Improved Streaming Algorithms for Maximizing Monotone Submodular Functions Under a Knapsack Constraint

Chien-Chung Huang[1] and Naonori Kakimura[2(✉)]

[1] CNRS, École Normale Supérieure, Paris, France
villars@gmail.com
[2] Keio University, Yokohama, Japan
kakimura@math.keio.ac.jp

Abstract. In this paper, we consider the problem of maximizing a monotone submodular function subject to a knapsack constraint in the streaming setting. In particular, the elements arrive sequentially and at any point of time, the algorithm has access only to a small fraction of the data stored in primary memory. For this problem, we propose a $(0.4-\varepsilon)$-approximation algorithm requiring only a single pass through the data. This improves on the currently best $(0.363-\varepsilon)$-approximation algorithm. The required memory space depends only on the size of the knapsack capacity and ε.

Keywords: Submodular functions · Streaming algorithm · Approximation algorithm

1 Introduction

A set function $f : 2^E \to \mathbb{R}_+$ on a ground set E is *submodular* if it satisfies the *diminishing marginal return property*, i.e., for any subsets $S \subseteq T \subsetneq E$ and $e \in E \setminus T$,

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

A set function is *monotone* if $f(S) \leq f(T)$ for any $S \subseteq T$. Submodular functions play a fundamental role in combinatorial optimization, as they capture rank functions of matroids, edge cuts of graphs, and set coverage, just to name a few examples. Besides their theoretical interests, submodular functions have attracted much attention from the machine learning community because they can model various practical problems such as online advertising [1,24,35], sensor location [25], text summarization [30,31], and maximum entropy sampling [28].

Many of the aforementioned applications can be formulated as the maximization of a monotone submodular function under a knapsack constraint. In this problem, we are given a monotone submodular function $f : 2^E \to \mathbb{R}_+$, a size

Supported by JSPS KAKENHI Grant Numbers JP17K00028 and JP18H05291.

Z. Friggstad et al. (Eds.): WADS 2019, LNCS 11646, pp. 438–451, 2019.
https://doi.org/10.1007/978-3-030-24766-9_32

function $c : E \to \mathbb{N}$, and an integer $K \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of positive integers. The problem is defined as

$$\text{maximize } f(S) \quad \text{subject to } c(S) \leq K, \quad S \subseteq E, \tag{1}$$

where we denote $c(S) = \sum_{e \in S} c(e)$ for a subset $S \subseteq E$. Note that, when $c(e) = 1$ for every item $e \in E$, the constraint coincides with a cardinality constraint. Throughout this paper, we assume that every item $e \in E$ satisfies $c(e) \leq K$ as otherwise we can simply discard it.

The problem of maximizing a monotone submodular function under a knapsack or a cardinality constraint is classical and well-studied [20,37]. The problem is known to be NP-hard but can be approximated within the factor of $1-e^{-1}$ (or $1 - e^{-1} - \varepsilon$); see e.g., [3,15,21,26,36,38].

In some applications, the amount of input data is much larger than the main memory capacity of individual computers. In such a case, we need to process data in a *streaming* fashion (see e.g., [32]). That is, we consider the situation where each item in the ground set E arrives sequentially, and we are allowed to keep only a small number of the items in memory at any point. This setting effectively rules out most of the techniques in the literature, as they typically require random access to the data. In this work, we assume that the function oracle of f is available at any point of the process. Such an assumption is standard in the submodular function literature and in the context of streaming setting [2,13,39].

Our main contribution is to propose a single-pass $(2/5 - \varepsilon)$-approximation algorithm for the problem (1), which improves on the previous work [23,39] (see Table 1). The space complexity is independent of the number of items in E.

Table 1. The knapsack-constrained problem. The algorithms [16,36] are not for the streaming setting. See also [15,26].

	Approx. ratio	#passes	Space	Running time
Ours	$2/5 - \varepsilon$	1	$O\left(K\varepsilon^{-4}\log^4 K\right)$	$O\left(n\varepsilon^{-4}\log^4 K\right)$
Huang *et al.* [23]	$4/11 - \varepsilon$	1	$O\left(K\varepsilon^{-4}\log^4 K\right)$	$O\left(n\varepsilon^{-4}\log^4 K\right)$
Yu *et al.* [39]	$1/3 - \varepsilon$	1	$O\left(K\varepsilon^{-1}\log K\right)$	$O\left(n\varepsilon^{-1}\log K\right)$
Huang *et al.* [23]	$2/5 - \varepsilon$	3	$O\left(K\varepsilon^{-4}\log^4 K\right)$	$O\left(n\varepsilon^{-4}\log^4 K\right)$
Huang-Kakimura [22]	$1/2 - \varepsilon$	$O\left(\varepsilon^{-1}\right)$	$O\left(K\varepsilon^{-7}\log^2 K\right)$	$O\left(n\varepsilon^{-8}\log^2 K\right)$
Ene and Nguyễn [16]	$1 - e^{-1} - \varepsilon$	—	—	$O\left((1/\varepsilon)^{O(1/\varepsilon^4)} n \log n\right)$
Sviridenko [36]	$1 - e^{-1}$	—	—	$O\left(Kn^4\right)$

Theorem 1. *There exists a single-pass streaming* $(2/5-\varepsilon)$*-approximation algorithm for the problem* (1) *requiring* $O\left(K\varepsilon^{-4}\log^4 K\right)$ *space.*

Our Technique. Let us first describe approximation algorithms for the knapsack-constrained problem (1) in the offline setting. The simplest algorithm is a greedy algorithm, that repeatedly takes an item with maximum marginal return. The greedy algorithm admits a $(1 - 1/\sqrt{e})$-approximation, together with taking one item with the maximum return, although it requires to read all the items K times. Sviridenko [36] showed that, by applying the greedy algorithm from each set of three items, we can find a $(1 - 1/e)$-approximate solution. Recently, such partial enumeration is replaced by a more sophisticated multi-stage guessing strategies (where fractional items are added based on the technique of multilinear extension) to improve the running time in nearly linear time [16]. However, all of them require large space and/or a large number of passes to implement.

In the streaming setting, Badanidiyuru *et al.* [2] proposed a single-pass thresholding algorithm that achieves a $(0.5 - \varepsilon)$-approximation for the cardinality-constrained problem. The algorithm just takes an arriving item e when the marginal return exceeds a threshold and the feasibility is maintained. However, this strategy gives us only a $(1/3-\varepsilon)$-approximation for the knapsack-constrained problem. This drop in approximation ratio comes from the fact that, while we can freely add an item as long as our current set is of size less than K for the cardinality constraint, we cannot take a new item if its addition exceeds the capacity of the knapsack.

To overcome this issue, in [23] a branching technique is introduced, where one stops at some point of the thresholding algorithm and use a different strategy to collect subsequent items. The ratio of this branching algorithm depends on the size of the largest item o_1 in the optimal solution; when o_1 is overly large, other strategies must be employed. Overall, the proposed approach of [23] gives a $(4/11 - \varepsilon)$-approximation.

How does one improve the ratio further when $c(o_1)$ is large? One can certainly guess the size $c(o_1)$ and the f-value $f(\{o_1\})$ beforehand and in the stream pick the item of similar size and f-value. The difficulty lies in how to pick such an item that, *together with the rest of the optimal solution (excluding o_1)*, guarantees a decent f-value. Namely, we need a good substitute of o_1. In [23], a single-pass procedure, called PickOneItem, is designed to find such an item (see Sect. 2 for details). Once equipped with such an item, it is not difficult to collect other items so as to improve the approximation ratio to $2/5-\varepsilon$. The down-side of this approach is that one needs multiple passes.

In this paper, we introduce new techniques to achieve the same ratio *without* the need to waste a pass to collect a good substitute of o_1. Depending on the relative size of o_1 and o_2 (second largest item in the optimal solution), we combine PickOneItem with the thresholding algorithm in two different ways. The first one is to perform both of them *dynamically*, that is, each time we find a candidate e for an approximation of o_1, we perform the thresholding algorithm starting from e with the current set. In contrast, the other runs both of them in a *parallel* way; we perform the thresholding algorithm and PickOneItem independently for some subset of items, and combine their results in the end. The details of these algorithms are described in Sects. 3.2 and 3.3, respectively.

Related Work. Maximizing a monotone submodular function subject to various constraints is a subject that has been extensively studied in the literature. We do not attempt to give a complete survey here and just highlight the most relevant results. Besides a knapsack constraint or a cardinality constraint mentioned above, the problem has also been studied under (multiple) matroid constraint(s), p-system constraint, multiple knapsack constraints. See [9,11,12,15,19,26,29] and the rences therein.

In the streaming setting, single-pass algorithms have been proposed for the problem with matroid constraints [10,18] and knapsack constraint [23,39], and without monotonicity [13,34]. *Multi-pass streaming algorithms*, where we are allowed to read a stream of the input multiple times, have also been studied [3,10,22,23]. In particular, Chakrabarti and Kale [10] gave an $O(\varepsilon^{-3})$-pass streaming algorithms for a generalization of the maximum matching problem and the submodular maximization problem with cardinality constraint. Huang and Kakimura [22] designed an $O(\varepsilon^{-1})$-pass streaming algorithm with approximation guarantee $1/2 - \varepsilon$ for the knapsack-constrained problem. Other than the streaming setting, recent applications of submodular function maximization to large data sets have motivated new directions of research on other computational models including parallel computation model such as the MapReduce model [6,7,27] and the adaptivity analysis [4,5,14,17].

The maximum coverage problem is a special case of monotone submodular maximization under a cardinality constraint where the function is a set-covering function. For the special case, McGregor and Vu [33] and Batani *et al.* [8] gave a $(1 - e^{-1} - \varepsilon)$-approximation algorithm in the multi-pass streaming setting.

2 Preliminaries

For a subset $S \subseteq E$ and an element $e \in E$, we use the shorthand $S + e$ and $S - e$ to stand for $S \cup \{e\}$ and $S \setminus \{e\}$, respectively. For a function $f : 2^E \to \mathbb{R}_+$, we also use the shorthand $f(e)$ to stand for $f(\{e\})$. The *marginal return* of adding $e \in E$ with respect to $S \subseteq E$ is defined as $f(e \mid S) = f(S + e) - f(S)$. Thus the submodularity means that $f(e \mid S) \geq f(e \mid T)$ for any subsets $S \subseteq T \subsetneq E$ and $e \in E \setminus T$.

In the rest of the paper, let $\mathcal{I} = (f, c, K, E)$ be an input instance of the problem (1). Let $\mathrm{OPT} = \{o_1, \ldots, o_\ell\}$ denote an optimal solution with $c(o_1) \geq c(o_2) \geq \cdots \geq c(o_\ell)$. We denote $r_i = c(o_i)/K$ for $i = 1, 2, \ldots, \ell$. Let v be an approximated value of $f(\mathrm{OPT})$ such that $v \leq f(\mathrm{OPT}) \leq (1 + \varepsilon)v$.

In the following sections, we review the previous results: the thresholding algorithm and the procedure PickOneItem.

2.1 Thresholding Algorithms

In this section, we present a thresholding algorithm with a single pass [2,23,39]. The algorithm just takes an arriving item e when the marginal return exceeds a threshold. That is, when a new item e arrives, we decide to add e to our current

set S if $c(S+e) \leq K$ and $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$, where α is a parameter. The performance is due to the following fact, which follows from submodularity.

Lemma 1. *Let $S = \{e_1, e_2, \ldots, e_s\}$. Suppose that $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq \alpha \frac{c(e_i)}{K} v$ for each $i = 1, \ldots, s$. Then it holds that $f(S) \geq \alpha \frac{c(S)}{K} v$.*

By setting $\alpha = 1/2$, the algorithm finds a set S such that $f(S) \geq v/2$ with a single pass for the cardinality-constrained problem [2]. Setting $\alpha = 2/3$, together with taking a singleton with maximum return in parallel, we can find a set S such that $f(S) \geq v/3$ with a single pass [23].

2.2 Guessing the Large Item

We here consider a procedure to approximate the largest item o_1 in OPT. It is difficult to correctly identify o_1 among the items in E, but we can nonetheless find a reasonable approximation of it by a single pass. This procedure is used to design multi-pass streaming algorithms [22,23]. Recall that we are given an approximated value v of $f(\mathrm{OPT})$ such that $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$.

We first present the following fact.

Lemma 2 ([23]). *Let $E_1 \subseteq E$ such that $e^* \in E_1 \cap \mathrm{OPT}$. Suppose that θ satisfies $\theta v/(1+\varepsilon) \leq f(e^*) \leq \theta v$. For a number t with $t > \frac{1}{\theta} - 2$, define*

$$\lambda = 2\left(\frac{\theta}{t+1} - \frac{1}{(t+1)(t+2)}\right). \tag{2}$$

Suppose that a set $X = \{e_1, e_2, \ldots, e_x\} \subseteq E_1$ satisfies that $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq (\theta - \lambda(i-1))v$ for each $i = 1, \ldots, x$. Then the following holds:

(i) If $x = t+1$, then at least one item $e \in X$ guarantees that $f(\mathrm{OPT} - e^* + e) \geq \Gamma(\theta)v - O(\varepsilon)v$.
(ii) If $x < t+1$ and $f(e^* \mid X) < (\theta - \lambda x)v$, then at least one item $e \in X$ satisfies $f(\mathrm{OPT} - e^* + e) \geq \Gamma(\theta)v - O(\varepsilon)v$.

Here Γ is the function defined by

$$\Gamma(\theta) = \frac{t(t+3)}{(t+1)(t+2)} - \frac{t-1}{t+1}\theta. \tag{3}$$

This lemma suggests the following procedure to approximate o_1, which we call PickOneItem. Suppose that we are given approximations $\underline{r}_1, \overline{r}_1$ of r_1 such that $\underline{r}_1 \leq r_1 \leq \overline{r}_1$ and $\overline{r}_1 \leq (1+\varepsilon)\underline{r}_1$. Define $E_1 = \{e \in E \mid \underline{r}_1 K \leq c(e) \leq \overline{r}_1 K, \theta v/(1+\varepsilon) \leq f(e) \leq \theta v\}$. Then we see that $o_1 \in E_1$. In a single pass, starting from $X = \emptyset$, we decide to add an item $e \in E_1$ to X if $f(e \mid X) \geq (\theta - \lambda|X|)v$. We stop this decision when $|X| = t+1$. Then, in each step, X always satisfies the assumption in Lemma 2, that is, $X = \{e_1, e_2, \ldots, e_x\} \subseteq E_1$ satisfies that $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq (\theta - \lambda(i-1))v$ for each $i = 1, \ldots, x$.

We claim that the output X contains an item $e \in X$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$. Indeed, we consider the situation just before o_1 arrives. If the current set X has size $t+1$, then Lemma 2 (i) implies that there exists $e \in X$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$. If X has size less than $t+1$, then either o_1 is put in X, or there exists $e \in X$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$ by Lemma 2 (ii). Hence, in any case, at least one item $e \in X$ guarantees that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$.

By choosing an optimal value t for a given θ, we can obtain $\Gamma(\theta) \geq 2/3$. More specifically, we have the following theorem.

Theorem 2 ([23]). *Let $E_1 \subseteq E$ such that $e^* \in E_1 \cap \mathrm{OPT}$. Suppose that θ satisfies $\theta v/(1+\varepsilon) \leq f(e^*) \leq \theta v$. Define t to be*

$$t = \begin{cases} 1 & \text{if} \quad \theta \geq \frac{1}{2} \\ 2 & \text{if} \quad \frac{1}{2} \geq \theta \geq \frac{2}{5} \\ 3 & \text{if} \quad \frac{2}{5} \geq \theta \geq 0. \end{cases} \tag{4}$$

Then, with a single pass and $O(1)$ space, we can find a set $X \subseteq E_1$ such that there exists $e \in X$ such that $f(\mathrm{OPT} - e^* + e) \geq \Gamma(\theta)v - O(\varepsilon)v$, where

$$\Gamma(\theta) \geq \begin{cases} \frac{2}{3} & \text{if} \quad \theta \geq \frac{1}{2} \\ \frac{5}{6} - \frac{\theta}{3} & \text{if} \quad \frac{1}{2} \geq \theta \geq \frac{2}{5} \\ \frac{9}{10} - \frac{\theta}{2} & \text{if} \quad \frac{2}{5} \geq \theta \geq 0. \end{cases}$$

3 Single-Pass $(2/5 - \varepsilon)$-Approximation Algorithm

In this section, we present a single-pass $(2/5 - \varepsilon)$-approximation algorithm for the problem (1). We first show that, if $c(o_1)$ is at most $K/2$ or more than $2/3K$, then the algorithm in [23] can be used. So we focus on the case when $c(o_1)$ is in $[K/2, 2/3K]$. For this case, we develop two algorithms by combining the technique in Sect. 2.2 into the thresholding algorithm in Sect. 2.1. The first one is useful when $c(o_2)$ is at most $K/3$, while the other is applied when $c(o_2)$ is more than $K/3$. Some proofs are omitted due to the page limitation.

In what follows, we often assume that we know in advance approximations of r_1 and r_2. That is, we are given $\underline{r}_\ell, \overline{r}_\ell$ such that $\underline{r}_\ell \leq r_\ell \leq \overline{r}_\ell$ and $\overline{r}_\ell \leq (1+\varepsilon)\underline{r}_\ell$ for $\ell \in \{1, 2\}$. These values can be guessed from a geometric series of some interval.

3.1 Algorithm When $c(o_1)$ is Small

It is known that when $c(o_1) \leq K/2$, we can improve the thresholding algorithm so that we can find a $(2/5 - \varepsilon)$-approximate solution in $O(K\varepsilon^{-4}\log^4 K)$ space with a single pass.

Theorem 3 ([23]). *Suppose that $c(o_1) \leq K/2$. We can find a $(2/5 - \varepsilon)$-approximate solution with a single pass for the problem (1). The space complexity of the algorithm is $O(K\varepsilon^{-4}\log^4 K)$.*

The above theorem can be extended to the case where we aim to find a set S of items that maximizes $f(S)$ subject to the relaxed constraint that the total size is at most pK, for a given number $p \geq 1$. We say that a set S of items is a (p, α)-*approximate solution* if $c(S) \leq pK$ and $f(S) \geq \alpha f(\mathrm{OPT})$, where OPT is an optimal solution of the original instance.

Theorem 4 ([23]). *For a constant number $p \geq 2r_1$, there is a $\left(p, \frac{2p}{2p+3} - \varepsilon\right)$-approximation streaming algorithm with a single pass. The space complexity of the algorithm is $O(K\varepsilon^{-3} \log^3 K)$.*

With the aid of this algorithm, we can find a $(2/5 - \varepsilon)$-approximate solution for some special cases even when $c(o_1) \geq K/2$.

Corollary 1. *If $c(o_1) > 2/3K$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass. The space complexity of the algorithm is $O(K\varepsilon^{-3} \log^3 K)$.*

Corollary 2. *Suppose that $c(o_1) > K/2$. If $f(o_1) \leq 3/10 f(\mathrm{OPT})$ then we can find a $(2/5 - \varepsilon)$-approximate solution with a single pass. The space complexity of the algorithm is $O(K\varepsilon^{-3} \log^3 K)$.*

3.2 Algorithm for Small $c(o_2)$

By Theorem 3 and Corollary 1, we may assume that $c(o_1)$ is in $[K/2, 2/3K]$. In this section, we describe a single-pass algorithm that works well when $c(o_2)$ is small.

Suppose that we know in advance the approximate value v of $f(\mathrm{OPT})$, i.e., $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. This assumption can be removed with dynamic update technique using $O(\varepsilon^{-1} \log K)$ additional space in a similar way to [2,23,39].

In addition, we suppose that we are given θ_1 such that $\theta_1 v/(1+\varepsilon) \leq f(o_1) \leq \theta_1 v$, which is an approximation of $f(o_1)$. Define $E_1 = \{e \in E \mid c(e) \in [\underline{r}_1 K, \overline{r}_1 K], f(e) \in [\theta_1 v/(1+\varepsilon), \theta_1 v]\}$. We can assume that E is the disjoint union of E_1 and $\overline{E_1} = \{e \mid c(e) \leq \overline{r}_2 K\}$, as we can discard the other items. We note that $o_1 \in E_1$ and $\mathrm{OPT} - o_1 \subseteq \overline{E_1}$.

We propose a single-pass streaming algorithm, where the target approximation ratio is $\beta = 2/5$. The algorithm description is given in Algorithm 1.

In the algorithm, we basically run the thresholding algorithm for $\overline{E_1}$ to collect a set S of items. In the same pass in parallel, we try to find a subset $X \subseteq E_1$ that contains a good approximation of o_1, based on Lemma 2. That is, when an item e in E_1 arrives, we add e to X if $|X| < t+1$ and $f(e \mid X) \geq (\theta_1 - \lambda|X|)v$. Each time an item e is added to X, since e may be a good approximation of o_1, we create a new feasible set $S+e$, and start to run the thresholding algorithm to $S+e$ in parallel. Thus a feasible set is generated for each e in X, and the family of these feasible sets is maintained as $\mathcal{T}$ in the algorithm. We remark that, to guarantee the approximation ratio of the algorithm starting from $S+e$, we need to satisfy the thresholding condition for e in X as well (Line 7): $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$ for the current set S. Thus the above algorithm performs *dynamically* the thresholding algorithm to $\overline{E_1}$ and $\overline{E_1} + e$ for each $e \in X$.

Algorithm 1.

procedure Dynamic(v)
  $S := \emptyset$; $\mathcal{T} := \emptyset$; $X := \emptyset$.
  $\alpha := \beta \frac{1}{1-\overline{r}_2}$.
  Define t and λ from θ_1 by (4) and (2).
  while item e is arriving **do** ▷ First phase
    if $e \in E_1$ **then**
      if $|X| < t+1$ and $f(e \mid X) \geq (\theta_1 - \lambda|X|)v$ and $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$ **then**
        $\mathcal{T} := \mathcal{T} \cup \{S+e\}$ and $X := X + e$.
    else
      if $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$ and $c(S+e) \leq K$ **then** $S := S + e$.
      for each $T \in \mathcal{T}$ **do**
        if $f(e \mid T) \geq \alpha \frac{c(e)}{K} v$ and $c(T+e) \leq K$ **then** $\mathcal{T} := \mathcal{T} \setminus \{T\} \cup \{T+e\}$.
      if $c(S) \geq (1 - \overline{r}_1 - \overline{r}_2)K$ **then** $S'_0 := S$ **and break**.
  $S' := S'_0$.
  while item e is arriving **do** ▷ Second phase
    if $e \in E_1$ **then**
      if $f(S') < f(S'_0 + e)$ **then** $S' := S'_0 + e$.
    else
      if $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$ and $c(S+e) \leq K$ **then** $S := S + e$.
      for any $T \in \mathcal{T}$ **do**
        if $f(e \mid T) \geq \alpha \frac{c(e)}{K} v$ and $c(T+e) \leq K$ **then** $\mathcal{T} := \mathcal{T} \setminus \{T\} \cup \{T+e\}$.
  return the best one among $\{S, S'\} \cup \mathcal{T}$.

However, the above strategy does not work when the size of S becomes large. Indeed, as we perform the thresholding algorithm to $S+e$ for each $e \in X$, it is necessary that $S+e$ is feasible, that is, $c(S) \leq K - c(e)$ when e arrives. Moreover, since we have the additional condition $f(e \mid S) \geq \alpha \frac{c(e)}{K} v$ to pick an item to X, we may throw away an approximation of o_1 when $f(e \mid S)$ is small (even if Lemma 2 is applicable). To avoid them, we adopt another strategy when $c(S)$ becomes large. Let S'_0 be the set we have the first time when $c(S)$ is at least $(1 - \overline{r}_1 - \overline{r}_2)K$. It follows from Lemma 1 that $f(S'_0)$ is relatively large as (6) below. Moreover, since $c(S)$ is at most $(1 - \overline{r}_1)K$, we can add any item in E_1 to S'_0. In the rest of a stream, we just take one item $e \in E_1$ that maximizes $f(S'_0 + e)$. At the same time, we continue to run the thresholding algorithm to S and every set in $\mathcal{T}$. In the end, the algorithm returns the best one among all the candidates.

Theorem 5. *Suppose that $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. Then Algorithm* Dynamic(v) *returns a set S such that $c(S) \leq K$ and*

$$f(S) \geq \min\left\{\beta, 1 - \beta\frac{1}{1-\overline{r}_2}, \Gamma(\theta) - \beta\frac{1-r_1}{1-\overline{r}_2}\right\} v - O(\varepsilon)v. \tag{5}$$

The space complexity is $O(K)$.

Let $\tilde{S}$ be the final set of S, and $\tilde{S}'$ be the output obtained by adding one item in E_1 to S'_0 (Line 17). Let $\tilde{T}_e$ be the final set in $\mathcal{T}$ containing an item $e \in X$.

We remark that the sets $\tilde{S}$ and $\tilde{T}_e$ are obtained by adding an item satisfying the thresholding condition repeatedly. Also, $\tilde{S}'$ and $\tilde{T}_e$ contain exactly one item e in E_1.

It is not difficult to see that all the obtained sets are of size at most K. We note that $c(S'_0) < (1-\overline{r}_1)K$, since S'_0 is the set the first time the size exceeds $(1-\overline{r}_1-\overline{r}_2)K$ by adding an item of size at most $\overline{r}_2 K$.

In the rest of this subsection, we will show (5). We first claim that, by Lemma 1, we have

$$f(S'_0) \geq \alpha(1-\overline{r}_1-\overline{r}_2)v, \tag{6}$$

since $c(S'_0) \geq (1-\overline{r}_1-\overline{r}_2)K$. Let $\tilde{X}$ be the final set of X.

Lemma 3. *At the end of the algorithm, one of the following holds.*

- *There exists an item $e \in \tilde{X}$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$.*
- *It holds that $f(o_1 \mid \tilde{S}) < \alpha\overline{r}_1 v$.*
- *It holds that $f(\tilde{S}') \geq \beta v$.*

Proof. Suppose that o_1 arrives during the first while-loop. Let $X = \{e_1, e_2, \ldots, e_x\}$ be the set just before o_1 arrives such that items are sorted in the ordering of the addition. Then X satisfies that $f(e_i \mid \{e_1, e_2, \ldots, e_{i-1}\}) \geq (\theta - \lambda(i-1))v$ for each $i = 1, \ldots, x$. Note that, when o_1 will be contained in X, clearly the first statement holds. Thus we may assume that o_1 does not satisfy the condition in Line 7, that is, one of the following three conditions holds: $|X| = t+1$, $f(o_1 \mid X) < (\theta - \lambda|X|)v$, and $f(o_1 \mid S) < \alpha c(o_1)v \leq \alpha\overline{r}_1 v$. It follows from Lemma 2 that, if one of the first two conditions holds, then at least one item $\overline{e} \in X$ satisfies $f(\mathrm{OPT} - o_1 + \overline{e}) \geq \Gamma(\theta)v - O(\varepsilon)v$. If $f(o_1 \mid S) < \alpha\overline{r}_1 v$, then $f(o_1 \mid \tilde{S}) \leq f(o_1 \mid S) < \alpha\overline{r}_1 v$ by submodularity. Thus one of the first two statements of Lemma 3 is satisfied since $X \subseteq \tilde{X}$.

Next suppose that o_1 arrives after constructing S'_0, and suppose that $f(\tilde{S}') < \beta v$. From Line 17, we see that $f(S'_0 + o_1) \leq f(\tilde{S}') < \beta v$. Hence we have

$$f(o_1 \mid S'_0) = f(S'_0 + o_1) - f(S'_0) < \beta v - f(S'_0).$$

By (6), it holds that

$$f(o_1 \mid S'_0) < \beta v - \alpha(1-\overline{r}_1-\overline{r}_2)v \leq \alpha\overline{r}_1 v = \beta\frac{\overline{r}_1}{1-\overline{r}_2}v,$$

where we recall $\beta = \alpha(1-\overline{r}_2)$. Therefore, by submodularity, we obtain $f(o_1 \mid \tilde{S}) \leq f(o_1 \mid S'_0) \leq \alpha\overline{r}_1 v$. Thus the lemma follows. □

We then show that, for each case of Lemma 3, the approximation ratio can be bounded as below. Combining all the cases, we can prove Theorem 5.

Lemma 4. *Suppose that there exists $e \in \tilde{X}$ such that $f(\mathrm{OPT} - o_1 + e) \geq \Gamma(\theta)v - O(\varepsilon)v$. Then it holds that*

$$f(\tilde{T}_e) \geq \min\left\{\beta, \Gamma(\theta) - \beta\frac{1-r_1}{1-\overline{r}_2}\right\}v - O(\varepsilon)v.$$

Lemma 5. *If $f(o_1 \mid \tilde{S}) < \alpha \overline{r}_1 v$, then we have $f(\tilde{S}) \geq \min\{\beta, 1-\alpha\}v - O(\varepsilon)v$.*

It turns out from Theorem 5 that the algorithm works well when $c(o_2)$ is small or $f(o_2)$ is small.

Corollary 3. *Suppose that v satisfies $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. If $c(o_1) \geq K/2$ and $c(o_2) \leq K/3$, then we find a set S such that $c(S) \leq K$ and $f(S) \geq (2/5 - O(\varepsilon))v$. The space complexity is $O(K\varepsilon^{-3}\log K)$.*

Corollary 4. *Suppose that $2K/3 \geq c(o_1) > K/2$ and $c(o_2) > K/3$. If $f(o_2) < 1/5 f(\mathrm{OPT})$, then we can find a set S such that $c(S) \leq K$ and $f(S) \geq (2/5 - O(\varepsilon))f(\mathrm{OPT})$. The space complexity of the algorithm is $O(K\varepsilon^{-4}\log^3 K)$.*

In summary, we have the following theorem, together with the dynamic update technique to guess the approximate value v of $f(\mathrm{OPT})$.

Theorem 6. *If $c(o_1) \geq K/2$ and $c(o_2) \leq K/3$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass. The space complexity is $O(K\varepsilon^{-4}\log^2 K)$.*

3.3 Algorithm for Large $c(o_2)$

In this section, we propose our second algorithm that is efficient when $c(o_2)$ is large. Since $o_1, o_2 \in \mathrm{OPT}$, it is clear that $c(o_1) + c(o_2) \leq K$, and hence $r_2 \leq 1 - r_1$. We here assume that $r_2 \leq 1 - r_1 - \varepsilon$, where the other case when r_2 is too large is easier as in the following lemma. Note that the assumption implies that $\overline{r}_1 + \overline{r}_2 \leq 1$.

Lemma 6. *If $c(o_1) + c(o_2) \geq (1-\varepsilon)K$, then we can find a $(2/5-\varepsilon)$-approximate solution with a single pass using $O(K\varepsilon^{-3}\log^3 K)$ space.*

Similarly to the previous section, we assume that we know in advance the approximate value v of $f(\mathrm{OPT})$, i.e., $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. This assumption can be removed using $O(\varepsilon^{-1}\log K)$ additional space. We also assume that we are given θ_ℓ such that $\theta_\ell v/(1+\varepsilon) \leq f(o_\ell) \leq \theta_\ell v$ for $\ell \in \{1,2\}$. Define $E_\ell = \{e \in E \mid c(e) \in [\underline{r}_\ell K, \overline{r}_\ell K], f(e) \in [\theta_\ell v/(1+\varepsilon), \theta_\ell v]\}$ for $\ell \in \{1,2\}$. Then $o_\ell \in E_\ell$ holds. We can assume that E is the union of E_1, E_2 and $\overline{E} = \{e \mid c(e) \leq \overline{r}_2 K\}$, as we can discard the other items.

In the algorithm, we perform the thresholding algorithm and the procedure PickOneItem in Sect. 2.2 *in parallel*. We apply PickOneItem to both E_1 and E_2 to obtain approximations of o_1 and o_2. Then X_ℓ includes an approximation of o_ℓ for $\ell = 1, 2$. While finding X_1 and X_2, we check in Line 11 whether there exists a pair of items each from X_1 and X_2, respectively, whose f-value is more than βv. In parallel, we run the thresholding algorithm with α_ℓ to $\overline{E}$ to obtain a set S_ℓ, where $\alpha_\ell := \frac{\beta}{1-\overline{r}_\ell}$, for $\ell = 1, 2$. If the output S_ℓ has large size, then Lemma 1 guarantees that $f(S_\ell)$ is large. Otherwise, $c(S_\ell)$ is small, meaning that there is a room for adding an item from X_ℓ. The algorithm returns the set that maximizes $f(S_\ell + e)$ for $e \in X_\ell$ and $\ell = 1, 2$. Intuitively, the algorithm partitions the ground set E into three parts E_1, E_2 and $\overline{E}$, and then it returns the best set that can be obtained from two of the three parts.

Algorithm 2.

1: **procedure** Parallel(v)
2: $S_\ell := \emptyset$; $X_\ell := \emptyset$ for $\ell = 1, 2$.
3: $\alpha_\ell := \frac{\beta}{1-\overline{r}_\ell}$ for $\ell = 1, 2$.
4: Define t_ℓ and λ_ℓ from θ_ℓ by (4) and (2) for $\ell = 1, 2$.
5: **while** item e is arriving **do**
6: **for** each $\ell \in \{1, 2\}$ **do**
7: **if** $e \in E_\ell$ **then**
8: **if** $|X_\ell| < t_\ell + 1$ and $f(e \mid X_\ell) \geq (\theta_\ell - \lambda_\ell |X_\ell|)v$ **then**
9: $X_\ell := X_\ell + e$.
10: **else if** $e \in E_1 \cup E_2 \setminus E_\ell$ **then**
11: **if** there exists an item $\bar{e} \in X_\ell$ such that $f(\{\bar{e}, e\}) \geq \beta v$ **then return** $\{\bar{e}, e\}$.
12: **else**
13: **if** $f(e \mid S_\ell) \geq \alpha_\ell \frac{c(e)}{K} v$ and $c(S_\ell + e) \leq K$ **then** $S_\ell := S_\ell + e$.
14: **if** $c(S_\ell) \geq (1 - \overline{r}_\ell)K$ for some $\ell \in \{1, 2\}$ **then return** S_ℓ.
15: **else return** the set that achieves $\max_{\ell \in \{1,2\}, e \in X_\ell} f(S_\ell + e)$.

Theorem 7. *Suppose that $v \leq f(\mathrm{OPT}) \leq (1 + \varepsilon)v$. If $r_1 + r_2 \leq 1 - \varepsilon$, then Algorithm* Parallel$v$ *returns a set S such that $c(S) \leq K$ and $f(S) \geq \gamma v - O(\varepsilon)v$, where*

$$\gamma = \min \left\{ \beta, \Gamma(\theta_2) + \theta_2 - \beta \frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}, \Gamma(\theta_1) + \theta_1 - \beta \frac{2 - 2r_1 - r_2}{1 - \overline{r}_1} \right\}. \quad (7)$$

The space complexity is $O(K)$.

Let $\tilde{S}_\ell$ ($\ell = 1, 2$) be the final set of S_ℓ in the algorithm. We also denote by $\tilde{X}_\ell$ the final set of X_ℓ. Let $\tilde{S}'_\ell$ be the set that achieves $\max_{e \in \tilde{X}_\ell} f(\tilde{S}_\ell + e)$ for $\ell = 1, 2$. The set $\tilde{S}_\ell$ is obtained by adding an item based on the thresholding condition $f(e \mid S_\ell) \geq \alpha_\ell \frac{c(e)}{K} v$.

In the algorithm, each item in $\overline{E}$ is added to S_1 or S_2 only when it does not exceed the knapsack capacity. Hence $c(\tilde{S}_\ell) \leq K$ for $\ell = 1, 2$. Also clearly $c(\tilde{S}'_\ell) \leq K$ for $\ell = 1, 2$ if $c(\tilde{S}_\ell) \leq (1 - \overline{r}_\ell)K$. On the other hand, if the algorithm terminates in Line 11, then the output has only two items each from E_1 and E_2, and hence the size is at most K since $\overline{r}_1 + \overline{r}_2 \leq 1$ by the assumption. Thus the output of the algorithm is of size at most K.

From now on, we will prove (7). We consider the following two cases separately: the case when o_2 arrives before o_1 and when o_1 arrives before o_2.

Case 1: Suppose that o_2 Arrives Before o_1. We consider the case when $\ell = 2$. We may assume that the algorithm terminates in the end (not in Line 11). Moreover, if $c(\tilde{S}_2) \geq (1 - \overline{r}_2)K$, then $f(\tilde{S}_2) \geq \beta v$ by Lemma 1. Thus we may assume that $c(\tilde{S}_2) < (1 - \overline{r}_2)K$.

Let $X_2 = \{e_1, e_2, \ldots, e_x\}$ be the set collected just before o_2 arrives. Then X_2 satisfies that $f(e_j \mid \{e_1, e_2, \ldots, e_{j-1}\}) \geq (\theta_2 - \lambda_2(j - 1))$ for each $j = 1, \ldots, x$.

When o_1 arrives, we return the set $\{e, o_1\}$ for some $e \in X_2$ if $f(\{o_1, e\}) \geq \beta v$ at Line 11. Thus we may assume that $f(\{o_1, e\}) < \beta v$ for any $e \in X_2$.

Lemma 7. *Suppose that $c(\tilde{S}_2) < (1 - \overline{r}_2)K$ and that, for any $e \in X_2$, we have $f(\{o_1, e\}) < \beta v$. There exists an item $e \in X_2$ such that*

$$f(\tilde{S}_2 + e) \geq \Gamma(\theta_2)v + \theta_2 v - \beta \frac{2 - 2r_2 - r_1}{1 - \overline{r}_2} v.$$

Proof. By Lemma 2, we have $e \in X_2$ such that $f(\mathrm{OPT} - o_2 + e) \geq \Gamma(\theta_2)v - O(\varepsilon)v$. Since $f(\{o_1, e\}) < \beta v$ and $f(e) \geq \theta_2 v$, we see that $f(o_1 \mid e) = f(\{o_1, e\}) - f(e) < (\beta - \theta_2)v$. It then holds by submodularity that

$$\begin{aligned} f(\mathrm{OPT} - o_2 + e) &\leq f(e) + f(o_1 \mid e) + f(\mathrm{OPT} - o_1 - o_2 \mid e) \\ &\leq (\beta - \theta_2)v + f(\mathrm{OPT} - o_1 - o_2 + e). \end{aligned}$$

Hence, since $f(\mathrm{OPT} - o_2 + e) \geq \Gamma(\theta_2)v - O(\varepsilon)v$,

$$\Gamma(\theta_2)v - (\beta - \theta_2)v - O(\varepsilon)v \leq f(\mathrm{OPT} - o_1 - o_2 + e).$$

On the other hand, it follows from submodularity that

$$\begin{aligned} f(\mathrm{OPT} - o_1 - o_2 + e) &\leq f(\tilde{S}_2 + e) + f(\mathrm{OPT} - o_1 - o_2 - \tilde{S}_2 \mid \tilde{S}_2 + e) \\ &\leq f(\tilde{S}_2 + e) + f(\mathrm{OPT} - o_1 - o_2 - \tilde{S}_2 \mid \tilde{S}_2) \\ &\leq f(\tilde{S}_2 + e) + \alpha_2(1 - r_1 - r_2)v, \end{aligned}$$

where the last inequality follows from the fact that, since $c(\tilde{S}_2) \leq (1 - c(o_2))K$, any item $o \in \mathrm{OPT} - o_1 - o_2 - \tilde{S}_2$ is discarded due to the thresholding condition, implying $f(o \mid \tilde{S}_2) \leq \alpha_2 c(o)v$. Combining them, we obtain

$$\begin{aligned} f(\tilde{S}_2 + e) &\geq \left(\Gamma(\theta_2) - (\beta - \theta_2) - \alpha(1 - r_1 - r_2)\right) v - O(\varepsilon)v \\ &\geq \left(\Gamma(\theta_2) + \theta_2 - \beta \frac{2 - 2r_2 - r_1}{1 - \overline{r}_2}\right) v - O(\varepsilon)v. \end{aligned}$$

□

Case 2: Suppose that o_1 Arrives Before o_2. Let X_1 be the set just before o_1 arrives. We can use the symmetrical argument to Case 1. We omit the proof.

Lemma 8. *Suppose that $c(\tilde{S}_1) < (1 - \overline{r}_1)K$ and that, for any $e \in X_1$, we have $f(\{o_2, e\}) < \beta v$. There exists an item $e \in X_1$ such that*

$$f(\tilde{S}_1 + e) \geq \Gamma(\theta_1)v + \theta_1 v - \beta \frac{2 - 2r_1 - r_2}{1 - \overline{r}_1} v.$$

Combining the above two lemmas, we have Theorem 7. It follows from Theorem 7 that the algorithm admits a $(2/5 - \varepsilon)$-approximation when $r_2 \geq 1/3$.

Corollary 5. *Suppose that* $v \leq f(\mathrm{OPT}) \leq (1+\varepsilon)v$. *Consider the case when* $K/2 < c(o_1) \leq 2/3K$ *and* $K/3 \leq c(o_2) < (1-r_1-\varepsilon)K$. *We can find a set* S *such that* $c(S) \leq K$ *and* $f(S) \geq (2/5 - O(\varepsilon))v$. *The space complexity of the algorithm is* $O(K\varepsilon^{-2})$.

Theorem 8. *If* $c(o_1) \geq K/2$ *and* $K/3 \leq c(o_2) < (1-r_1-\varepsilon)K$, *then we can find a* $(2/5-\varepsilon)$*-approximate solution with a single pass using* $O(K\varepsilon^{-3}\log K)$ *space.*

Theorem 1 follows from Theorem 6, Lemma 6, and Theorem 8.

References

1. Alon, N., Gamzu, I., Tennenholtz, M.: Optimizing budget allocation among channels and influencers. In: WWW, pp. 381–388 (2012)
2. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: Streaming submodular maximization: massive data summarization on the fly. In: KDD, pp. 671–680 (2014)
3. Badanidiyuru, A., Vondrák, J.: Fast algorithms for maximizing submodular functions. In: SODA, pp. 1497–1514 (2013)
4. Balkanski, E., Rubinstein, A., Singer, Y.: An exponential speedup in parallel running time for submodular maximization without loss in approximation. In: SODA, pp. 283–302 (2019)
5. Balkanski, E., Singer, Y.: The adaptive complexity of maximizing a submodular function. In: STOC, STOC 2018, pp. 1138–1151 (2018)
6. Barbosa, R.D.P., Ene, A., Nguyen, H.L., Ward, J.: A new framework for distributed submodular maximization. In: FOCS, pp. 645–654 (2016)
7. Barbosa, R., Ene, A., Le Nguyen, H., Ward, J.: The power of randomization: Distributed submodular maximization on massive datasets. In: ICML, ICML 2015, pp. 1236–1244 (2015). JMLR.org
8. Bateni, M., Esfandiari, H., Mirrokni, V.: Almost optimal streaming algorithms for coverage problems. In: SPAA, SPAA 2017, pp. 13–23. ACM, New York (2017)
9. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a monotone submodular function subject to a matroid constraint. SIAM J. Comput. **40**(6), 1740–1766 (2011)
10. Chakrabarti, A., Kale, S.: Submodular maximization meets streaming: matchings, matroids, and more. Math. Program. **154**(1–2), 225–247 (2015)
11. Chan, T.H.H., Huang, Z., Jiang, S.H.C., Kang, N., Tang, Z.G.: Online submodular maximization with free disposal: Randomization beats for partition matroids online. In: SODA, pp. 1204–1223 (2017)
12. Chan, T.H.H., Jiang, S.H.C., Tang, Z.G., Wu, X.: Online submodular maximization problem with vector packing constraint. In: ESA (2017)
13. Chekuri, C., Gupta, S., Quanrud, K.: Streaming algorithms for submodular function maximization. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9134, pp. 318–330. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47672-7_26
14. Chekuri, C., Quanrud, K.: Submodular function maximization in parallel via the multilinear relaxation. In: SODA, pp. 303–322 (2019)
15. Chekuri, C., Vondrák, J., Zenklusen, R.: Submodular function maximization via the multilinear relaxation and contention resolution schemes. SIAM J. Comput. **43**(6), 1831–1879 (2014)

16. Ene, A., Nguyễn, H.L.: A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In: ICALP (2019)
17. Ene, A., Nguyễn, H.L.: Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In: SODA, pp. 274–282 (2019)
18. Feldman, M., Karbasi, A., Kazemi, E.: Do less, get more: streaming submodular maximization with subsampling. In: NeurIPS 2018, pp. 730–740 (2018)
19. Filmus, Y., Ward, J.: A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. SIAM J. Comput. **43**(2), 514–542 (2014)
20. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions I. Math. Program. **14**, 265–294 (1978)
21. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions II. Math. Program. Study **8**, 73–87 (1978)
22. Huang, C., Kakimura, N.: Multi-pass streaming algorithms for monotone submodular function maximization (2018). arXiv http://arxiv.org/abs/1802.06212
23. Huang, C.C., Kakimura, N., Yoshida, Y.: Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. In: APPROX (2017)
24. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
25. Krause, A., Singh, A.P., Guestrin, C.: Near-optimal sensor placements in gaussian processes: theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. **9**, 235–284 (2008)
26. Kulik, A., Shachnai, H., Tamir, T.: Maximizing submodular set functions subject to multiple linear constraints. In: SODA, pp. 545–554 (2013)
27. Kumar, R., Moseley, B., Vassilvitskii, S., Vattani, A.: Fast greedy algorithms in mapreduce and streaming. ACM Trans. Parallel Comput. **2**(3), 14:1–14:22 (2015)
28. Lee, J.: Maximum Entropy Sampling, Encyclopedia of Environmetrics, vol. 3, pp. 1229–1234. Wiley (2006)
29. Lee, J., Sviridenko, M., Vondrák, J.: Submodular maximization over multiple matroids via generalized exchange properties. Math. Oper. Res. **35**(4), 795–806 (2010)
30. Lin, H., Bilmes, J.: Multi-document summarization via budgeted maximization of submodular functions. In: NAACL-HLT, pp. 912–920 (2010)
31. Lin, H., Bilmes, J.: A class of submodular functions for document summarization. In: ACL-HLT, pp. 510–520 (2011)
32. McGregor, A.: Graph stream algorithms: a survey. SIGMOD Rec. **43**(1), 9–20 (2014)
33. McGregor, A., Vu, H.T.: Better streaming algorithms for the maximum coverage problem. In: ICDT (2017)
34. Mirzasoleiman, B., Jegelka, S., Krause, A.: Streaming non-monotone submodular maximization: personalized video summarization on the fly. In: AAAI (2018)
35. Soma, T., Kakimura, N., Inaba, K., Kawarabayashi, K.: Optimal budget allocation: theoretical guarantee and efficient algorithm. In: ICML, pp. 351–359 (2014)
36. Sviridenko, M.: A note on maximizing a submodular set function subject to a knapsack constraint. Oper. Res. Lett. **32**(1), 41–43 (2004)
37. Wolsey, L.: Maximising real-valued submodular functions: primal and dual heuristics for location problems. Math. Oper. Res. **7**, 410–425 (1982)
38. Yoshida, Y.: Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint (2016). https://arxiv.org/abs/1607.04527
39. Yu, Q., Xu, E.L., Cui, S.: Streaming algorithms for news and scientific literature recommendation: submodular maximization with a d-knapsack constraint. In: IEEE Global Conference on Signal and Information Processing (2016)