主题。目标是最大化所有访问中用户点击所覆盖的不同主题总数的期望值。由于其在线特性，这一设定颇具挑战性：

• **按需**：在任何时刻，系统都必须能够输出新闻条目的一个子集。

• **不可撤销性**：一旦向用户呈现某个子集，就无法对其进行修改。

据我们所知，现有算法中没有任何一种能够直接解决*S3MOR* 问题。

我们使用竞争比这一概念，评估作出不可撤销决策的在线流式算法的性能。如果对于每个输入流 σ，算法 $\mathcal{A}$ 所达到的值 $\mathcal{A}(\sigma)$都满足 $^{\mathcal{A}}{}_{(\sigma)}\ \mathbf{OPT}(\sigma) \geq c$，则称该算法为 -竞争的；其中，$\mathbf{OPT}(\sigma)$ 表示在能够完整访问输入流的情况下，最优离线解的值。

为评估我们算法的性能，我们明确响应时间复杂度、时间复杂度和空间复杂度的概念。

响应时间复杂度与时间复杂度。对于*S3MOR* 问题，我们假设存在一个预言机，能够针对任意可行候选集 S 评估目标函数 $f(S)$。我们将算法的响应时间复杂度定义为用户提交请求后，生成大小为 k 的推荐结果所需的预言机调用次数的最坏情况。该指标反映了用户体验到的延迟。相比之下，时间复杂度定义为算法在整个数据流中所需预言机调用次数的最坏情况。

空间复杂度。 在实践中，所有用户共享用于存储原始新闻条目的存储空间。我们区分两种存储类型：共享存储，其中包含所有用户均可访问的原始项目（不计入空间复杂度）；以及用户级存储，用于存放每位用户计算个性化推荐所需的额外内存，包括文档标识符和用户—项目关联概率。本文始终将“空间复杂度”专指用户级存储。

我们的贡献。 我们将*S3MOR* 归约为分区拟阵约束下的流式次模最大化问题，从而能够利用现有的在线流式算法。基于这一归约，我们提出了三种算法，并从多个维度对其进行评估：竞争比、空间复杂度、时间复杂度和响应时间复杂度。我们的主要结果如下：

• 我们证明，只要有足够的内存存储整个数据流，贪心算法 LMGREEDY 就能为*S3MOR* 问题实现可能达到的最佳 $1/2$ 竞争比。

• 如果已知用户访问的确切数量，则该归约能够保留底层在线流式次模算法的竞争比。

• 然而，在现实场景中，用户访问次数 T 是未知的，我们需要将访问次数猜测为 $T' > T$。针对这种情况，我们提出了一种算法 STORM，其竞争比为 $1\,4(T'-T+1)$。

• 为了获得更好的竞争比，我们提出了第二种算法 STORM++，它针对 T 同时进行多次猜测，并为每次猜测维护一个解。随后，它通过贪心方式汇总各次猜测的结果。该算法的竞争比为 $1/(8\delta)$，

其中，δ 是一个可调参数，用于平衡效率与解的质量。与底层流式算法相比，空间和时间复杂度增加了 $\delta T'$ 倍。

- STORM 和 STORM++ 的竞争比不如 LMGREEDY，但它们占用的空间要小得多，响应时间也显著更短。

我们在表1中总结了上述所有算法的结果。我们通过在大型和小型真实世界数据集上进行实证实验，验证了我们的方法。结果表明，与相关基线方法相比，我们的算法在覆盖质量和内存使用量方面表现出色。

符号与背景。 我们假设读者熟悉受约束子模最大化和竞争分析。我们在附录A中给出形式化定义。

相关工作

子模优化在机器学习、组合优化和数据分析中发挥着基础性作用， [13, 26, 29],用于刻画具有递减收益的问题，并应用于数据摘要、 [34, 42], 非参数学习、 [17, 45], 推荐系统、 [1, 33, 38, 44], 影响力最大化 [24], 和网络监控 [13]。经典贪心算法在基数约束下实现单调子模最大化的 $(1 - 1/e)$ 近似， [39], 并可扩展至拟阵约束 [6, 12]。

我们研究随机子模覆盖函数，其目标取决于随机实现，但选择必须以非自适应方式进行。这与自适应子模性形成对比，后者建模了带反馈的序贯决策，并为贪心策略提供了强有力的保证。相关工作包括集合覆盖和子模覆盖的随机变体，其中目标本身是随机的；典型例子分析了自适应性差距，以及在预言机访问下的近似界。

（受约束的）子模优化的经典贪心算法假设可以随机访问完整数据集，这在大规模或流式处理设定下并不可行。这一挑战促使研究者提出流式算法，在内存约束下做出不可撤销决策。Buchbinder 等人采用抢占范式，即用更优元素替换已有元素，从而在基数约束下针对单调函数实现紧确的 $(1/2 - \varepsilon)$ 近似。Sieve 使用阈值化，在仅需 $\mathcal{O}(k)$ 内存的情况下，于基数约束下实现 $(1/2 - \varepsilon)$ 近似。针对拟阵和分区约束的扩展包括 [7,8, 10]；例如，Chekuri 等人提出了一种针对单调 p-匹配拟阵的 $1/4p$-竞争比在线流式算法。然而，这些方法无法处理随机项目效用或重复且不可预测的用户访问。我们通过研究流式模型下、内存有限且用户访问请求未知时，分区约束下的随机子模覆盖问题，弥补这一空缺。

为将次模性推广到序列，Alaei 等人 [1]引入了序列次模性和序列单调性，并证明贪心算法可实现 $(1 - 1/e)$-近似，其应用包括在线广告分配。类似地，Ohsaka 和 Yoshida [40] 提出了 k-次模函数，用于对 k 个不相交子集进行优化，并为单调情形提供常数因子近似。与这些设定不同，我们的问题要求随着项目到达流的推进做出不可撤销决策，因此每个 k 子集都必须逐步构建，而不能等看到完整输入后一次性构建。

最后，多样性感知推荐与覆盖问题密切相关，因为许多系统必须推荐物品集合，而非单个物品，这促使人们采用次模方法来最大化覆盖和用户满意度 [2, 43]。Ashkan 等人 [3] 提出了一种离线贪心方法，用于在次模多样性约束下最大化效用。Yue 和 Guestrin [44] 采用次模老虎机进行多样化检索，通过自适应方式学习用户交互概率，而不像我们的设定那样依赖已知的预言机。

2 流式子模最大化与按需请求

问题设定。 令 $\mathcal{V} = \{V_1, V_2, \ldots, V_N\}$ 为数据流中出现的 N 个新闻条目组成的有序集合，其中，项目 V_i 在时间步 i到达。每个新闻条目 $V_i \in \mathcal{V}$ 涵盖预定义主题集合 $C = \{c_1, c_2, \ldots, c_d\}$中的一个子集，其中 d 表示主题总数。形式化地，

$V_i \subseteq C$对于每个 $i \in [N]$。此外，每个新闻条目 V_i都关联一个概率[1] $p_i \in [0,1]$，用于表示用户看到该项目时点击它的概率。当用户点击某个新闻条目时，我们称该用户接触到了该新闻条目中的所有主题。

此外，我们假设可以观察到一个二元变量 $\tau_i \in \{0,1\}$，用于指示用户是否在时间步 i访问系统。具体而言，$\tau_i = 1$表示用户在时间步 i访问系统，$\tau_i = 0$则表示未访问。每当 $\tau_i = 1$时，系统应从可用新闻条目 $\{V_1, \ldots, V_i\}$中至多展示 k条新闻条目。现在，假设用户访问系统 T次。系统输出 T个集合 $\mathcal{S}^1, \cdots, \mathcal{S}^T$，其中 $\mathcal{S}^t$表示用户第 t次访问时向其展示的物品，且 $t \in [T]$。我们允许同一物品在多次访问中重复展示；不过，我们将每次展示视为一个独立的副本。之所以这样处理，是因为如果某个物品被展示多次，用户便有多次点击该物品的机会。因此，对于任意 $t \neq s$，都有 $\mathcal{S}^t \cap \mathcal{S}^s = \emptyset$。我们将整个访问序列中向用户展示的所有物品构成的集合定义为 $\mathcal{S} = \bigcup_{t=1}^{T} \mathcal{S}^t$。

目标函数。我们将 $f(\mathcal{S})$定义为用户在 T次访问后覆盖的主题数的期望值，其中期望是对点击概率取的。令 $\sigma_j \in \{0,1\}$表示主题 c_j是否被覆盖，即用户是否点击了至少一个覆盖来自所展示集合 c_j的项目 $\mathcal{S}^1, \cdots, \mathcal{S}^T$。主题 c_j仍未被覆盖的概率 $\Pr(\sigma_j = 0)$等于用户未点击任何覆盖 c_j的项目的概率。
由于用户以概率 p_i点击项目 V_i，因此不点击该项目的概率为 $1 - p_i$，从而 $\Pr(\sigma_j = 0) = \prod_{t\in[T]} \prod_{V_i \in \mathcal{S}^t, V_i \ni c_j} (1 - p_i)$，进而得到期望$\mathbb{E}[\sigma_j] = 1 - \prod_{t\in[T]} \prod_{V_i \in \mathcal{S}^t, V_i \ni c_j} (1 - p_i)$。根据期望的线性性质，我们将用户覆盖的主题数的期望值记为 $f(\mathcal{S})$，其表达式为

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

引理 2.1。 $f(\mathcal{S})$ 是一个非递减的子模集合函数。

问题定义。我们正式定义以下新问题： 带按需请求的流式随机次模最大化 (*S3MOR*)。

问题 2.2 (***S3MOR***)。我们将新闻条目流 $\mathbf{S}$ 定义为由三元组组成的序列：$\mathbf{S} = ((V_1, p_1, \tau_1) \ldots, (V_N, p_N, \tau_N))$其中 N是数据流的（未知）长度。在每个时间步 i，系统都会接收一个三元组 (V_i, p_i, τ_i)，其中 $V_i \in \mathcal{V}$、 $p_i \in [0,1]$和 $\tau_i \in \{0,1\}$。每当$\tau_i = 1$时，系统会从迄今收到的物品集合 $\{V_1, \ldots, V_i\}$ 中选择至多 k个新闻条目呈现给用户。令 $\mathcal{S} = \{\mathcal{S}^1, \ldots, \mathcal{S}^T\}$ 表示在 T次访问中呈现给用户的新闻条目集合。目标是最大化用户在数据流结束前接触到的不同主题的期望数量，该数量由函数 $f(\mathcal{S})$度量。

我们的流式设定面临两个关键挑战：(i) 系统无法预先知道用户何时访问系统或会访问多少次；(ii) 系统必须根据目前可获取的信息做出决策，而且这些决策不可撤销。接下来，我们将展示如何将该问题归约为一个受分区拟阵约束的次模最大化问题。然后，我们利用这一联系，在各种内存约束下设计解决方案。
归约为分区拟阵约束下的子模最大化。我们首先将 *S3MOR* 问题（问题 2.2）归约为分区拟阵约束下的子模最大化，从而能够应用具有可证明保证的高效流式算法。令 $r\colon t \mapsto r_t$ 将用户的第 t次访问映射到其访问时间，并令 $(r_t)_{t=1}^{T}$ 表示用户在一段未知时间内访问系统的时间序列。每当用户访问系统时，系统都会从自数据流开始以来可用的所有新闻条目中选出至多 k 个新闻条目，并将其呈现出来，记为$\{V_i\}_{i=1}^{r_t}$。

[1]这些概率可视为用户-条目偏好或参与度的度量，通常通过逻辑回归、因子分解机或神经网络等机器学习模型，根据用户行为模式和条目特征从历史点击数据中估计得出。我们假设这些概率是静态的，并且随时间保持不变。

令 V_i^t 为与第 t 次访问关联的项目 V_i 的一个副本。令 $\mathcal{V}^t$ 表示截至第 t 次访问收集到的这些副本的集合，即 $\mathcal{V}^t = \{V_i^t\}_{i=1}^{r_t} = \{V_1^t, \ldots, V_{r_t}^t\}$。不失一般性地，我们将每个副本视为不同的项，即对于所有 $t \neq s$，均有 $V_i^t \neq V_i^s$。显然，对于 $t \neq s$，有 $\mathcal{V}^t \cap \mathcal{V}^s = \emptyset$。
此外，令 $\mathcal{S}^t \subseteq \mathcal{V}^t$ 为系统在用户第 t 次访问时向其展示的新闻条目集合。令 $\mathcal{S} = \bigcup_{q=1}^{T} \mathcal{S}^q$ 为所有已展示物品集合的并集。类似于公式（1），我们对函数 f： $2^{\cup_{t=1}^{T} \mathcal{V}^t} \to \mathbb{R}$ 进行调整，使得 $f(\mathcal{S})$ 表示在向用户展示 $\mathcal{S}^1$、.....、$\mathcal{S}^T$ 后，用户点击的新闻条目（副本）所涵盖的主题数的期望值。

观察 2.3。 令 $(\cup_{t=1}^{T} \mathcal{V}^t, \mathcal{F})$ 为一个集合系统，其中 $\mathcal{F}$ 是 $\cup_{t=1}^{T} \mathcal{V}^t$ 的子集族，而 $\mathcal{F}$ 按如下方式构造：对于任意集合 $X \subseteq \cup_{t=1}^{T} \mathcal{V}^t$，若对于任意 $1 \leq t \leq T$，都有 $|X \cap \mathcal{V}^t| \leq k$，则 $X \in \mathcal{F}$。注意，$(\cup_{t=1}^{T} \mathcal{V}^t, \mathcal{F})$ 是一个分划拟阵。问题 *2.2* 可以等价地表述为 最大化$_{X \in \mathcal{F}} f(X)$。

基于观察 2.3，我们为问题 2.2 开发了一种基线算法。假设系统会存储截至用户第 t 次访问系统时收到的所有新闻条目，并且还存储了此前向该用户展示过的所有新闻条目集合，即 $\cup_{q=1}^{t-1} \mathcal{S}^q$。在这种情况下，截至时间 r_t 存储的新闻条目构成拟阵的一个分区，即 $\mathcal{V}^t = \{V_i^t\}_{i=1}^{r_t}$。随后，我们应用 Fisher 等人提出的局部贪心算法 [12], ，也就是说，系统会从 $\mathcal{V}^t$ 中贪心地选择 k 个新闻条目，以最大化目标函数相对于 f 的边际增益（$\cup_{q=1}^{t-1} \mathcal{S}^q$）。该算法不需要了解任何关于 T 的信息，且竞争比为 $1/2$。由于所有新到达的新闻条目都需要存储，内存使用量为 $\Omega(N + kT)$，运行时间为 $\sum_{q=1}^{T} \mathcal{O}(|\mathcal{V}^q|k)$。响应时间为 $\mathcal{O}(Nk)$，因为该算法会从 $\mathcal{O}(N)$ 个项目中迭代选择 k 个项目。我们在附录 B 的算法 3（LMGREEDY）中给出了该算法。

定理 2.4。 针对问题 *2.2* 的算法 *3* 具有 $1/2$ 的竞争比、 Θ（$N + kT$）的空间复杂度、 $\mathcal{O}$（NTk）的时间复杂度，以及 $\mathcal{O}$（Nk）的响应时间复杂度。此外， $1/2$ 的竞争比是紧的，即对于问题 *2.2*，任何流式算法若不违反不可撤销约束，都无法达到优于 $1/2$ 的竞争比。

LMGREEDY 的缺点是其内存使用量随流长度线性增长，因此对于大型数据流而言不可行。因此，我们考虑系统具有有限内存、只能存储用户特定子集的新闻条目的情形。

3 有限内存下的算法

现在，我们考虑系统具有有限内存的更现实情形。我们的目标是在实现有界近似比的同时，大幅降低内存使用量和响应时间复杂度。我们首先证明，任何能够实现相当良好性能的算法都至少需要使用 $\Theta(Tk)$ 内存。

为说明这一上界，考虑以下示例：假设用户发出 T 个请求；输入流包含超过 Tk 个物品，每个物品包含一个不同主题，且其接受概率为 1。在数据流结束时，用户依次提交 T 个请求。如果算法仅分配到 $\Theta(T'k)$ 内存（其中 $T' \leq T$），则最多只能存储 $\Theta(T'k)$ 个物品，因此最多只能存储 $\Theta(T'k)$ 个不同主题。然而，最优情况下，用户可能访问 $\Theta(Tk)$ 个不同主题。由此可知，在此示例中，竞争比至多为 $\Theta(T'/T)$；当 $T' \ll T$ 时，该比值可以任意差。

由于通常无法预先得知请求的确切数量 T，我们假设系统可以获取其上界 $T' \geq T$，并分配 $\Omega(kT')$ 内存。这一分配足以在用户交互期间存储所有候选物品。实践中，可以根据历史交互数据估算 T'，从而使系统能够在合理精度范围内将 T' 设为接近 T，确保内存利用率高效，同时提供有意义的性能保证。

3.1 首个内存高效的流式算法

我们的第一个节省内存的算法按如下方式处理到达流$\mathbf{S}$。首先，它为 $i \in [T']$初始化T'空候选集 $\mathcal{A}^i$，这些候选集最初都处于活跃状态。数据流$\mathbf{S}$中有新闻条目到达时，对于每个活跃候选集，算法要么将该项目添加到集合中，要么

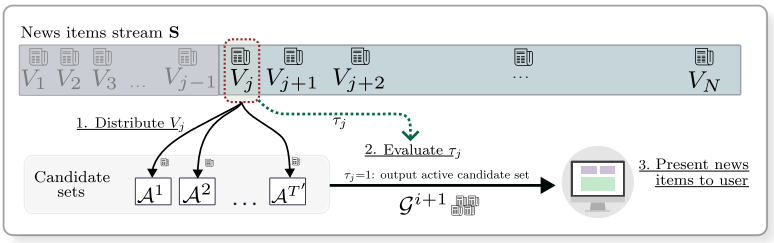


图 1：算法 STORM 的示意图。我们首先初始化 T' 个空的活跃候选集。对于数据流 **S** 中每个到达的新闻条目，我们都会判断是否可以将其添加到每个活跃候选集中，或替换其中的项目。当用户提交请求时，即 $\tau_j = 1$，我们选择最佳的活跃候选集，将其呈现给用户，并将其停用。

算法 1： STORM

输入： 估计访问次数 $T' \in \mathbb{N}$，预算 $k \in \mathbb{N}$。

$\mathcal{A}^1, \cdots, \mathcal{A}^{T'} \leftarrow \emptyset$, $\mathcal{T} = \{1, \cdots, T'\}$, $\mathcal{G}^0 \leftarrow \emptyset$, $i \leftarrow 0$

对 (V, τ, p) 在数据流中**执行**

  对 $j \in \mathcal{T}$ **执行**

    若 $|\mathcal{A}^j| < k$ **则**

      $\mathcal{A}^j \leftarrow \mathcal{A}^j \cup \{V\}$

    else

      $V' \leftarrow \arg\min_{V \in \mathcal{A}^j} \nu(f, \cup_{q \in [T']} \mathcal{A}^q, V)$

      若 $f(V \mid \cup_{q \in [T']} \mathcal{A}^q) \geq 2\nu(f, \cup_{q \in [T']} \mathcal{A}^q, V')$ **则**

        $\mathcal{A}^j \leftarrow \mathcal{A}^j \setminus \{V'\} \cup \{V\}$

  如果 $\tau = 1$**那么**

    $j^* \leftarrow \arg\max_{j \in \mathcal{T}'} f(\mathcal{A}^j \mid \bigcup_{t=0}^{i} \mathcal{G}^t)$

    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{j^*\}$ ▷停用候选集 $\mathcal{A}^{j*}$

    $\mathcal{G}^{i+1} \leftarrow \mathcal{A}^{j^*}$

    输出 $\mathcal{G}^{i+1}$

    $i \leftarrow i + 1$

在集合已满时替换其中一个现有项目。用户访问系统时，算法会从活跃候选集中选出一个呈现给用户。随后，所选候选集会被标记为非活跃，不再进行更新。图 1 展示了整体概况。

基于将问题归约为分区拟阵约束下的子模最大化，我们可以调整任何针对拟阵约束设计的流式算法，前提是该算法生成不可撤销的输出。值得一提的例子包括 Chakrabarti 和 Kale [7], 、Chekuri 等人 [8], 以及 Feldman 等人 [10], 提出的算法，它们都达到了 $1/4$ 近似比。在我们的工作中，我们具体调整了 Chekuri 等人提出的算法 [8]。

我们的算法名为 STORM，如算法 1 所示。它使用计数器 i 跟踪访问次数。假设新闻项目 V_j 到达时，用户已经访问了 i 次。这意味着在 V_j 到达之前，来自 i 个分区的 i 个活跃候选集已呈现给用户，因此，V_j 及其后的任何项目都不会被添加到这些分区中。算法遍历剩余的 $T' - i$ 个活跃候选集，并确定是否应将 V_j 添加到其中任何一个集合中，必要时进行替换。

是否用当前项目 V_j 替换现有项目 V，取决于二者相对于集合 $\cup_{t \in [T']} \mathcal{A}^t$ 的增量值比较。具体而言，我们采用 Chekuri 等人 [8]提出的记号 $\nu(f, \cup_{t \in [T']} \mathcal{A}^t, V)$，并定义

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

算法 2： STORM++

输入：估计访问次数 $T' \in \mathbb{N}$、**预算** $k \in \mathbb{N}$、**参数** $\delta \in \mathbb{N}$。

$\mathcal{G}^0 \leftarrow \emptyset$ $\mathcal{B}^0 \leftarrow \emptyset$ $i \leftarrow 0$ $\mathcal{P} \leftarrow \left\{\delta\ 2\delta\ \dots\ \lceil \frac{T'}{\delta} \rceil \delta\right\}$，，，，，，对于 执行 初始化 STORM (，

$\tilde{T} \in \mathcal{P}$ $\tilde{T}$ k

) ▷ 针对不同的 $\tilde{T}$ 维护 STORM 3 对于数据流中的 (V, τ, p) 执行 4 对于 $T \in \mathcal{P}$ 执行

$\mathcal{G}^{i+1,(\tilde{T})} \leftarrow$ **STORM (** $\tilde{T}$, k**).step(**V, τ, p**)** ▷ 通过执行算法 1 的第 3–15 行处理 (V, τ, p)

6 如果 $i = 1$ 则 7 $\tilde{T}^* \leftarrow \arg\max_{\tilde{T} \in \mathcal{P}} f(\mathcal{G}^{i+1,(T)} \mid \bigcup_{j=0}^{i} \mathcal{B}^j)$ 8 $\mathcal{B}^{i+1} \leftarrow \mathcal{G}^{i+1,(\tilde{T}^*)}$ 9 输出 $\mathcal{B}^{i+1}$ 10 $i \leftarrow i + 1$

这里，$V' < V$表示项目 V' 在项目 V之前被加入集合 $\cup_{t \in [T']} \mathcal{A}^t$。对于特定的集合 $\mathcal{A}^{t'}$，如果 $f_{\cup_{t \in [T']} \mathcal{A}^t}(V_j) \geq 2 \cdot \min_{V \in \mathcal{A}^{t'}} \nu(f, \cup_{t \in [T']} \mathcal{A}^t, V)$，则 V_j 会替换 $\mathcal{A}^{t'}$ 中的一个项目。

在第 t 次用户访问时，系统以贪心方式选择要输出的活跃候选集。具体而言，令 $\mathcal{G}^1$，$\cdots$，$\mathcal{G}^{t-1}$ 表示已向用户展示过的候选集，并令 $\mathcal{T}$表示剩余活跃集的索引。系统选择相对于此前已展示集合的并集能够带来最高增量收益的活跃集。形式上，令 $j^* = \arg\max_{j \in \mathcal{T}} f(\mathcal{A}^j \mid \cup_{j=1}^{t-1} \mathcal{G}^j)$，系统输出 $\mathcal{G}^t = \mathcal{A}^{j^*}$。

定理 3.1. 设 T为用户访问次数，T'为给定的上界。算法 *1* 的竞争比为 $\frac{1}{4(T'-T+1)}$，空间复杂度为 $\mathcal{O}(T'k)$，时间复杂度为 $\mathcal{O}(NkT')$，响应时间复杂度为 $\mathcal{O}(T')$。

如果已知确切的访问次数 T，我们可以设置上界 $T' = T$，并使用算法 1 获得 $1/4$-竞争比。然而，如果 T'远大于 T，算法 1 的竞争比可能会变得任意差。请注意，我们的分析是紧确的。我们可以证明，对于任意固定的 T，都存在一个示例，使得算法 1 的竞争比至多为 $\frac{1}{T'-T+1}$，这与我们的分析仅相差一个常数因子。证明见 C 节。

3.2 具有有界竞争比的内存高效流式算法

如上一节所示，要使 STORM 达到足够大的竞争比，系统必须能够较准确地估计用户访问次数 T。然而在实际中，通常无法预先获得这一信息。为了解决这个问题，我们将范围 $[T']$离散化，为 T生成多个猜测值，并针对每个猜测值分别运行一个算法 1 实例。当用户访问系统时，算法 1 的每个副本都会输出一个解，我们从所有输出中选出最佳解。

具体而言，算法 2（命名为 STORM++）首先将 T的猜测值集合初始化为 $\mathcal{P} = \{\delta i \mid i \in [\lceil T'/\delta \rceil]\}$，其中 $\delta \in \mathbb{N}$ 是一个参数，用于控制竞争比与空间和时间复杂度之间的权衡。在这一构造中，始终存在一个猜测值 $T \in \mathcal{P}$，使其介于T和 $T+\delta$之间。因此，如果我们以 T作为输入参数运行算法 1，并将结果呈现给用户，那么就能保证竞争比至少为 $1/4\delta$，如定理 3.1 所示；然而，我们无法预先知道 T的值。因此，我们采用贪心策略，汇总算法 1 所有副本的结果，为每次用户访问确定一个输出。在第 i次用户访问时，我们收集每个以不同猜测值运行的 STORM 副本的第 i个输出 $T \in \mathcal{P}$，并选择增量收益最大的输出。该选择规则带来如下结果。

定理 3.2。 算法 *2* 的竞争比为 $1/(8\delta)$，空间复杂度为 $\mathcal{O}(T'^2 k/\delta)$，时间复杂度为 $\mathcal{O}(NkT'^2/\delta)$，响应时间复杂度为 $\mathcal{O}(T')$。

4 实证评估

我们通过实证评估所提出的算法 STORM 和 STORM++ 的性能。我们将 STORM(T) 和 STORM(T') 分别记为输入参数设为 T（即用户访问的确切数量）和 T'（即用户访问数量的上界）的 STORM 算法。

我们设计实验来回答以下研究问题：**(Q1)** 与其他基线方法相比，以及彼此相比，STORM++ 和 STORM(T')提供的解决方案有多有效？**(Q2)** 参数 δ 和 k 对 STORM++ 和 STORM(T') 性能有何影响？**(Q3)** STORM++ 和 STORM(T')对上界 T' 的敏感程度如何？我们的算法是否需要对 T' 进行非常精确的估计才能取得良好效果?
我们使用以下**基线方法**:

- LMGREEDY：我们的上界基线方法，假设内存大小随数据流长度线性增长，即内存可能不受限制（算法 3）。
- SIEVE++：基于 Kazemi 等人 [23], , 我们实现了 SIEVE++算法的启发式版本来解决 *S3MOR* 问题。具体调整如下：**(1)** 在首次用户访问之前，运行精确的 SIEVE++ 算法，并获得输出集合 $\mathcal{S}^1$。(2) 对于此后的每第 i 个输出，使用边际收益函数 $f(\cdot|\bigcup_{j=1}^{i-1}\mathcal{S}^i)$ 作为目标函数，在第 $(i-1)$ 次与第 i 次用户访问之间的数据流上运行 SIEVE++ 。
- PREEMPTION：我们以与上文介绍 SIEVE++ 相同的方式，改编 Buchbinder 等人 [5] 提出的算法来求解*S3MOR*问题。

数据集： 我们使用六个真实世界数据集，包括四个评分数据集 KuaiRec、Anime、Beer 和雅虎，以及两个多标签数据集 RCV1 和亚马逊。这些数据集包含的项目数量从几千到一百万不等，主题集合的大小则从约 40 到约 4000 不等。详情见 D.1 节。

实验设置： 为了模拟新闻条目流，我们随机打乱物品，构成长度为 N 的数据流。为了模拟用户访问序列，我们先创建一个全零序列，然后随机选择 T 个索引，并将对应位置设为 1。我们通过设置 $T' = T + \Delta T$ （其中 ΔT 为某个正数）来确定访问次数的上界。我们生成合成点击概率，详情见 D.1 节。
在所有实验中，我们随机选取了 50 位用户，并报告了雅虎、RCV1 和 Amazon 数据集的平均期望覆盖率。其他三个数据集的标准差和期望覆盖率结果见 D 节。我们还报告了最大数据集 Amazon 上的运行时间和内存使用量。

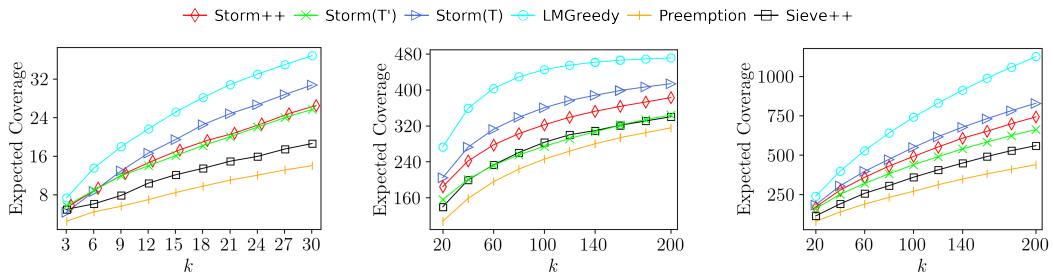
所有实验均在一台 MacBook Air（型号 Mac15,12）上运行，该设备搭载苹果 M3 芯片（8 核：4 个性能核心和 4 个能效核心）、16 GB 内存和固态硬盘（SSD）。源代码和数据集可在此处获取。[2]

结果与讨论

对 k 和 T 对期望覆盖率的影响。如图 2 所示，LMGREEDY 在所有数据集和 k 取值下均取得最佳性能。在我们提出的方法中，STORM(T) 始终表现最佳，其次是 STORM++，然后是 STORM(T')。与基线方法 SIEVE++ 和 PREEMPTION 相比，我们的算法在不同数据集和 k 取值下通常表现更好；但在 RCV1 上例外，PREEMPTION 的表现与 STORM(T') 相当。在图 3 中，我们观察到趋势发生了变化：在不同数据集和 T 取值下，STORM++ 略优于 STORM(T') 和 STORM(T)。STORM(T') 和 STORM(T) 的表现相当。这种性能变化是由于本次实验中的 δ 和 ΔT 较小，从而提高了 STORM(T') 和 STORM++ 的竞争比。尽管 STORM(T) 在这三种算法中具有最强的理论保证，但 STORM(T') 和 STORM++ 在实际测试中的表现可能优于它（我们在 D.2 节中举例说明了这一现象）。

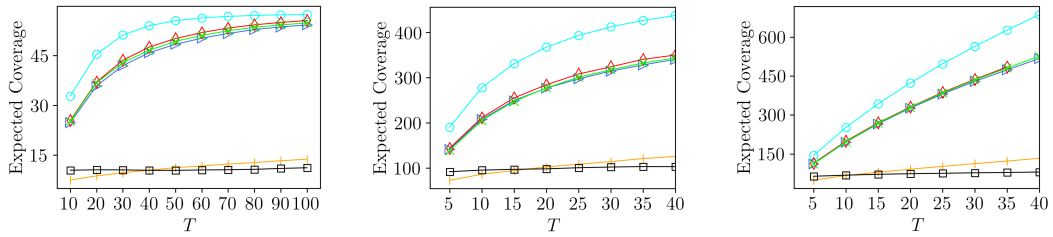
参数相关的性能 δ 和 ΔT。图 4a 显示，STORM++ 的期望覆盖率会随着 δ 增大而下降，这与理论一致：当 δ 从 3 增至 30 时，性能保证

[2] https://github.com/H ongLWang/按需用户请求下的流式次模最大化 ser-request



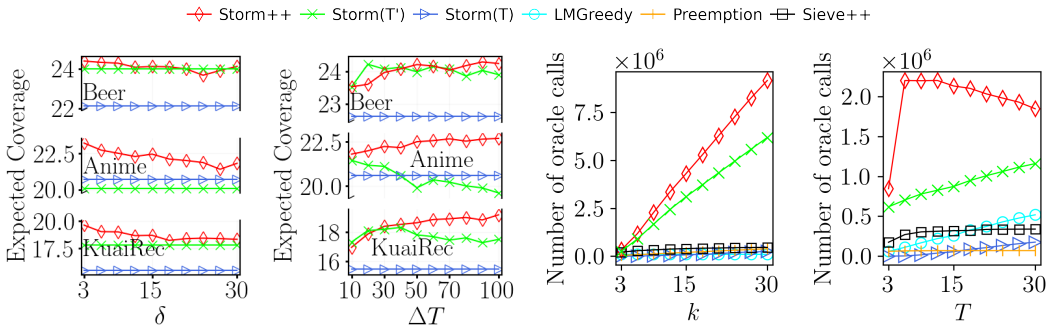
(a) 雅虎 (b) RCV1 (c) 亚马逊

图 2：所有数据集上期望覆盖率随预算变化的实证结果。参数 $T = 5$、$\delta = 25$ 和 $\Delta T = 45$ 固定



(a) 雅虎 (b) RCV1 (c) 亚马逊

Fig图3：所有数据集上期望覆盖率随访问次数变化的实证结果。参数 r
$k = 10$、$\delta = 10$ 和 $\Delta T = 10$ 为固定值。



(a) (b) (c) (d)

图4：左侧两图： ΔT和 δ对期望覆盖率的影响。在(a)和(b)中，我们固定 $k = 5$和 $T = 10$。改变 δ时，固定 $\Delta T = 60$；改变 ΔT时，固定 $\delta = 10$。右侧两图：Beer数据集上的预言机调用次数。我们固定 $\delta = 25$和 $\Delta T = 45$。改变 k时，固定 $T = 5$；改变 T时，固定 $k = 5$。

表2：运行时间和内存 用量（Amazon数据集）。我们设置 k， T，以及 δ 至 10， $T' = 20$.

	STORM++	S暴风雨(T')	S风暴(T)	LMGREEDY	SIEVE++	PREEMPTION
运行时间（秒）	35.13	7.94	4.38	88.13	307.43	136.04
内存（MB）	2.00	1.28	0.72	28.99	0.91	0.03

会降低 10 倍。这一点也体现在各数据集的覆盖率均有所下降，例如，亚马逊上的覆盖率从 130 降至 120。

图4b显示，随着 ΔT增加，STORM(T')的性能下降，这与竞争比理论上的下降相符（例如，从 $\Delta T = 10$ 到100，下降了10倍）。相比之下，理论上STORM++ 不受影响。

预言机调用次数、运行时间和内存。图4c显示，对于 STORM(T') 和 STORM++，预言机调用次数会随着预算 k 快速线性增长，而其他算法的调用次数则相对稳定

。类似地，图4d表明，STORM(T')、STORM(T) 和 LMGREEDY 的预言机调用次数会随着用户访问次数 T 线性增长，而 SIEVE++ 和 PREEMPTION 的增长速度则慢得多。值得注意的是，图 4d 还揭示了一种非单调趋势：随着 T 增加，预言机调用次数先上升后下降。这种增长是由于在 $\delta = 25$ 和 $\Delta T' = 45$ 下，对 T 采用了猜测策略，猜测次数 $\mathcal{T}$ 会增加（$\mathcal{T} = \{25, 50\}$ 次用于 $T = 3$，$\mathcal{T} = \{25, 50, 75\}$ 次用于 $T \in [6, 30]$）。调用次数下降则是由于 STORM++ 的不可撤销性——随着向用户提供的子集越来越多，可供交换的物品在 $\mathcal{S}$ 中越来越少，因此后续的预言机调用次数也随之减少。

表2显示，我们提出的算法所需内存远少于LMGREEDY，并且与另外两种基线方法的内存使用量相当。STORM(T')和STORM(T)也明显快于其余算法。

5 结论

我们研究了一种新颖的在线设置，用于流式子模最大化，其中用户会在数据流期间提交按需请求。在此背景下，我们提出了具有按需请求的流式随机子模最大化（*S3MOR*）问题，旨在最大化所选物品的期望覆盖率。我们首先提出了一种内存高效的近似算法，突出了内存使用量与解的质量之间的权衡，但其竞争比可能较差。为解决这一问题，我们提出了一种依赖参数的内存高效近似算法，以在竞争比、内存使用量和时间使用量之间进行权衡。我们的实证评估表明，所提出的算法始终优于基线方法。我们相信，我们的方法将产生积极的社会影响。例如，它可以用于提高向用户推荐的新闻内容的多样性，从而帮助用户接触多种观点并拓展知识。尽管我们的方法优化了主题覆盖，并可用于促进内容多样性，但如果被滥用，例如故意优化偏向两极分化或带偏见内容的次模效用函数，可能会放大现有偏差。然而，这类结果需要对目标函数进行蓄意操纵。在负责任地应用时，我们的方法不会带来此类风险。

致谢与经费披露

本研究得到了ERC高级资助项目REBOUND（834862）、瑞典研究理事会项目ExCLUS（2024-05603），以及由克努特与爱丽丝·瓦伦堡基金会资助的瓦伦堡人工智能、自主系统与软件计划（WASP）的支持。

参考文献

[1] Saeed Alaei、Ali Makhdoumi 和 Azarakhsh Malekian。2021。序列子模函数最大化及其在在线广告中的应用。《管理科学》67, 10 (2021), 6030–6054。[2] Bushra Alhijawi、Arafat Awajan 和 Salam Fraihat。2022。推荐系统目标综述：指标、解决方案、评估方法与新视角。《计算机调查》55, 5 (2022), 1–38。[3] Azin Ashkan、Branislav Kveton、Shlomo Berkovsky 和 Zheng Wen。2015。推荐中的最优贪心多样性。。载于*IJCAI*，第15卷。1742–1748。[4] Ashwinkumar Badanidiyuru、Baharan Mirzasoleiman、Amin Karbasi 和 Andreas Krause。2014。流式子模最大化：即时海量数据摘要。载于第*20*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集。671–680。[5] Niv Buchbinder、Moran Feldman 和 Roy Schwartz。2019。带抢占的在线次模最大化。《*ACM*算法汇刊》15, 3 (2019), 1–31。[6] Gruia Calinescu、Chandra Chekuri、Martin Pal 和 Jan Vondrák。2011。在拟阵约束下最大化单调子模函数。*SIAM*计算杂志40, 6 (2011), 1740–1766。

[7] Amit Chakrabarti 和 Sagar Kale。2015。子模最大化遇上流式处理：匹配、拟阵及更多内容。数学规划154 (2015), 225–247。[8] Chandra Chekuri、Shalmoli Gupta 和 Kent Quanrud。2015。子模函数最大化的流式算法。载于自动机、语言与编程：第*42*届国际学术研讨会，*ICALP 2015*，日本京都，*2015*年*7*月*6*日至*10*日，论文集，第一部分*42*。施普林格，318–330。[9] Marwa El Halabi、Federico Fusco、Ashkan Norouzi-Fard、Jakab Tardos 和 Jakub Tarnawski。2023。拟阵约束下流式子模最大化中的公平性。载于国际机器学习会议。PMLR，9150–9171。[10] Moran Feldman、Amin Karbasi 和 Ehsan Kazemi。2018。少做多得：采用子采样的流式子模最大化。神经信息处理系统进展31 (2018)。[11]Moran Feldman、Paul Liu、Ashkan Norouzi-Fard、Ola Svensson 和 Rico Zenklusen。2021。拟阵约束下的流式子模最大化。*arXiv*预印本 *arXiv:2107.07183* (2021)。[12] Marshall L Fisher、George L Nemhauser 和 Laurence A Wolsey。1978。最大化子模集合函数的近似分析——*II*。施普林格。[13] Satoru Fujishige。2005。子模函数与优化。第58卷。爱思唯尔。[14] Chongming Gao、Shijun Li、Wenqiang Lei、Jiawei Chen、Biao Li、Peng Jiang、Xiangnan He、Jiaxin Mao 和 Tat-Seng Chua。2022。KuaiRec：用于评估推荐系统的全观测数据集及其见解。载于第*31*届*ACM*信息与知识管理国际会议论文集。540–550。[15] Michel Goemans 和 Jan Vondrák。2006。随机覆盖与自适应性。载于拉丁美洲理论信息学研讨会。施普林格，532–543。[16] Daniel Golovin 和 Andreas Krause。2011。自适应子模性：主动学习与随机优化中的理论及应用。《人工智能研究杂志》42 (2011), 427–486。[17] Ryan Gomes 和 Andreas Krause。2010。*ICML*中的数据流预算约束非参数学习。。第1卷。Citeseer，3。[18] Alkis Gotovos、Amin Karbasi 和 Andreas Krause。2015。非单调自适应子模最大化。载于*IJCAI*。1996–2003。[19] Fabrizio Grandoni、Anupam Gupta、Stefano Leonardi、Pauli Miettinen、Piotr Sankowski 和 Mohit Singh。2008。闭着眼睛做集合覆盖。载于*2008*年第*49*届*IEEE*计算机科学基础年度研讨会。IEEE, 347–356。[20] Ran Haba、Ehsan Kazemi、Moran Feldman 和 Amin Karbasi。2020。k-集合系统约束下的流式子模最大化。载于国际机器学习会议。PMLR，3939–3949。[21] Chien-Chung Huang、Naonori Kakimura 和 Yuichi Yoshida。2020。背包约束下最大化单调子模函数的流式算法。《*Algorithmica*》82, 4 (2020), 1006–1032。[22] Chien-Chung Huang、Theophile Thiery 和 Justin Ward。2021。用于拟阵约束下子模最大化的改进多遍流式算法。*arXiv*预印本 *arXiv:2102.09679* (2021)。[23] Ehsan Kazemi、Marko Mitrovic、Morteza Zadimoghaddam、Silvio Lattanzi 和 Amin Karbasi。2019。子模流式处理的全面成果：紧确近似、最小内存与低自适应复杂度。载于国际机器学习会议。PMLR，3311–3320。[24] David Kempe、Jon Kleinberg 和 Éva Tardos。2003。通过社交网络最大化影响力传播。载于第九届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集。137–146。[25] Yehuda Koren、Robert Bell 和 Chris Volinsky。2009。推荐系统的矩阵分解技术。《*Computer*》42, 8 (2009), 30–37。[26] Andreas Krause 和 Daniel Golovin。2014。子模函数最大化。可处理性3, 71-104 (2014), 3。

[27] Jure Leskovec、Andreas Krause、Carlos Guestrin、Christos Faloutsos、Jeanne VanBriesen 和 Natalie Glance。2007。网络中的高性价比疫情检测。载于第*13*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集。420–429。[28] David D Lewis、Yiming Yang、Tony G Rose 和 Fan Li。2004。RCV1：用于文本分类研究的新基准数据集。机器学习研究期刊5，4月（2004），361–397。[29] Yajing Liu、Edwin KP Chong、Ali Pezeshki 和 Zhenliang Zhang。2020。子模优化问题与贪心策略：综述。离散事件动态系统30，3（2020），381–412。[30] Julian McAuley 和 Jure Leskovec。2013。隐因子与隐主题：通过评论文本理解评分维度。载于第*7*届*ACM*推荐系统会议论文集。165–172。[31] Julian McAuley、Jure Leskovec 和 Dan Jurafsky。2012。从多方面评论中学习态度与属性。载于*2012*年*IEEE*第*12*届数据挖掘国际会议。IEEE，1020–1025。[32] Julian John McAuley 和 Jure Leskovec。2013。从业余爱好者到行家：通过在线评论建模用户专业程度的演变。载于第*22*届万维网国际会议论文集。897–908。[33] Anay Mehrotra 和 Nisheeth K Vishnoi。2023。在存在偏差的情况下最大化子模函数以进行推荐。载于*2023*年*ACM Web*会议论文集。3625–3636。[34] Baharan Mirzasoleiman、Ashwinkumar Badanidiyuru 和 Amin Karbasi。2016。快速受约束子模最大化：个性化数据摘要。载于国际机器学习会议。PMLR，1358–1367。[35] Baharan Mirzasoleiman、Ashwinkumar Badanidiyuru、Amin Karbasi、Jan Vondrák 和 Andreas Krause。2015。比惰性贪心更惰性的算法。载于*AAAI*人工智能会议论文集，第29卷。[36] Baharan Mirzasoleiman、Stefanie Jegelka 和 Andreas Krause。2018。流式非单调子模最大化：即时个性化视频摘要。载于*AAAI*人工智能会议论文集，第32卷。[37] Marko Mitrovic、Ehsan Kazemi、Moran Feldman、Andreas Krause 和 Amin Karbasi。2019。自适应序列子模性。神经信息处理系统进展32（2019）。[38] Houssam Nassif、Kemal Oral Cansizlar、Mitchell Goodman 和 SVN Vishwanathan。2018。多样化音乐推荐。*arXiv*预印本 *arXiv:1810.01482*（2018）。[39] George L Nemhauser、Laurence A Wolsey 和 Marshall L Fisher。1978。最大化子模集合函数的近似分析——I。数学规划14（1978），265–294。[40] Naoto Ohsaka 和 Yuichi Yoshida。2015。具有规模约束的单调k-子模函数最大化。神经信息处理系统进展28（2015）。[41] Shaojie Tang 和 Jing Yuan。2020。自适应级联子模最大化。*arXiv*预印本 *arXiv:2007.03592*（2020）。[42] Kai Wei、Rishabh Iyer 和 Jeff Bilmes。2015。数据子集选择与主动学习中的次模性。载于国际机器学习会议。PMLR，1954–1963。[43] Qiong Wu、Yong Liu、Chunyan Miao、Yin Zhao、Lu Guan 和 Haihong Tang。2019。多样化推荐的最新进展。*arXiv*预印本 *arXiv:1905.06589*（2019）。[44] Yisong Yue 和 Carlos Guestrin。2011。线性子模老虎机及其在多样化检索中的应用。神经信息处理系统进展24（2011）。[45] Ghahramani Zoubin。2013。通过子模最大化扩展印度自助餐过程。载于国际机器学习会议。PMLR，1013–1021。

附录

A 预备知识

我们使用 $[j]$表示集合 $\{1,2,\ldots,j\}$，其中 $j \in \mathbb{N}$。集合系统是一个二元组（U，$\mathcal{I}$），其中 U是一个基集，$\mathcal{I}$是 U的一个子集族。

定义 A.1（拟阵）。集合系统（U、$\mathcal{I}$）是拟阵，当且仅当

（i）$\emptyset \in \mathcal{I}$，（ii）对于 $B \in \mathcal{I}$ 和任意 $A \subseteq B$，有 $A \in \mathcal{I}$，并且（iii）如果 $A \in \mathcal{I}$、$B \in \mathcal{I}$ 和 $|A| < |B|$，则存在 $e \in B \setminus A$ 使得 $A \cup \{e\} \in \mathcal{I}$。

定义 A.2（分划拟阵）。拟阵 $\mathcal{M} = (U、\mathcal{I})$ 是分划拟阵，当且仅当存在 U 的一个分区 $\{E_i\}_{i=1}^r$，即对于所有 $i \neq j$，有 $U = \cup_{i=1}^r E_i$ 和 $E_i \cap E_j = \emptyset$，并且存在非负整数 k_1、...、k_r，使得 $\mathcal{I} = \{I \subseteq U : |I \cap E_i| \leq k_i$ 对所有 $i \in [r]\}$ 成立。

非负集合函数 $f\colon\ 2^U \to \mathbb{R}_{\geq 0}$ 称为次模函数，当且仅当对于所有集合 $A \subseteq B \subseteq U$ 和所有元素 $e \in U \setminus B$，都有 $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$。函数 f称为单调函数，当且仅当对于所有 $A \subseteq B$，都有 $f(A) \leq f(B)$。设 U为有限基集，且 $f\colon\ 2^U \to \mathbb{R}_{\geq 0}$为单调次模集合函数。约束次模最大化问题是求解 $\max_{S \in \mathcal{I}} f(S)$，其中 $\mathcal{I} \subseteq 2^U$ 表示可行性约束。

为简化记号，对于任意两个集合 $A \subseteq U$和 $B \subseteq U$以及任意项目 $e \in U$，我们将 $A + B$记作 $A \cup B$，将 $A + e$ 记作 $A \cup \{e\}$，将 $A - B$ 记作 $A \setminus B$，并将 $A - e$ 记作 $A \setminus \{e\}$。我们分别将 $f(A \mid B)$ 和 $f(e \mid B)$ 记作增量收益，其定义为 $f(A \cup B) - f(B)$ 和 $f(A \cup \{e\}) - f(A)$。

B 第 2 节省略的证明

引理 2.1。 $f(\mathcal{S})$ 是一个单调的子模集合函数。

证明。对于任意集合 $\mathcal{S}$，我们可以重写公式（1）。记作 $\mathcal{S} = \cup_{t=1}^T \mathcal{S}^t$，并允许 V_i 的多个副本出现在 $\mathcal{S}$ 中（这些副本分别来自每个 $\mathcal{S}^t$）。注意，$\mathcal{S}$ 是一个多重集，并且

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

令 $g_j(\mathcal{S}) = 1 - \prod_{V_i \in \mathcal{S},\ V_i \ni c_j} (1 - p_i)$。那么 $f(\mathcal{S}) = \sum_{j=1}^{d} g_j(\mathcal{S})$。要证明该引理，只需证明对于所有 $j \in [d]$，g_j 都是单调且次模的。

单调性：对于任意两个集合 $\mathcal{X} \subseteq \mathcal{Y}$ 以及任意 $j \in [d]$，有

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

次模性：设 $\mathcal{X} \subseteq \mathcal{Y}$，并设 $Z \in \mathcal{Y}$ 是一个不属于 $\mathcal{Y}$的元素。那么，对于所有 $j \in [d]$，以下结论成立：

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

□

算法 3： LMGREEDY

输入： 预算 $k \in \mathbb{N}$。

1 $\mathcal{G}^0,\ \cdots,\ \mathcal{G}^T \leftarrow \emptyset,\ \ \mathcal{G} \leftarrow \emptyset,\ \ \mathcal{V}^1 \leftarrow \emptyset,\ \ t = 1$
2 **对于**(V, τ, p) 在数据流中**执行**
3 $\mathcal{V}^t \leftarrow \mathcal{V}^t \cup \{V\}$
4 **如果** $\tau = 1$**则**
5 $\mathcal{N}^t \leftarrow \mathcal{V}^t$
6 **对于** $j = 1$到 k **执行**
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 **输出** $\mathcal{G}^t$
11 $\mathcal{G} = \mathcal{G} \cup \mathcal{G}^t$
12 $\mathcal{V}^{t+1} \leftarrow \mathcal{V}^t$
13 $t \leftarrow t + 1$

定理 2.4。 问题 *2.2* 的算法 *3* 的竞争比为 $1/2$，空间复杂度为 $\Theta(N + kT)$，时间复杂度为 $\mathcal{O}(NTk)$，响应时间复杂度为 $\mathcal{O}(Nk)$。此外， $1/2$ 的竞争比是紧的，即对于问题 *2.2*，任何流式算法都无法在不违反不可撤销约束的情况下实现优于 $1/2$ 的竞争比。

证明。 我们首先证明，算法 3 对问题 2.2 的竞争比为 $1/2$ 。

设 $\mathcal{O}^1, \cdots, \mathcal{O}^T$为最优输出集，并设 $\mathcal{G}^1, \cdots, \mathcal{G}^T$为算法 3 得到的输出集。由于我们将同一项目的每个副本视为不同的项，因此对于任意$i \neq j$、 $\mathcal{O}^i \cap \mathcal{O}^j = \emptyset$ 和 $\mathcal{G}^i \cap \mathcal{G}^j = \emptyset$. 均有。令 $\mathcal{G} = \bigcup_{t=1}^{T} \mathcal{G}^t$ 和 $\mathcal{O} = \bigcup_{t=1}^{T} \mathcal{O}^t$。

我们证明以下结论成立：

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

不等式 (*a*) 和 (*d*) 由次模性成立，(*c*) 由单调性成立，而等式 (*b*) 成立是因为 $\{\mathcal{O}^1, \cdots, \mathcal{O}^T\}$ 构成了 $\mathcal{O}$ 的一个分区。

接下来，我们对公式（2）的最后一项进行界定。我们记 $\mathcal{O}^i = \{O_1^i, \cdots, O_k^i\}$，并记 $\mathcal{G}^i = \{G_1^i, \cdots, G_k^i\}$，其中 G_j^i表示在贪心选择步骤中加入 $\mathcal{G}^i$的第 j 个项目。注意，我们可以假设 $|\mathcal{G}| = |\mathcal{O}| = k$，因为 f 是单调的。于是有

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

其中，不等式（a）由次模性成立，不等式（b）则由贪心算法（算法 3 第 7 行）的设计成立。

结合公式（2）和公式（3），证明得以完成：

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

此外，我们证明了 $1/2$ 对于问题 2.2 的竞争比是紧的。
考虑一个预算为 $k=1$ 的简单对抗性示例，其数据流如下：

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

其中 $V_1=\{c_1,c_2\}$ 且 $V_2=\{c_3,c_4\}$。对于任意算法，如果它选择 $\mathcal{S}^1=\{V_1\}$ 作为第一个要展示的集合，对手就会设置 $V_3=\{c_1,c_2\}$。反之，如果算法选择 $\mathcal{S}^1=\{V_2\}$，对手就会设置 $V_3=\{c_3,c_4\}$。

无论哪种情况，算法都只能覆盖两个主题，而最优解可以覆盖全部四个主题。因此，竞争比至多为 $\frac{2}{4}=\frac{1}{2}$，证明完毕。

□

C 第 3 节省略的证明

在给出第 3 节省略的证明之前，我们先详细描述分析中使用的归约。

我们首先正式定义受分区拟阵约束的流式子模最大化问题（*S2MM*）。我们的定义改编自 Chekuri 等人 [8], 提出的、受 p-匹配拟阵约束的流式子模最大化框架，具体如下：

问题 C.1 (*S2MM*)： 设 $\mathcal{M}=(\mathbf{S},\mathcal{I})$ 为一个分划拟阵，f 为一个子模函数。$\mathbf{S}$ 的元素以数据流形式呈现，我们按照出现顺序对$\mathbf{S}$ 进行排序。S2MM 问题的目标是选择一个物品子集 $\mathcal{A}\in\mathcal{I}$，使 $f(\mathcal{A})$最大化。

我们展示从一个*S3MOR* 实例到一个*S2MM*实例的两种归约方式，具体取决于我们是否知道 T的确切值。正文中介绍的算法 1基于第二种归约，在这种情况下，我们只知道 T的一个上界 T'。

C.1 已知确切访问次数 T 的情况

首先，我们证明，如果已知确切的用户访问次数 T，则*S3MOR* 问题可归约为*S2MM* 问题。具体而言，任何满足不可撤销输出要求 的*S2MM* 流式算法，都可以在保持相同竞争比的情况下解决*S3MOR* 问题。不可撤销输出要求 要求：对于任意分区，一旦其中的所有物品都已出现在数据流中，其输出子集就必须不可撤销地固定。

归约。给定一个*S3MOR* 问题实例，其数据流由

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

给定预算 k，以及公式（1）中定义的函数 f，我们可以构造一个对应的*S2MM* 问题实例。

回忆一下，r_t 表示用户提交第 t 次访问的时间步（其中 $r_0=0$），而 V_i^t表示项目 V_i 的一个副本，其副本标识符为 t。我们用 $\bigsqcup$ 表示数据流的串接。输入流为$\mathbf{S}$ 的*S2MM* 问题实例可按如下方式构造：

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

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

2. 令 $\bar{\mathbf{S}}=\bigsqcup_{t=1}^{T}\bar{\mathbf{S}}_t$ 为子数据流 $\bar{\mathbf{S}}_1,\cdots,\bar{\mathbf{S}}_T$的串接，即 $\bar{\mathbf{S}}=(\bar{\mathbf{S}}_1,\cdots,\bar{\mathbf{S}}_T)$.
3. 令 $\mathcal{V}^t=\{V_i^t\}_{i=1}^{r_t}=\{V_1^t,\ldots,V_{r_t}^t\}$。
4. 按以下方式构造 $\mathcal{I}$：对于任意集合 $X\subseteq\bigcup_{t=1}^{T}\mathcal{V}^t$，如果对所有$1\le t\le T$均有 $|X\cap\mathcal{V}^t|\le k$，则 $X\in\mathcal{I}$。

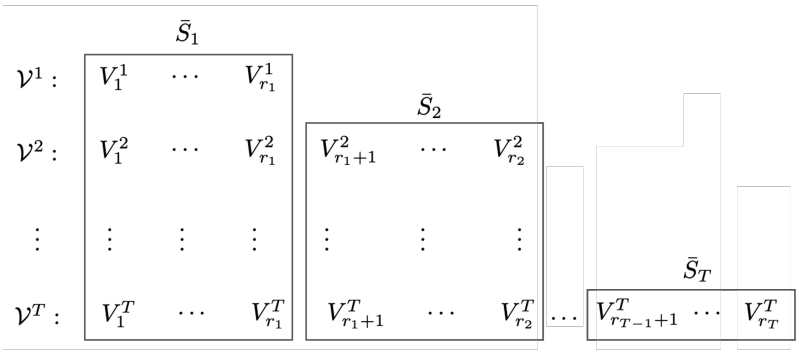


图 5：已知确切访问次数 T 时的归约示意图。*S3MOR* 问题的原始项目数据流为 (V_1，……，V_{r_T})，用户访问发生在时间步r_1，……，r_T。为*S2MM* 问题构造的数据流为 $\bar{\mathbf{S}} = \bigsqcup_{t=1}^{T} \bar{\mathbf{S}}_t$。

5. 令 $p_i^t = p_i$ 表示 p_i 的一个副本，其副本标识符为 $t \in [T]$。定义 $\hat{f}$：$2^{\bigcup_{t=1}^{T} \mathcal{V}^t} \to \mathbb{R}_{\geq 0}$。具体而言，对于任意 $\mathcal{A} \subseteq \bigcup_{t=1}^{T} \mathcal{V}^t$，令 $\mathcal{A}^t = \mathcal{A} \cap \mathcal{V}^t$。然后，$\hat{f}$ 定义如下：

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

$\hat{f}$，*S2MM* 的目标是找到使 $f(\mathcal{A})$ 最大化的 $\mathcal{A} \in \mathcal{I}$。

我们观察到，按上述定义，$\mathcal{M} = (\bar{\mathbf{S}}, \mathcal{I})$ 在构造的数据流$\bar{\mathbf{S}}$上构成一个分划拟阵，其中 $\bar{\mathbf{S}} = \bigcup_{t=1}^{T} \mathcal{V}^t$，且 $\mathcal{V}^i \cap_{i \neq j} \mathcal{V}^j = \emptyset$。根据构造，对于每次访问 $t \in [T]$，分区 $\mathcal{V}^t$ 恰好包含在 *S3MOR* 问题中第 t 次用户访问之前进入系统的所有项目。由于*S2MM* 问题与*S3MOR* 问题具有等价的目标函数和等价的约束，因此我们得出以下结论：对于*S2MM* 问题的任意最优解 $\mathcal{O} \in \mathcal{I}$，其子集序列

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

在每个对应的第 t 次用户访问时，为*S3MOR* 问题生成最优输出。

我们还可以得出结论：任何能为*S2MM* 问题提供 α 竞争比解的流式算法，也能为*S3MOR* 问题提供 α 竞争比解。要使这种等价关系成立，针对*S2MM* 的流式算法必须满足不可撤销输出要求：对于数据流的每个分区 $\mathcal{V}^t$，一旦该分区的最后一个项目到达，属于该分区的解 $\mathcal{A} \cap \mathcal{V}^t$ 就不能再更改。这一要求排除了 Feldman 等人 [11]，提出的算法，因为其解集只有在数据流结束后才会生成。相比之下，Chakrabarti 和 Kale [7]，、Chekuri 等人 [8]，以及 Feldman 等人 [10] 提出的算法则适用。这三种算法都满足这一要求，同时为*S2MM* 问题提供 $1/4$ 竞争比解。

C.2 仅当已知 T' T 的上界时的情形

考虑仅知道真实用户访问次数 T 的上界 T' 的情形。我们证明如下结论：(1) 可以构造一个 *S2MM* 问题的实例，使得该 *S2MM* 问题的任何可行解都包含 *S3MOR* 问题的一个有效解。(2) 任何能够解决*S2MM* 问题且满足不可撤销输出要求 的流式算法，都可以调整为解决 *S3MOR* 问题，但竞争比会有所降低。具体而言，竞争比会按因子 $1/(T' - T + 1)$ 衰减。

与我们在 C.1 节中提出的归约不同，*S2MM* 问题的数据流构造同时依赖于原始数据流$\mathbf{S}$和算法 1 中描述的算法过程。我们将证明：

1. 算法 1 为 *S2MM* 问题生成相应的数据流$\mathbf{S}$

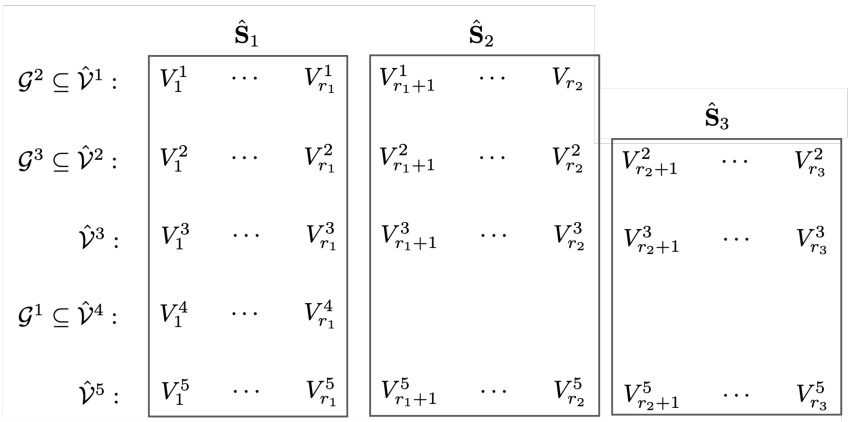


图 6：示例说明仅有真实访问次数 T 的上界 T' 可用时的归约过程。在此示例中，$T = 3$ 和 $T' = 5$。S3MOR 问题的原始项目数据流为（V_1、V_2、...、V_{r_3}），用户访问发生在时间步 r_1、r_2 和 r_3。为 S2MM 问题构造的数据流为 $\hat{\mathbf{S}} = \bigsqcup_{t=1}^{3} \hat{\mathbf{S}}_t$，其中 $\hat{\mathbf{S}}_1 = \bigsqcup_{i=1}^{r_1} \bigsqcup_{a\in[5]}(V_i^a)$、$\hat{\mathbf{S}}_2 = \bigsqcup_{i=r_1+1}^{r_2} \bigsqcup_{a\in\{1,2,3,5\}}(V_i^a)$ 和 $\hat{\mathbf{S}}_3 = \bigsqcup_{i=r_2+1}^{r_3} \bigsqcup_{a\in\{2,3,5\}}(V_i^a)$。算法 1 在 $\hat{\mathbf{S}}$ 中每个项目到达时，都会维护并更新 $\mathcal{A}^t \in \hat{\mathcal{V}}^t$（对所有 $t \in [5]$）。在访问时间步 r_1、r_2 和 r_3，它分别输出 $\mathcal{G}^1 = \mathcal{A}^4$、$\mathcal{G}^2 = \mathcal{A}^1$ 和 $\mathcal{G}^3 = \mathcal{A}^2$。数据流结束时，并集 $\bigcup_{t=1}^{5} \mathcal{A}^t$ 构成了针对输入 $\hat{\mathbf{S}}$ 的 S2MM 问题的 $1/4$-竞争比解。组合解 $\bigcup_{t=1}^{3} \mathcal{G}^t$ 是原始 S3MOR 问题的 $4\times \frac{1}{(5-3+1)}$-竞争比解。

2. 算法 1 维护的候选集 $\{\mathcal{A}^1, \cdots, \mathcal{A}^{T'}\}$ 是 S2MM 问题的一个解；算法 1 的输出 $\{\mathcal{G}^1, \cdots, \mathcal{G}^T\}$ 是 S3MOR 问题的一个解。

根据算法 1 第 13 行，我们可以在 $\{\mathcal{G}^1, \ldots, \mathcal{G}^T\}$ 和 $\{\mathcal{A}^1, \ldots, \mathcal{A}^{T'}\}$ 的索引集之间构造一个双射 π。该映射满足以下等式：

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

令 $\mathcal{T}^0 = \{1, 2, \ldots, T'\}$ 表示所有活跃候选集的初始索引集。对于每个 $t \geq 1$，令 $\mathcal{T}^t$ 表示第 t 次访问之前和第 $(t-1)$ 次访问之后活跃候选集的索引集。活跃索引集的演变如下：

(1) 对所有 $t \in [T]$：$\mathcal{T}^t = \mathcal{T}^{t-1} \setminus \{\pi(t)\}$，以
及 (2) 对所有 $t \in \{T+1, \ldots, T'\}$：$\mathcal{T}^t = \mathcal{T}^T$.

归约。

1. 我们按如下方式为 $\mathbf{S}$ 构造 *S2MM* 的数据流

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

2. 令 $\hat{\mathbf{S}} = \bigsqcup_{t=1}^{T} \hat{\mathbf{S}}_t$ 为 $\hat{\mathbf{S}}_1, \cdots, \hat{\mathbf{S}}_T$ 的拼接。

3. 令 $\rho(t) = \pi^{-1}(t)$，并且

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

4. 按照以下方式构造 $\hat{\mathcal{I}}$：对于任意集合 $X \subseteq \bigcup_{t=1}^{T'} \hat{\mathcal{V}}^t$，如果对所有 $1 \leq t \leq T'$ 都有 $|X \cap \hat{\mathcal{V}}^t| \leq k$，则 $X \in \hat{\mathcal{I}}$。

5. 令 $p_i^t = p_i$ 表示 p_i 的一个副本，其副本标识符为 $t \in [T']$。目标是选择 $\mathcal{A} = \bigcup_{t=1}^{T'} \mathcal{A}^t \in \hat{\mathcal{I}}$，使 $f(\mathcal{A})$ 最大化。子模函数 $\hat{f}$ 定义如下：

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

基于上述构造，我们可以验证，$\mathcal{M} = (\hat{\mathbf{S}}, \hat{\mathcal{I}})$是定义在所构造数据流$\hat{\mathbf{S}}$上的一个分划拟阵实例，其中$\hat{\mathbf{S}} = \bigcup_{t=1}^{T'} \hat{\mathcal{V}}^t$ 且$\hat{\mathcal{V}}^i \cap_{i\neq j} \hat{\mathcal{V}}^j = \emptyset$。由构造可知，$\hat{\mathbf{S}}$ 的分区$\hat{\mathcal{V}}^{\pi(t)}$ 满足：(1) 对每个 $t \in [T]$，$\hat{\mathcal{V}}^{\pi(t)}$ 恰好包含在*S3MOR* 问题中第 t次用户访问之前到达的物品集合；(2) 其余 $T' - T$ 个分区各自包含在最后一次用户访问之前到达的同一物品集合。该设计确保，对所有 $t \in [T]$，每个输出 $\mathcal{G}^t$均满足 $\mathcal{G}^t \subseteq \hat{\mathcal{V}}^{\pi(t)}$ 和 $|\mathcal{G}^t| \leq k$。因此，序列 $\mathcal{G}^1, \ldots, \mathcal{G}^T$构成 *S3MOR*问题的可行输出。

根据算法 1，我们得出结论：对于所有 $t \in [T']$，我们有 $\mathcal{A}^t \subseteq \hat{\mathcal{V}}^t$，且满足 $|\mathcal{A}^t| \leq k$。因此，并集 $\bigcup_{t=1}^{T'} \mathcal{A}^t$ 构成 *S2MM* 问题的一个可行解。

由于算法 1 本质上是将 Chekuri 等人 [8] 的流式算法应用于所构造的数据流 $\hat{\mathbf{S}}$，解 $\bigcup_{t=1}^{T'} \mathcal{A}^t$ 对 *S2MM* 问题实现了 $1/4$-近似保证。

C.3 省略的证明

定理 3.1. 设 T为用户访问次数，T'为给定的上界。算法 *1* 的竞争比为 $\frac{1}{4(T'-T+1)}$，空间复杂度为 $\mathcal{O}(T'k)$，时间复杂度为 $\mathcal{O}(NkT')$，响应时间复杂度为 $\mathcal{O}(T')$。

证明。 沿用第 C.2 节中的记号与结论，我们即可证明定理 3.1。设 $\mathcal{O}^1, \cdots, \mathcal{O}^T$为 *S3MOR* 问题的最优解，并设 $\hat{\mathcal{O}} = \bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t$（其中 $\hat{\mathcal{O}}^t = \hat{\mathcal{O}} \cap \hat{\mathcal{V}}^t$）表示 *S2MM*的最优解。我们有

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

其中，不等式（a）成立，是因为根据算法 1 第 11 行，$\mathcal{A}^{\pi(T)}$ 是依据 $f(\bigcup_{t=1}^{T-1} \mathcal{A}^{\pi(t)})$ 以最大的边际增益被选中的。不等式（b）由次模性得出。不等式（c）成立，是因为如 C.2 节所述，$\bigcup_{t=1}^{T'} \mathcal{A}^t$ 对 *S2MM* 问题达到了 $1/4$-近似保证。最后，不等式（d）可通过反证法证明：如果 $f(\bigcup_{t=1}^{T} \mathcal{O}^t) > f(\bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t)$，那么对于任意 $t \in [T]$、$\mathcal{A}^t = \mathcal{O}^t$，无论选择 $\mathcal{A}^{T+1}$、$\cdots$、$\mathcal{A}^T$为何，都存在一个 *S2MM* 问题的可行解，使得 $f(\bigcup_{t=1}^{T'} \mathcal{A}^t) \geq f(\bigcup_{t=1}^{T} \mathcal{O}^t) > f(\bigcup_{t=1}^{T'} \hat{\mathcal{O}}^t)$，这与 $\mathcal{O}$ 是最优解相矛盾。

复杂度分析。 空间复杂度为 $\mathcal{O}(T'k)$，因为 STORM 只需维护T' 个大小为 k 的候选集。每当一个项目到达时，STORM 会进行 $T'k$ 次预言机调用，因此

时间复杂度为 $\mathcal{O}(NT'k)$。当用户提交访问请求时，STORM 会从至多 T' 个分区中选择一个并输出，因此响应时间复杂度为 $\mathcal{O}(T')$。

□

我们结果的紧性。请注意，公式 (7) 中的分析是紧的。下面我们将证明，对于任意的 T，都可以构造一个数据流，使得 STORM 的表现不可能优于 $\frac{1}{T'-T+1}$；这与定理 3.1 中的分析仅相差一个常数因子。

为说明这一点，考虑一个预算为 $k=1$ 的 *S3MOR*问题实例：用户在数据流末尾访问一次，且我们估计的用户访问次数为 $T'>1$。数据流构造如下：

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

STORM(T') 算法输出前 T' 个物品中的一个，获得的期望主题覆盖值为 1；而最优解是选择最后一个物品，获得期望覆盖率 T'。在此实例中，竞争比为 $\frac{1}{T'}$，而算法 1 的竞争比为 $\frac{1}{4(T'-T+1)} = \frac{1}{4T'}$。

注意，对于任意可能的用户访问次数 T和 $T'>T$，我们都可以构造一个数据流，使得 STORM 的表现不会优于 $\frac{1}{T'-T+1}$。例如，当 ~~$T=2$~~时，我们可以按如下方式构造一个子数据流 $\mathbf{S}_2$

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

将 $\mathbf{S}_1$ 和 $\mathbf{S}_2$, 拼接起来，我们得到一个新的数据流，其中用户访问 2 次。在预算$k=1$下，STORM(T') 的期望覆盖率为 3，而最优期望覆盖率为 3($T'-1$)，因此竞争比为 $\frac{1}{T'-1}$，这同样比 $\frac{1}{4(T'-T+1)} = \frac{1}{T'-1}$差一个 $1/4$因子。我们可以重复这一过程，使得对于任意 T，都能构造一个数据流，使其竞争比等于 $\frac{1}{T'-T+1}$。

引理 C.2. 设 T 为用户访问次数，T' 为给定的上界。算法 *2* 将 T生成一组猜测值，如下：$\mathcal{P}=\{\delta i \mid i\in[\lceil T'/\delta\rceil]\}$. 设 T^* 为 $\mathcal{P}$中大于或等于 T的最小整数。设 $\mathcal{G}^1,\cdots,\mathcal{G}^T$为 STORM($T^*$)的输出，并设 $\mathcal{B}^1,\cdots,\mathcal{B}^T$为算法 *2* 的输出，则有

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

证明。 我们通过归纳法证明公式（8）。

当 $T=1$ 时，根据算法 2 的第 7 行，选择边际增益最大的 $\mathcal{B}^1$，因此有 $f(\mathcal{B}^1) \geq f(\mathcal{G}^1)$。

归纳法的基例从 $T=2$ 开始。其形式为

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

其中，不等式 (a) 成立是因为 $f(\mathcal{B}^1) \geq f(\mathcal{G}^1)$ 且 $f(\mathcal{B}^2 \mid \mathcal{B}^1) > f(\mathcal{G}^2 \mid \mathcal{B}^1)$。不等式 (b) 根据次模性成立。

接下来，我们证明归纳步骤。假设对于 $T=j$，有

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

我们可以用 ϕ_j 表示公式（9）中的第二项，即 $f(\bigcup_{t=1}^{j}\mathcal{B}^t) \geq f(\bigcup_{t=1}^{j}\mathcal{G}^t) + \phi_j$，进而可以证明对于 $T = j+1$，有

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

Her因此，根据归纳假设，不等式（a）成立；根据次模性，不等式（b）成立。 s
co归纳过程由此完成。 □

引理 C.3。 设 T 为用户访问次数， T' 为给定的上界。算法 *2* 将 T的一组猜测生成为 $\mathcal{P} = \{\delta i \mid i \in [\lceil T'/\delta \rceil]\}$. 设 T^* 为 $\mathcal{P}$中大于或等于 T的最小整数。设 $\mathcal{G}^1$、 $\cdots$、 $\mathcal{G}^T$为 STORM(T^*) 的输出，设 $\mathcal{B}^1$、 $\cdots$、 $\mathcal{B}^T$为算法 *2* 的输出。则有

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

证明。 引理 C.3 是引理 C.2 的直接结果，因为我们可以证明

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

为证明上述不等式成立，我们接下来证明

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

事实上，它

$$\sum_{t=1}^{T-1}\Big[-f(\bigcup_{s=1}^{t}\mathcal{B}^s \mid \bigcup_{s=1}^{t+1}\mathcal{G}^s)+f(\bigcup_{s=1}^{t}\mathcal{B}^s \mid \bigcup_{s=1}^{t}\mathcal{G}^s)\Big] \tag{11}$$

$$\begin{aligned}
&=\sum_{t=1}^{T-1}\Big[f(\bigcup_{s=1}^{t}\mathcal{B}^s \mid \bigcup_{s=1}^{t}\mathcal{G}^s)-f(\bigcup_{s=1}^{t}\mathcal{B}^s \mid \bigcup_{s=1}^{t+1}\mathcal{G}^s)\Big]\\
&=f(\mathcal{B}^1 \mid \mathcal{G}^1)+\sum_{t=1}^{T-2}\left[f(\bigcup_{s=1}^{t+1}\mathcal{B}^s \mid \bigcup_{s=1}^{t+1}\mathcal{G}^s)-f(\bigcup_{s=1}^{t}\mathcal{B}^s \mid \bigcup_{s=1}^{t+1}\mathcal{G}^s)\right]-f(\bigcup_{s=1}^{T-1}\mathcal{B}^s \mid \bigcup_{s=1}^{T}\mathcal{G}^s)\\
&\le f(\mathcal{B}^1)+\sum_{t=1}^{T-2}\left[f(\bigcup_{s=1}^{t+1}\mathcal{B}^s \mid \bigcup_{s=1}^{t+1}\mathcal{G}^s)-f(\bigcup_{s=1}^{t}\mathcal{B}^s \mid \bigcup_{s=1}^{t+1}\mathcal{G}^s)\right]\\
&=f(\mathcal{B}^1)+\sum_{t=1}^{T-2}\left[f(\bigcup_{s=1}^{t+1}\mathcal{B}^s+\bigcup_{s=1}^{t+1}\mathcal{G}^s)-f(\bigcup_{s=1}^{t+1}\mathcal{G}^s)-f(\bigcup_{s=1}^{t}\mathcal{B}^s+\bigcup_{s=1}^{t+1}\mathcal{G}^s)+f(\bigcup_{s=1}^{t+1}\mathcal{G}^s)\right]\\
&=f(\mathcal{B}^1)+\sum_{t=1}^{T-2}f(\mathcal{B}^{t+1} \mid \bigcup_{s=1}^{t}\mathcal{B}^s+\bigcup_{s=1}^{t+1}\mathcal{G}^s)\le f(\mathcal{B}^1)+\sum_{t=1}^{T-2}f(\mathcal{B}^{t+1} \mid \bigcup_{s=1}^{t}\mathcal{B}^s)\\
&=f(\bigcup_{t=1}^{T-1}\mathcal{B}^t)\le f(\bigcup_{t=1}^{T}\mathcal{B}^t)
\end{aligned} \tag{12}$$

结合公式（12）与公式（8），我们可以得到公式（10）。

□

定理 3.2。 算法 *2* 的竞争比为 $1/(8\delta)$，空间复杂度为 $\mathcal{O}(T'^2k/\delta)$，时间复杂度为 $\mathcal{O}(NkT'^2/\delta)$，响应时间复杂度为 $\mathcal{O}(T')$。

证明。 令 T^* 为 $\mathcal{P}=\{\delta i \mid i\in[\lceil T'/\delta\rceil]\}$ 中大于或等于 T 的最小整数。令 $\mathcal{G}^1$、$\cdots$、$\mathcal{G}^T$ 为 STORM(T^*) 的输出
注意到 $T^*\le T+\delta-1$，否则我们可以将 T^* 设为 $T^*-\delta$，且它仍然是 T 的上界。根据定理 3.1，有

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

结合公式（13）与引理 C.3，我们可以得出

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

由于 STORM++ 维护了 STORM 的 $\lceil T'/\delta\rceil$ 个副本，因此每个 STORM 副本的时间复杂度为 $\mathcal{O}(NkT')$，空间复杂度为 $\mathcal{O}(T'k)$；STORM++ 的空间复杂度为 $\mathcal{O}(T'^2k/\delta)$，时间复杂度为 $\mathcal{O}(NkT'^2/\delta)$。当用户提交访问请求时，每个 STORM 副本从最多 T' 个分区中选择一个作为候选结果集，然后 STORM++ 从 T'/δ 个候选结果集中选择最佳集合作为输出。因此，STORM++ 的响应时间为 $\mathcal{O}(T')$。

□

D 第 4 节的补充内容

D.1 数据集与实验设置

我们使用以下六个数据集：

• **KuaiRec** [14]：来自视频分享移动应用的推荐日志，包含每个视频的类别和观看比例等元数据。我们将观看比例（即用户观看时长除以视频时长）作为用户评分。该数据集包含 100 个类别中的 7043 个视频。

• **Anime**:[3] 一个评分范围为 $[1, 10]$, 的动画评分数据集，包含 43 种类型的 7745 个动画项目。

• **Beer**:[4] 该数据集包含 BeerAdvocate 评论 [32, 31], 以及每种啤酒的类别属性。剔除评论数少于 10 条的啤酒和评论者后，最终数据集包含 70 个类别的 9000 种啤酒。

• **Yahoo**:[5] 一个音乐评分数据集，评分范围为 $[1, 5]$。剔除评分少于 20 条的用户后，该数据集包含 58 种类型的 136 736 首歌曲。

• **RCV1** [28]：一个多标签数据集，其中每个项目都是 Reuters, Ltd. 的一篇供研究使用的新闻报道。我们抽样选取了 476 个独特标签下的 462225 篇报道。

• **Amazon** [30]：一个多标签数据集，包含 3 750 个独特标签下的 1 117 006 个 Amazon 商品。该数据集是公开数据集的一个子集。

实验设置 对于 RCV1 和亚马逊数据集，我们在 $[0, 0.2]$ 范围内均匀随机生成各项目的点击概率。对于四个物品评分数据集，我们通过使用矩阵分解对用户-物品评分矩阵进行低秩补全，来估计点击概率 [25]。由此可得到每个用户 w_u u 和每个项目 v_m m的潜在特征向量，其中内积 $w_u^\top v_m$ 近似表示用户 u对项目 m的评分。随后，我们对预测评分进行标准的最小-最大归一化，并将结果线性缩放至 $[0, 0.5]$ 范围，从而得到点击概率。对于 KuaiRec 数据集，我们将潜在特征维度设为 15；对于其余三个评分数据集，则设为 20。现实场景中的点击概率范围可能有所不同，范围较大时可能更快收敛至最大期望覆盖值；但在实验中，我们有意将概率设得较小，以便在更广泛的参数设置下观察性能变化。

实现为了提高 LMGREEDY 算法（见算法 3）的计算效率，我们采用 Mirzasoleiman 等人提出的 STOCHASTIC-GREEDY 方法 [35] 实现第 6–9 行。对于 STORM 和 STORM++ 的实现，我们采用 Feldman 等人提出的采样技术 [10]。具体而言，将每个项目（可能替换原有项目）添加到每个候选集时，我们以 2/3 的概率忽略该项目。

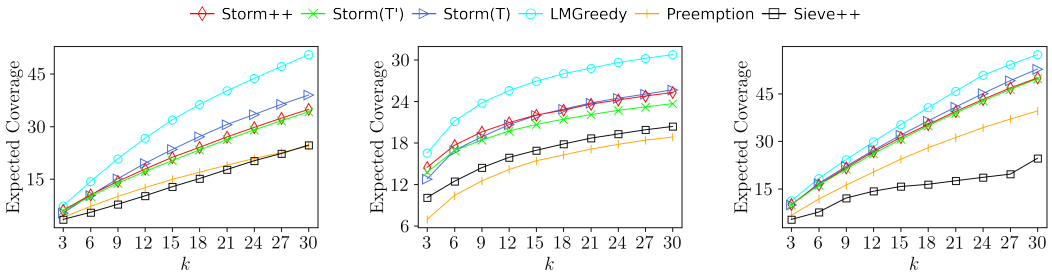
D.2 省略的结果

本节将介绍更多实验结果和分析。具体而言，我们展示了期望覆盖率如何随预算 k 的变化而变化（见图7），以及如何随用户访问次数 T 的变化而变化（见图8）。当 k 和 T 变化时，期望覆盖率的标准差分别见图10和图11。最后，我们展示了 δ 和 ΔT 的变化对期望覆盖率的影响（见图9）。

为什么 STORM++ 和 STORM(T') 优于 STORM(T)? 我们给出一个简单示例，其中 STORM(T') 优于 STORM(T)。考虑输入流 **S**=(($\{1\}$，1，0)，($\{2\}$，1，0)，($\{3, 4\}$，0。9，1)，($\{5, 6\}$，1，1))，预算为 $k = 1$，实际访问次数为$T = 2$。算法 STORM(2) 在第一次访问时选择 $\{1\}$ 或 $\{2\}$，第二次访问时选择 $\{5, 6\}$，达到 3 的期望覆盖率。
如果我们高估访问次数并设定 $T' = 3$，STORM(3) 则会先选择 $\{3, 4\}$，再选择 $\{5, 6\}$，从而得到 3.8 的期望覆盖率，高于准确知道访问次数时的结果。同样，STORM++ 在 $T' = 3$ 和 $\delta = 3$ 的情况下与 STORM(3) 相同，达到 3.8 的覆盖率。

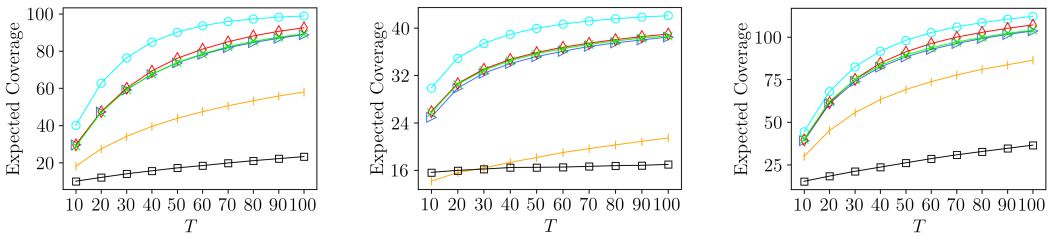
这个例子表明，尽管 STORM(T) 具有最强的理论保证，但 STORM(T') 和 STORM++ 在实证结果上可能优于它，这解释了我们在实验中观察到的趋势。

[3]https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database[4]
https://cseweb.ucsd.edu/~jmcauley/datasets.html[5]
https://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67



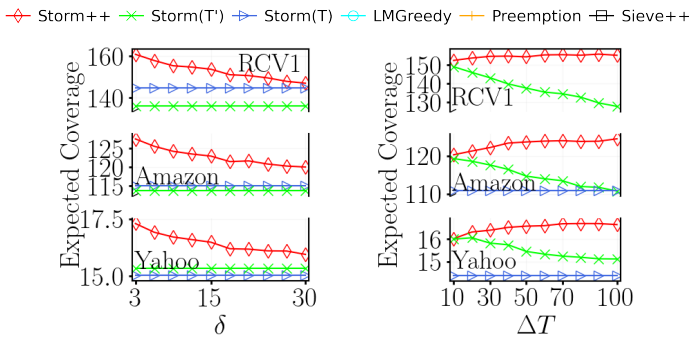
(a) KuaiRec (b) Anime (c) Beer

图7：三个较小数据集上的期望覆盖率随预算 k 变化的实证情况。参数 $T = 5$、$\delta = 25$ 和 $\Delta T = 45$ 固定不变



(a) KuaiRec (b) Anime (c) Beer

图8：三个较小数据集上的期望覆盖率随访问次数 T 变化的经验结果。参数 $k = 10$、$\delta = 10$ 和 $\Delta T = 10$ 固定。



(a) (b)

图9： ΔT 和 δ 对三个较大数据集上的期望覆盖率的影响。在 (a) 和 (b) 中，我们固定 $k = 5$ 和 $T = 10$。改变 δ 时，我们固定 $\Delta T = 60$；改变 ΔT 时，我们固定 $\delta = 10$。

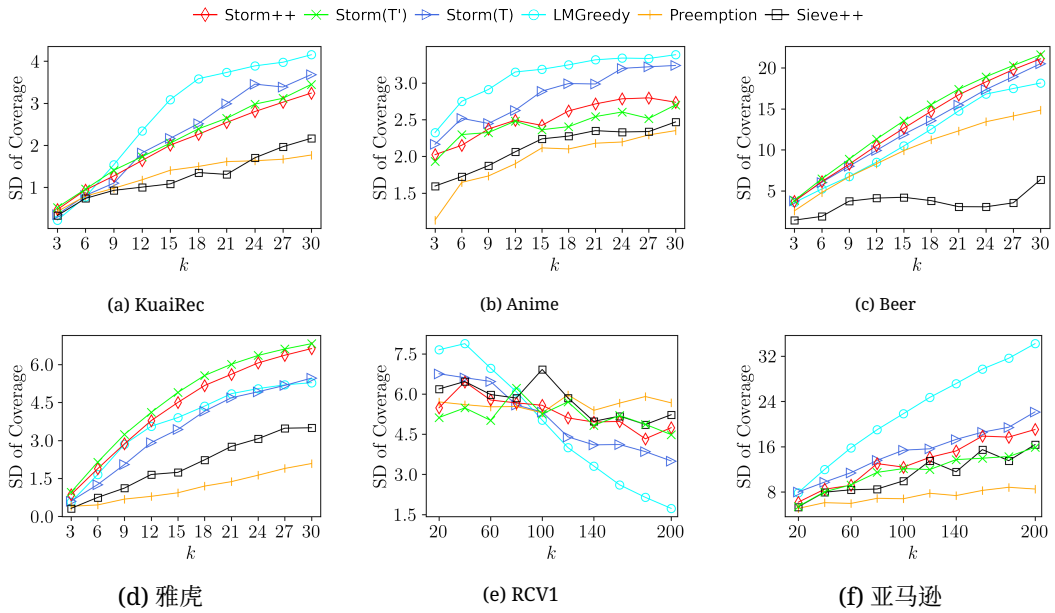


图10：所有数据集上期望覆盖率的标准差随预算变化的实证结果。参数 $T = 5$、$\delta = 25$ 和 $\Delta T = 45$ 固定。

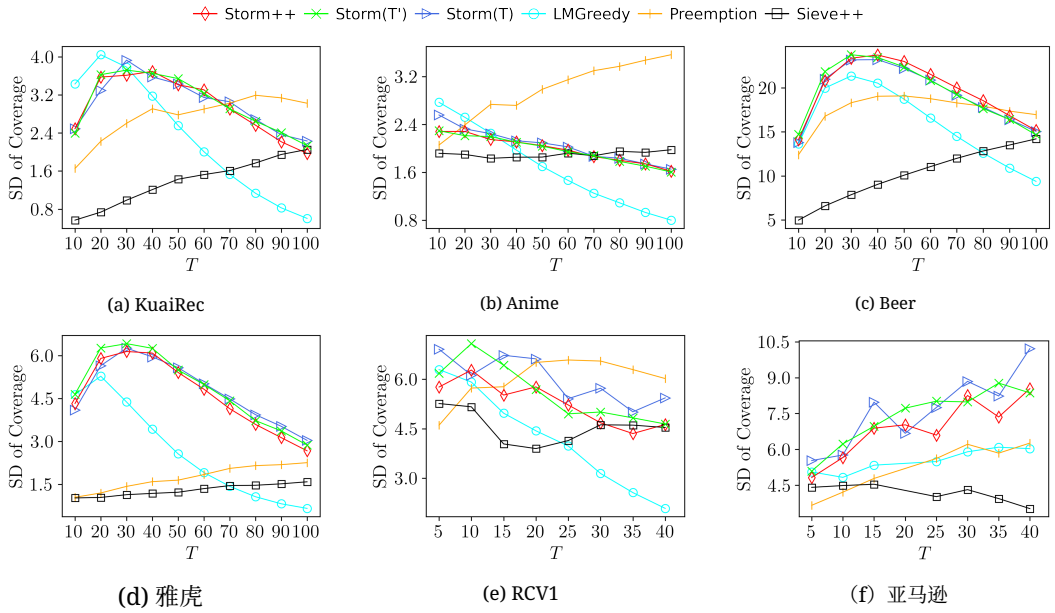


图11：所有数据集上，期望覆盖率的标准差随用户访问次数 T 变化的经验变化情况。参数 $k = 10$、$\delta = 10$ 和 $\Delta T = 10$ 固定。